

Credit Score Classification (Multi-Class): Optimizing Financial Decision-Making

Deep Learning Techniques: Artificial Neural Network (ANN) Model

Duygu Jones | Data Scientist | Sep 2024

Follow me: duygujones.com | [Linkedin](#) | [GitHub](#) | [Kaggle](#) | [Medium](#) | [Tableau](#)



Project Overview: Credit Score Classification Using ANN

This project focuses on building a robust **Artificial Neural Network (ANN)** model to predict customer credit scores based on various attributes, including demographic, financial, and behavioral data.

The objective is to classify customers into three categories: **Good**, **Standard**, and **Poor** credit scores, assisting financial institutions in making informed lending decisions.

Importance of Credit Score in Finance

A **credit score** reflects a customer's creditworthiness, indicating how likely they are to repay debts on time.

Financial institutions use credit scores to assess risk, set interest rates, and decide loan terms.

Customers with **Good** credit scores are seen as low-risk and typically receive better financial terms, while those with **Poor** credit scores are considered high-risk, potentially facing higher interest rates or loan rejections.

Categorizing customers based on their credit scores helps lenders make faster, data-driven decisions.

Key Steps in the Project

Before training the **ANN model**, the following steps are performed to prepare the dataset:

- Handling missing and unusual values:** Addressing gaps and anomalies in the dataset through imputation and cleaning techniques.
- Encoding categorical features:** Converting categorical data, such as payment behavior, into numerical values for model compatibility.
- Scaling numerical features:** Ensuring features like **Annual Income** and **Outstanding Debt** are normalized to improve model performance.
- Data Cleaning:** Removing unnecessary columns, such as **ID** and **Name**, that do not contribute predictive value.

5. **Model Development (ANN):** The **ANN** captures complex, non-linear relationships between features like **Outstanding Debt, Loan Amount, and Payment Behavior** to predict the customer's **Credit Score**.
6. **Business Application:** By accurately predicting credit scores, the model supports financial institutions in making better lending decisions, setting appropriate interest rates, and segmenting customers for targeted financial products.

Why ANN for Credit Score Prediction?

Artificial Neural Networks are ideal for this task due to their ability to identify patterns and relationships in complex data. Credit scoring involves multiple interconnected financial factors, and an ANN can capture these interactions to deliver accurate credit score predictions.

Business Application

- **Loan Approvals:** By predicting a customer's credit score, the ANN model helps financial institutions assess the risk of lending to an individual.
- **Interest Rates:** Customers with higher credit scores receive more favorable loan terms, while those with lower scores are considered higher risk.
- **Risk Management:** Accurate credit score predictions enable banks to reduce financial risks and make better lending decisions, ultimately leading to better customer segmentation and optimized loan offerings.

Conclusion

This project highlights the crucial role that **credit scores** play in both financial decision-making and risk assessment. The use of an **ANN model** enables the accurate classification of customers based on their financial behavior, supporting real-world applications such as loan approval processes and interest rate determination. By leveraging deep learning techniques, we aim to create a highly effective model that can assist financial institutions in managing risk and improving customer service.

About the Dataset

The dataset consists of customer data, including demographic information, financial history, and payment behavior, which are important factors in determining credit scores. This data will be processed, cleaned, and engineered to extract the most important features for training the ANN model.

The target feature is the **Credit Score**, which classifies customers into three categories based on their financial history:

1. **Good:** Low-risk customers with strong financial management.
 2. **Standard:** Average-risk customers with moderate financial reliability.
 3. **Poor:** High-risk customers who may struggle with debt repayments.
- **Dataset:** Credit Score Classification Dataset
 - **Content:** Customer demographic, financial, and credit history details.
 - **Number of Rows:** 100.000
 - **Number of Columns:** 28

No	INPUTS	Description
1	ID	Unique identifier for each record.
2	Customer_ID	Unique identifier for each customer.
3	Month	Month of the transaction or record.
4	Name	Customer's name.
5	Age	The customer's age.
6	SSN	Customer's social security number.
7	Occupation	The customer's occupation.
8	Annual_Income	The customer's annual income.
9	Monthly_Inhand_Salary	The customer's monthly in-hand salary.

No	INPUTS	Description
10	Num_Bank_Accounts	Number of bank accounts owned by the customer.
11	Num_Credit_Card	Number of credit cards owned by the customer.
12	Interest_Rate	The interest rate applied to loans or credit.
13	Num_of_Loan	Number of loans taken by the customer.
14	Type_of_Loan	Type of loan taken by the customer.
15	Delay_from_due_date	The delay in payment from the due date.
16	Num_of_Delayed_Payment	Number of delayed payments made by the customer.
17	Changed_Credit_Limit	Changes made to the customer's credit limit.
18	Num_Credit_Inquiries	Number of credit inquiries made.
19	Credit_Mix	The mix of credit types the customer uses (e.g., loans, credit cards).
20	Outstanding_Debt	Total outstanding debt the customer has.
21	Credit_Utilization_Ratio	The ratio of credit used to the total credit limit.
22	Credit_History_Age	The length of the customer's credit history.
23	Payment_of_Min_Amount	Whether the customer pays the minimum amount required each month.
24	Total_EMI_per_month	The total EMI (Equated Monthly Installment) the customer pays each month.
25	Amount_invested_monthly	The amount invested by the customer each month.
26	Payment_Behaviour	The payment behavior of the customer.
27	Monthly_Balance	The customer's remaining balance at the end of each month.
28	Credit_Score	The customer's credit score (target variable: "Good," "Poor," "Standard").

- This dataset is ideal for developing credit risk assessment models, allowing for a detailed analysis of the factors that impact an individual's credit score.
- Original dataset is available on Kaggle: [Credit score classification](#)

Further Summary of the Project:

- **Comprehensive EDA:** Addressed missing and unusual values while retaining outliers to maintain the data's natural characteristics.
- **Preprocessing:** Meticulously encoded categorical columns and employed pipelines to prevent data leakage, ensuring data integrity.
- **Data Scaling:** Appropriately scaled the data to prepare for modeling.
- **Handling Imbalanced Classes:** Applied both SMOTE and class weighting techniques, with SMOTE proving more effective.
- **Model Optimization:** Iteratively optimized the ANN model architecture across eight configurations.
- **Final Model Selection:** Chose the ANN-7 model with SMOTE for its robust performance.
- **Model Testing:** Made predictions on the test dataset to validate the model's effectiveness.

For additional details about each model tested in this project, please visit the repository on my [GitHub profile](#).

Table of Contents:

1. Setup and Initialization
 - 1.1 Import the Libraries
 - 1.2 Load the Dataset
2. Exploratory Data Analysis (EDA)
3. Data Cleaning and Transformation
 - 3.1 Handling Missing Values and Cleaning
 - 3.2 Dropping Unnecessary Features
 - 3.3 Outliers

- 4. Data Preprocessing
 - 4.1 Encode Categoricals
 - 4.2 Train-Test Split
 - 4.3 Scale: MinMaxScaler
 - 4.4 SMOTE for Imbalanced Data
- 5. Correlation Analysis
- 6. ANN Model
- 7. Final Model
 - 7.1 Prediction with the Test Data
- 8. Project Summary

Setup and Initialization

Import the Libraries

```
In [1]: #!pip install missingno
```

```
In [2]: import numpy as np
import pandas as pd
import seaborn as sns
import missingno
import matplotlib.pyplot as plt

%matplotlib inline
# %matplotlib notebook
plt.rcParams["figure.figsize"] = (12, 6)
# plt.rcParams['figure.dpi'] = 100
sns.set_style("whitegrid")
import warnings

warnings.filterwarnings("ignore")
warnings.warn("this will not show")
pd.set_option('display.float_format', lambda x: '%.3f' % x)
```

```
In [3]: # Pre-Processing
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score, cross_validate
# Metrics
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import roc_auc_score, roc_curve, precision_recall_curve, average_precision_score

# Model relevant Libraries
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout, BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.saving import save_model
from keras.optimizers import Adam
from keras.regularizers import l2
```

Load the Dataset

```
In [4]: train0 = pd.read_csv('train.csv')
train = train0.copy()

train.head(3)
```

Out[4]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_B
0	0x1602	CUS_0xd40	January	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843	
1	0x1603	CUS_0xd40	February	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	NaN	
2	0x1604	CUS_0xd40	March	Aaron Maashoh	-500	821-00-0265	Scientist	19114.12	NaN	

3 rows × 28 columns

◀	▶
---	---

In [5]:

```
test0 = pd.read_csv('test.csv')
test = test0.copy()

test.head(3)
```

Out[5]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_B
0	0x160a	CUS_0xd40	September	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843	
1	0x160b	CUS_0xd40	October	Aaron Maashoh	24	821-00-0265	Scientist	19114.12	1824.843	
2	0x160c	CUS_0xd40	November	Aaron Maashoh	24	821-00-0265	Scientist	19114.12	1824.843	

3 rows × 27 columns

◀	▶
---	---

In [6]:

```
# train + test data all together
df0 = pd.concat([train,test], sort=False).reset_index(drop=True)
df = df0.copy()

df.head(3)
```

Out[6]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_B
0	0x1602	CUS_0xd40	January	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843	
1	0x1603	CUS_0xd40	February	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	NaN	
2	0x1604	CUS_0xd40	March	Aaron Maashoh	-500	821-00-0265	Scientist	19114.12	NaN	

3 rows × 28 columns

◀	▶
---	---

In [7]:

```
print('Train Data Shape:', train.shape)
print('Test Data Shape:', test.shape)
```

Train Data Shape: (100000, 28)
Test Data Shape: (50000, 27)

Exploratory Data Analysis (EDA)

In [8]: `train.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 28 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   ID               100000 non-null   object  
 1   Customer_ID      100000 non-null   object  
 2   Month            100000 non-null   object  
 3   Name              90015 non-null   object  
 4   Age               100000 non-null   object  
 5   SSN               100000 non-null   object  
 6   Occupation        100000 non-null   object  
 7   Annual_Income    100000 non-null   object  
 8   Monthly_Inhand_Salary 84998 non-null   float64 
 9   Num_Bank_Accounts 100000 non-null   int64  
 10  Num_Credit_Card   100000 non-null   int64  
 11  Interest_Rate    100000 non-null   int64  
 12  Num_of_Loan       100000 non-null   object  
 13  Type_of_Loan     88592 non-null   object  
 14  Delay_from_due_date 100000 non-null   int64  
 15  Num_of_Delayed_Payment 92998 non-null   object  
 16  Changed_Credit_Limit 100000 non-null   object  
 17  Num_Credit_Inquiries 98035 non-null   float64 
 18  Credit_Mix        100000 non-null   object  
 19  Outstanding_Debt 100000 non-null   object  
 20  Credit_Utilization_Ratio 100000 non-null   float64 
 21  Credit_History_Age 90970 non-null   object  
 22  Payment_of_Min_Amount 100000 non-null   object  
 23  Total_EMI_per_month 100000 non-null   float64 
 24  Amount_invested_monthly 95521 non-null   object  
 25  Payment_Behaviour 100000 non-null   object  
 26  Monthly_Balance   98800 non-null   object  
 27  Credit_Score       100000 non-null   object  
dtypes: float64(4), int64(4), object(20)
memory usage: 21.4+ MB
```

In [9]: `# Display df.shape, duplicate and missing values count`

```
print(f'Data shape (rows, columns): {df.shape}')
print(f'Number of total duplicate rows: {df.duplicated().sum()}')
print(f'Number of missing values in Train: {train.isnull().sum().sum()}')
print(f'Number of missing values in Test: {test.isnull().sum().sum()}')
```

```
Data shape (rows, columns): (150000, 28)
Number of total duplicate rows: 0
Number of missing values in Train: 60071
Number of missing values in Test: 30053
```

In [10]: `# ===== User-Defined-Function =====`

```
# Summary of Categorical Features
def object_summary(df):
    obs = df.shape[0]

    object_df = df.select_dtypes(include='object')
    summary_df = pd.DataFrame({
        'Dtype': object_df.dtypes,
        'Counts': object_df.apply(lambda x: x.count()),
        'Nulls': object_df.apply(lambda x: x.isnull().sum()),
        'NullPercent': (object_df.isnull().sum() / obs) * 100,
        'Top': object_df.apply(lambda x: x.mode()[0] if not x.mode().empty else '-'),
        'Frequency': object_df.apply(lambda x: x.value_counts().max() if not x.value_counts().empty else '-'),
        'Uniques': object_df.apply(lambda x: x.unique().shape[0]),
        'UniqueValues': object_df.apply(lambda x: list(x.unique()) if x.unique() <= 20 else '-')
    })

    # Format 'NullPercent' to show percentages with two decimal points and a '%' sign
    summary_df['NullPercent'] = summary_df['NullPercent'].map("{:.2f}%".format)

    print('Categorical Features Summary:')
    print('_____ \nData Types:')
    print(summary_df['Dtype'].value_counts())
    print('_____')
```

```
return summary_df
```

```
object_summary(df)
```

Categorical Features Summary:

Data Types:

Dtype

object 20

Name: count, dtype: int64

Out[10]:

```
In [11]: # ===== User-Defined-Function =====  
# Summary of Numeric Features
```

```
def numeric_summary(df):  
    obs = df.shape[0]
```

```
numeric_df = df.select_dtypes(include='number')
summary_df = pd.DataFrame({
    'Dtype': numeric_df.dtypes,
    'Counts': numeric_df.apply(lambda x: x.count()),
    'Nulls': numeric_df.apply(lambda x: x.isnull().sum()),
    'NullPercent': (numeric_df.isnull().sum() / obs) * 100,
    'Min': numeric_df.min(),
    'Max': numeric_df.max(),
    'Uniques': numeric_df.apply(lambda x: x.unique().shape[0]),
    'UniqueValues': numeric_df.apply(lambda x: list(x.unique()) if x.nunique() <= 20 else '-')
})
```

```
# Format 'NullPercent' to show percentages with two decimal points and a '%' sign
summary_dsf[!NullPercent!] = summary_dsf[!NullPercent!] %>% format("%,.2f")
```

```

print('Numreical Features Summary:')
print('_____ \nData Types:')
print(summary_df['Dtype'].value_counts())
print('_____')

return summary_df

numeric_summary(df)

```

Numreical Features Summary:

Data Types:

Dtype	Count	Name
float64	4	
int64	4	
		Name: count, dtype: int64

	Dtype	Counts	Nulls	NullPercent	Min	Max	Uniques	UniqueValues
Monthly_Inhand_Salary	float64	127500	22500	15.00%	303.645	15204.633	13684	-
Num_Bank_Accounts	int64	150000	0	0.00%	-1.000	1798.000	1183	-
Num_Credit_Card	int64	150000	0	0.00%	0.000	1499.000	1344	-
Interest_Rate	int64	150000	0	0.00%	1.000	5799.000	2394	-
Delay_from_due_date	int64	150000	0	0.00%	-5.000	67.000	73	-
Num_Credit_Inquiries	float64	147000	3000	2.00%	0.000	2597.000	1608	-
Credit_Utilization_Ratio	float64	150000	0	0.00%	20.000	50.000	150000	-
Total_EMI_per_month	float64	150000	0	0.00%	0.000	82398.000	16960	-

Data Cleaning and Transformation

Based on the dataset's summary above, the following data cleaning steps are necessary:

- Handle Missing Values:** Columns like `Name`, `Type_of_Loan`, `Num_of_Delayed_Payment`, `Credit_History_Age`, `Amount_invested_monthly`, `Monthly_Balance`, and `Credit_Score` contain missing values. Imputation or removal of these rows might be required depending on the model.
- Fix Anomalies:** Correct unrealistic values in columns such as `Age` (e.g., -500) and `SSN` (e.g., invalid characters like #@%).
- Handle Duplicates:** There are no duplicated values in the dataset, so this step is not necessary.
- Outliers Treatment:** Handle outliers in numeric columns such as `Interest_Rate`, `Delay_from_due_date`, and `Total_EMI_per_month` to improve model performance.
- Encoding:** Convert categorical variables like `Credit_Mix`, `Payment_of_Min_Amount`, and `Payment_Behaviour` into numerical form for the ANN model.
- Scaling:** Normalize or scale features such as `Monthly_Inhand_Salary`, `Outstanding_Debt`, and `Total_EMI_per_month` to prepare them for the ANN model.

User-Defined-Functions

In [12]: # ===== User-Defined-Function =====

```

===== Get count and percentage of values for each column =====
def get_value_count(df, column_name):
    """
    This function calculates and returns a DataFrame with the value counts and
    their corresponding percentages for a specified column in the DataFrame.
    """

    vc = df[column_name].value_counts()

```

```

vc_norm = df[column_name].value_counts(normalize=True)

vc = vc.rename_axis(column_name).reset_index(name='counts')
vc_norm = vc_norm.rename_axis(column_name).reset_index(name='percent')
vc_norm['percent'] = (vc_norm['percent'] * 100).map('{:.2f}%'.format)

df_result = pd.concat([vc[column_name], vc['counts'], vc_norm['percent']], axis=1)

return df_result


# ===== User-Defined-Function for Missing Values =====
def missing_values(df):
    """This function calculates the missing values count and their percentage in a DataFrame."""

    missing_count = df.isnull().sum()
    value_count = df.isnull().count()
    missing_percentage = round(missing_count / value_count * 100, 2)

    # Format the percentage as '0.00%' with % symbol
    missing_percentage_formatted = missing_percentage.map("{:.2f}%".format)
    # Create a DataFrame to store the results
    missing_df = pd.DataFrame({'count': missing_count, "percentage": missing_percentage_formatted})

    return missing_df


# ===== Compare Missing Values (Train-Test) =====
def compare_missing_values(train, test):
    """
    Compares missing values between train and test datasets, returning counts, percentages, and data types.
    """

    def missing_data(df, label):
        missing_count = df.isna().sum()[df.isna().sum() > 0]
        total_count = len(df)
        missing_percentage = (missing_count / total_count * 100).map("{:.2f}%".format)
        return pd.DataFrame({
            f'{label} Missing Values': missing_count,
            f'{label} Missing Percentage': missing_percentage,
            f'{label} dtypes': df.dtypes[missing_count.index]
        })

    # Get missing data for train and test
    train_missing_df = missing_data(train, 'Train')
    test_missing_df = missing_data(test, 'Test')

    # Concatenate the missing values side by side
    return pd.concat([train_missing_df, test_missing_df], axis=1)


# ===== Plotting Missing Values =====
def na_ratio_plot(df):
    """Plots the ratio of missing values for each feature and prints the count of missing values."""

    sns.displot(df.isna().melt(value_name='Missing_data', var_name='Features')\
                ,y='Features',hue='Missing_data',multiple='fill',aspect=9/8)

    print(df.isna().sum()[df.isna().sum()>0])


#===== Detecting Non-Numerical Characters =====
import re

def find_non_numeric_values(df, column_name):
    """
    Finds unique non-numeric values in a specified column of the DataFrame.
    """

    pattern = r'\D+' # Pattern to match non-numeric characters
    # Find and flatten non-numeric values, then ensure uniqueness with set
    return set(re.findall(pattern, ' '.join(df[column_name].astype(str))))


#=====

```

Handling Missing Values and Cleaning Data

```
In [13]: # Comparing missing values in Train and Test data
compare_missing_values(train, test)
```

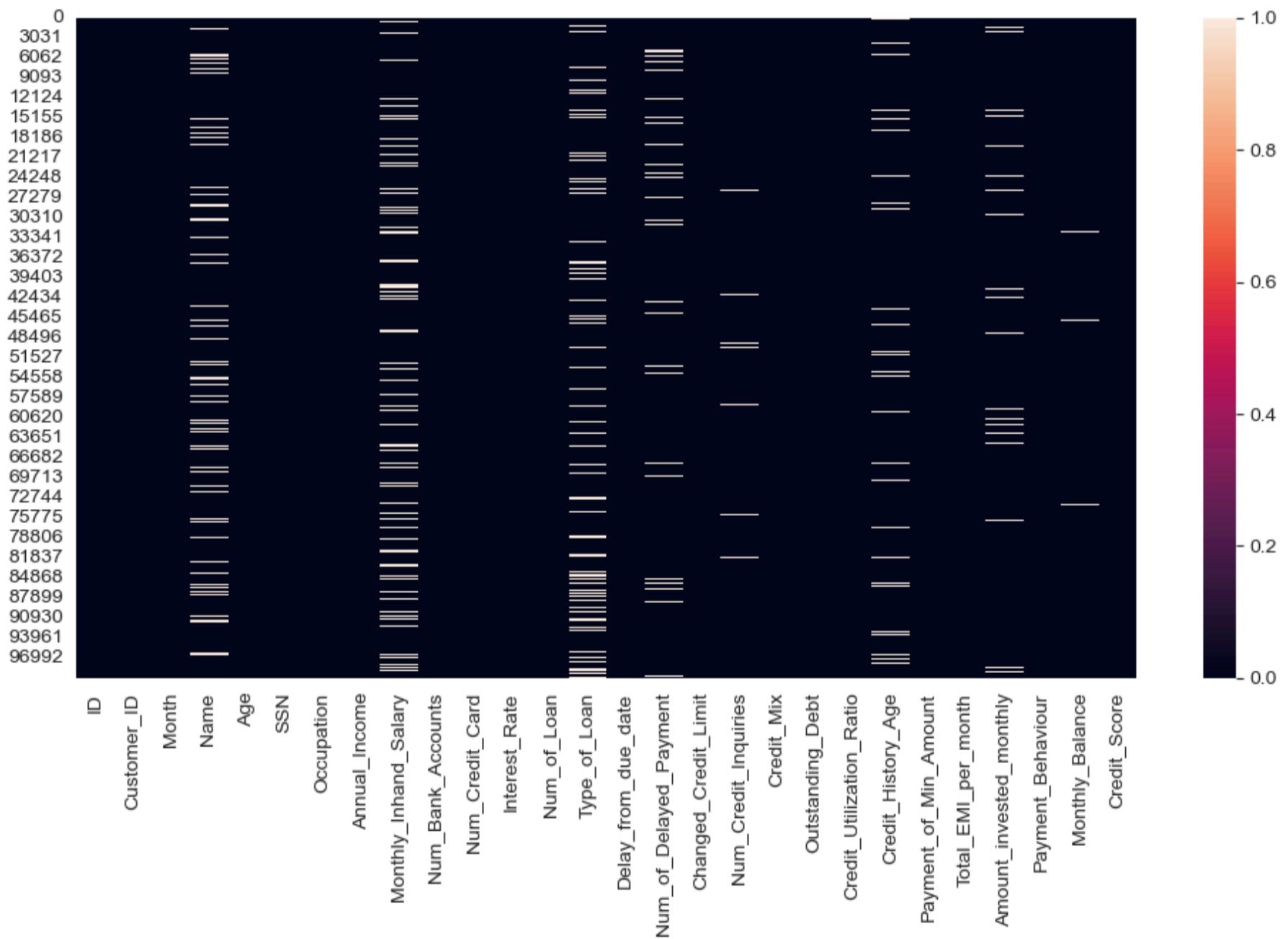
Out[13]:

	Train Missing Values	Train Missing Percentage	Train dtypes	Test Missing Values	Test Missing Percentage	Test dtypes
Name	9985	9.98%	object	5015	10.03%	object
Monthly_Inhand_Salary	15002	15.00%	float64	7498	15.00%	float64
Type_of_Loan	11408	11.41%	object	5704	11.41%	object
Num_of_Delayed_Payment	7002	7.00%	object	3498	7.00%	object
Num_Credit_Inquiries	1965	1.97%	float64	1035	2.07%	float64
Credit_History_Age	9030	9.03%	object	4470	8.94%	object
Amount_invested_monthly	4479	4.48%	object	2271	4.54%	object
Monthly_Balance	1200	1.20%	object	562	1.12%	object

```
In [14]: # TRAIN DATASET
```

```
sns.heatmap(train.isnull())
```

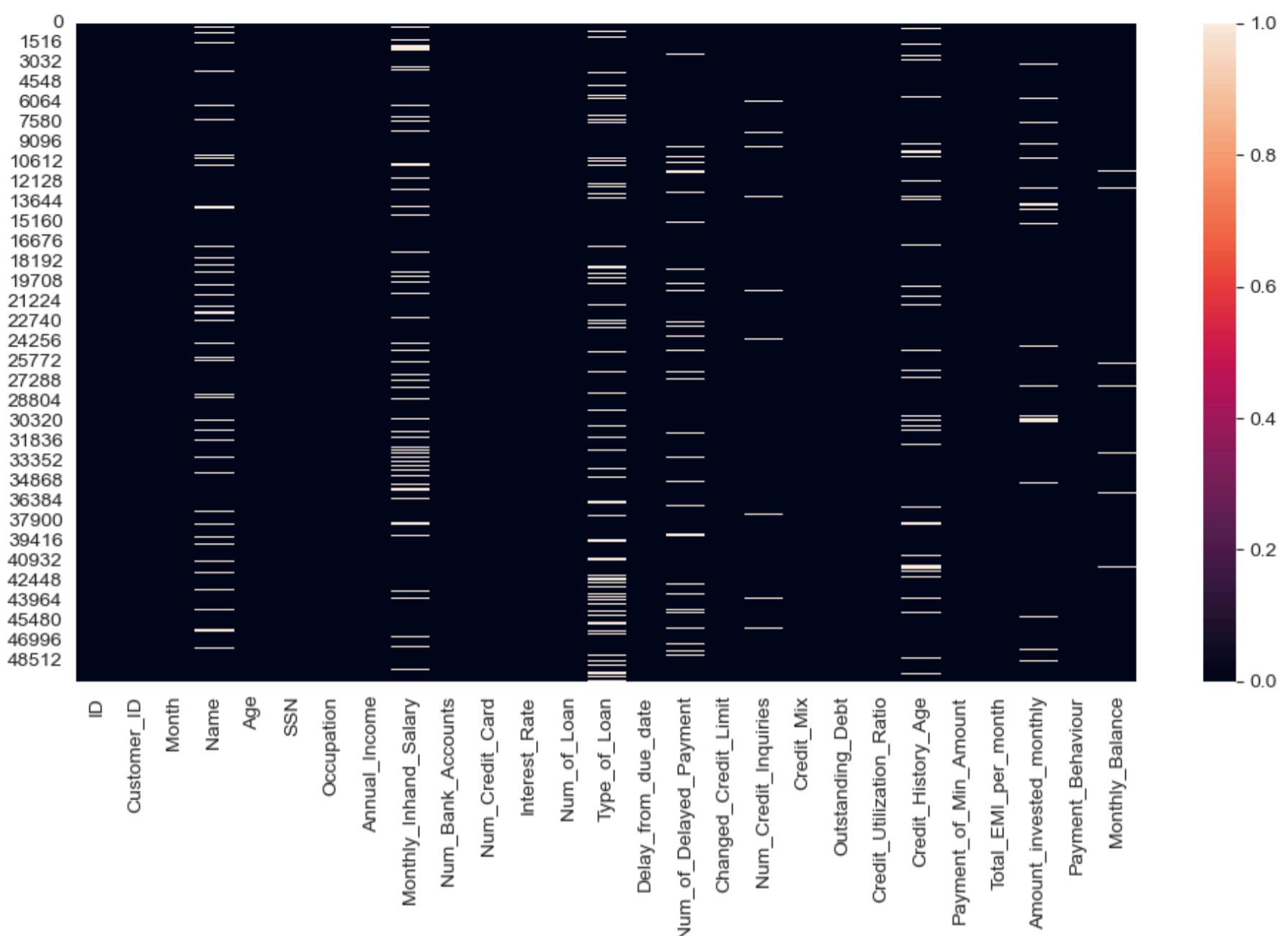
Out[14]: <Axes: >



```
In [15]: # TEST DATASET
```

```
sns.heatmap(test.isnull())
```

Out[15]: <Axes: >



'Name'

The **Name** column represents the full name of the customer. While it provides identification, it is typically not useful for analysis or modeling purposes and may be removed or anonymized to protect privacy during data preprocessing.

- Identification columns (ID, Customer_ID, Name, SSN) are not directly useful in classification tasks and can increase the complexity of the model rather than improving its performance.
- Therefore, these columns will be dropped at the end of the cleaning data step.

```
In [16]: # Name column unique values and percentage
get_value_count(df, 'Name')
```

	Name	counts	percent
0	Stevex	66	0.05%
1	Langep	65	0.05%
2	Jessicad	59	0.04%
3	Vaughanl	58	0.04%
4	Raymondr	58	0.04%
...
10134	Carrick Mollenkampe	7	0.01%
10135	Giuseppeh	7	0.01%
10136	Drivera	7	0.01%
10137	Emi Emotor	7	0.01%
10138	Timothyl	6	0.00%

10139 rows × 3 columns

```
In [17]: train[train['Name'].isnull()]
```

Out[17]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_S
7	0x1609	CUS_0xd40	August	NaN	23	#F%\$D@*&8	Scientist	19114.12	182
17	0x161b	CUS_0x2dbc	February	NaN	34	486-85-3974	Engineer	143162.64	1218
22	0x1620	CUS_0x2dbc	July	NaN	34	486-85-3974	Engineer	143162.64	1218
64	0x1662	CUS_0x4157	January	NaN	23	070-19-1622	Doctor	114838.41	984
80	0x167a	CUS_0xa66b	January	NaN	40	221-30-8554	Teacher	33751.27	294
...
99964	0x25fba	CUS_0x372c	May	NaN	18	340-85-7301	Lawyer	42903.79	346
99965	0x25fbb	CUS_0x372c	June	NaN	19	340-85-7301	Lawyer	42903.79	346
99969	0x25fc3	CUS_0xf16	February	NaN	45	868-70-2218	Media_Manager	16680.35	152
99973	0x25fc7	CUS_0xf16	June	NaN	45	868-70-2218	Media_Manager	16680.35	152
99986	0x25fdc	CUS_0x8600	March	NaN	28	031-35-0942	Architect	20002.88	192

9985 rows × 28 columns

'Annual_Income'

The **Annual_Income** column represents the total income a customer earns in a year before any deductions, such as taxes or other withholdings. This value is a crucial indicator of the customer's overall financial stability and their ability to repay loans or manage other financial obligations.

- The **Annual_Income** column currently has the **object** data type, but it should be converted to **float** since it contains numerical values.
- However, some values contain underscores (_) at the beginning or end.
- These characters need to be removed before converting the data type to ensure proper numerical processing.

In [18]:

```
# column unique values and percentage
get_value_count(train, 'Annual_Income')
```

Out[18]:

	Annual_Income	counts	percent
0	36585.12	16	0.02%
1	20867.67	16	0.02%
2	17273.83	16	0.02%
3	9141.63	15	0.01%
4	33029.66	15	0.01%
...
18935	20269.93_	1	0.00%
18936	15157.25_	1	0.00%
18937	44955.64_	1	0.00%
18938	76650.12_	1	0.00%
18939	4262933.0	1	0.00%

18940 rows × 3 columns

In [19]: # Check missing values and dtype

```
print('Remaining missing values in Train:', train['Annual_Income'].isna().sum())
print('Remaining missing values in Test:', test['Annual_Income'].isna().sum())
print('dtype: ', train['Monthly_Inhand_Salary'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Annual_Income')
```

Remaining missing values in Train: 0

Remaining missing values in Test: 0

dtype: float64

Out[19]: {' ', '.', '_', '_ '}

In [20]: # TRAIN

```
# Remove non-numeric characters from 'Annual_Income' column in train dataset
train['Annual_Income'] = train['Annual_Income'].apply(lambda x: re.sub(r'^0-9.', '', str(x)))

# Convert 'Annual_Income' to float in train dataset
train['Annual_Income'] = train['Annual_Income'].astype(float)
```

In [21]: # TEST

```
# Remove non-numeric characters from 'Annual_Income' column in test dataset
test['Annual_Income'] = test['Annual_Income'].apply(lambda x: re.sub(r'^0-9.', '', str(x)))

# Convert 'Annual_Income' to float in test dataset
test['Annual_Income'] = test['Annual_Income'].astype(float)
```

In [22]: train['Annual_Income'].describe()

```
Out[22]: count    1000000.000
mean      176415.701
std       1429618.051
min       7005.930
25%      19457.500
50%      37578.610
75%      72790.920
max     24198062.000
Name: Annual_Income, dtype: float64
```

In [23]: # Plot Average Annual Income by Credit Score

```
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Credit_Score', y='Annual_Income', data=train, ci=None, palette='Greens_r')

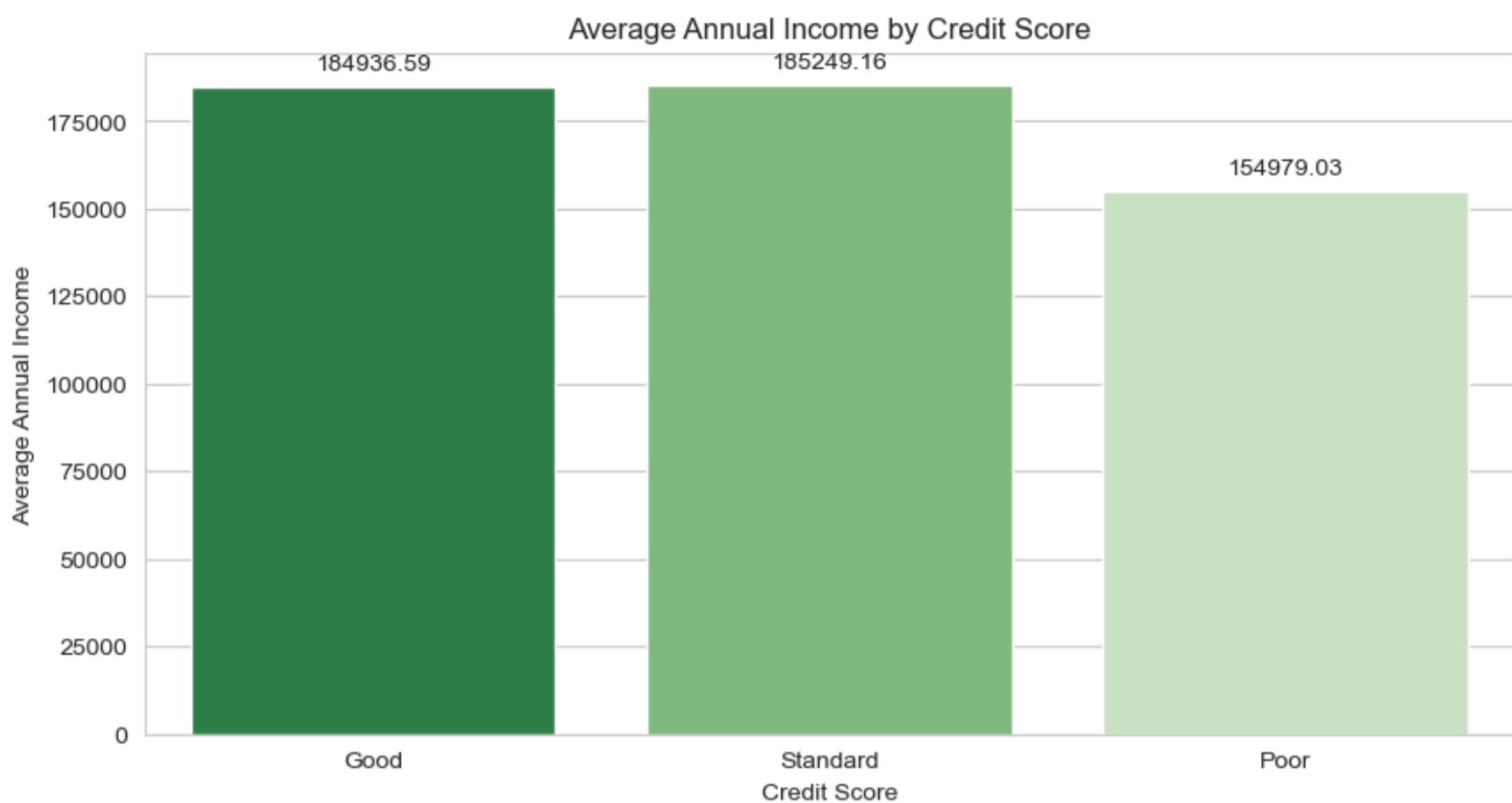
# Add values on top of the bars
for p in ax.patches: ax.annotate(format(p.get_height(), '.2f'),
                                    (p.get_x() + p.get_width() / 2., p.get_height()),
                                    ha='center', va='center',
                                    xytext=(0, 9), textcoords='offset points')
```

```

plt.title('Average Annual Income by Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Average Annual Income')

plt.show()

```



- This plot shows that individuals with **Standard** credit scores have the highest average annual income (185,249.16), followed closely by those with **Good** credit scores (184,936.59).
- Individuals with **Poor** credit scores have the lowest average annual income (154,979.03).

This indicates that higher incomes are generally associated with better credit scores.

'Monthly_Inhand_Salary'

The **Monthly_Inhand_Salary** column represents the amount of salary a customer receives after deductions, such as taxes and other withholdings. This is the net salary the customer takes home on a monthly basis, and it can be a key indicator of the customer's disposable income and financial capacity.

- Since there is no information in the dataset regarding deductions or taxes, we will use the approach of filling missing values in the **Monthly_Inhand_Salary** column by dividing the **Annual_Income** by 12.
- This method ensures that all records have a complete monthly salary value, allowing the model to work with consistent and filled data for better predictions.

```
In [24]: # column unique values and percentage
get_value_count(train, 'Monthly_Inhand_Salary')
```

Out[24]:

	Monthly_Inhand_Salary	counts	percent
0	6769.130	15	0.02%
1	6358.957	15	0.02%
2	2295.058	15	0.02%
3	6082.188	15	0.02%
4	3080.555	14	0.02%
...
13230	1087.546	1	0.00%
13231	3189.212	1	0.00%
13232	5640.118	1	0.00%
13233	7727.560	1	0.00%
13234	2443.654	1	0.00%

13235 rows × 3 columns

In [25]: # Check missing values and dtype

```
print('Remaining missing values in Train:', train['Monthly_Inhand_Salary'].isna().sum())
print('Remaining missing values in Test:', test['Monthly_Inhand_Salary'].isna().sum())
print('dtype: ', train['Monthly_Inhand_Salary'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Monthly_Inhand_Salary')
```

Remaining missing values in Train: 15002

Remaining missing values in Test: 7498

dtype: float64

Out[25]: {',',
' nan ',
' nan nan ',
' nan nan nan ',
' nan nan nan nan ',
' nan nan nan nan nan ',
.}'}

In [26]: # numeric columns

```
numeric_columns = train.select_dtypes(include=[np.number])

# Calculate and sort the absolute correlation matrix
df_corr = numeric_columns.corr().abs().unstack().sort_values(kind="quicksort", ascending=False).reset_index

# Filter the columns that are related to the 'Monthly_Balance' column
df_corr[df_corr['level_0'] == 'Monthly_Inhand_Salary'].head()
```

Out[26]:

	level_0	level_1	0
1	Monthly_Inhand_Salary	Monthly_Inhand_Salary	1.000
10	Monthly_Inhand_Salary	Delay_from_due_date	0.250
12	Monthly_Inhand_Salary	Credit_Utilization_Ratio	0.173
16	Monthly_Inhand_Salary	Annual_Income	0.031
22	Monthly_Inhand_Salary	Num_Bank_Accounts	0.011

In [27]: # TRAIN DATA

Filling missing values in Monthly_Inhand_Salary using Annual_Income for train dataset
train['Monthly_Inhand_Salary'].fillna(train['Annual_Income'] / 12, inplace=True)

TEST DATA

Filling missing values in Monthly_Inhand_Salary using Annual_Income for test dataset
test['Monthly_Inhand_Salary'].fillna(test['Annual_Income'] / 12, inplace=True)

Check missing values

```
print('Remaining missing values in Train:', train['Monthly_Inhand_Salary'].isna().sum())
print('Remaining missing values in Test:', test['Monthly_Inhand_Salary'].isna().sum())
```

Remaining missing values in Train: 0
 Remaining missing values in Test: 0

In [28]: `train['Monthly_Inhand_Salary'].describe()`

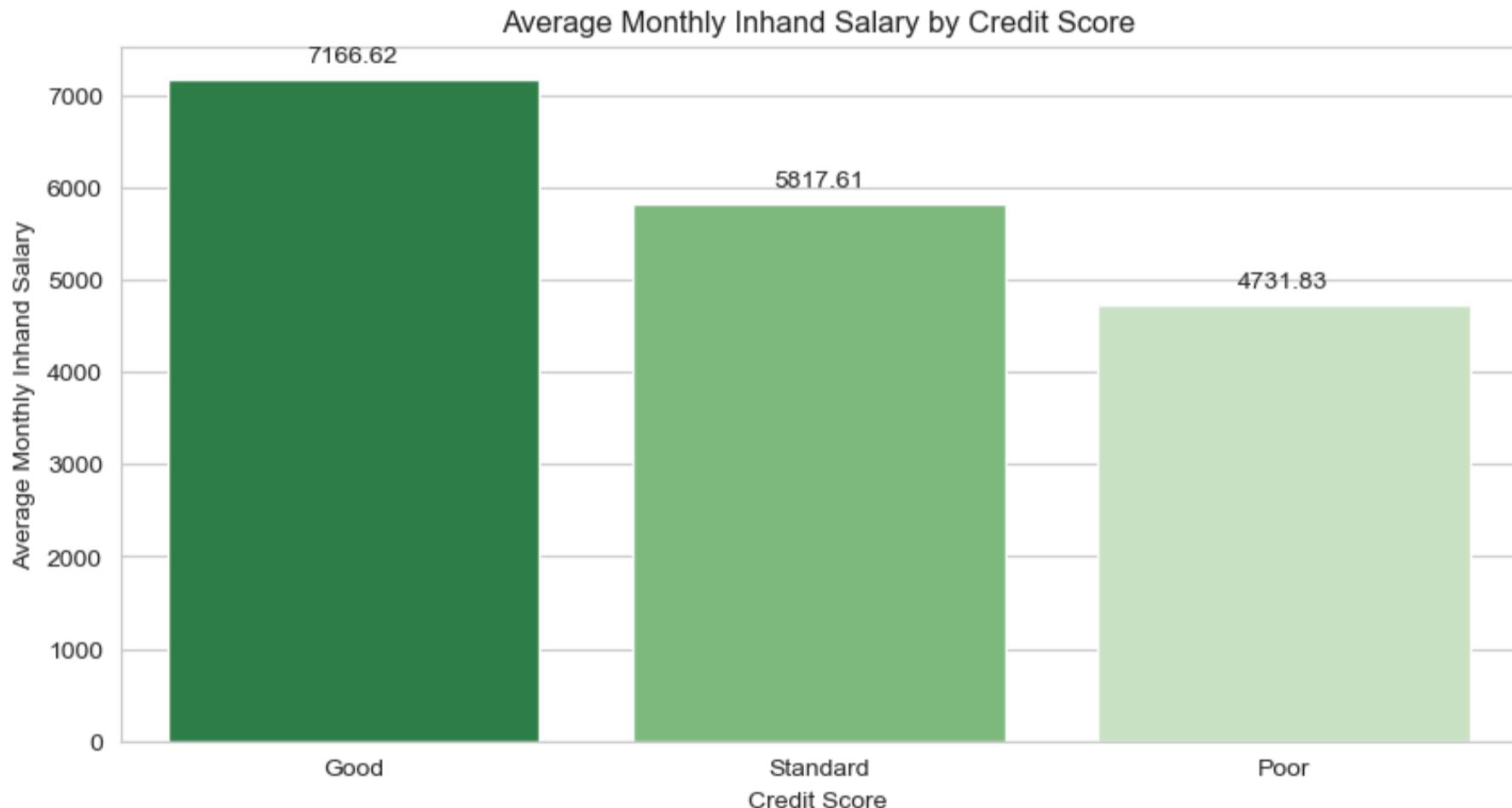
```
Out[28]: count    100000.000
mean      5743.259
std       45814.695
min       303.645
25%      1625.793
50%      3101.372
75%      5971.780
max     1990379.583
Name: Monthly_Inhand_Salary, dtype: float64
```

```
In [29]: # Plot Average Monthly Inhand Salary by Credit Score
plt.figure(figsize=(10, 5))
ax=sns.barplot(x='Credit_Score', y='Monthly_Inhand_Salary', data=train, ci=None, palette='Greens_r')

for p in ax.patches: ax.annotate(format(p.get_height(), '.2f'),
                                 (p.get_x() + p.get_width() / 2., p.get_height()),
                                 ha='center', va='center',
                                 xytext=(0, 9), textcoords='offset points')

plt.title('Average Monthly Inhand Salary by Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Average Monthly Inhand Salary')

plt.show()
```



- This plot shows that individuals with a **Good** credit score have the highest average monthly in-hand salary (7166.62), followed by those with a **Standard** credit score (5817.61).
- Individuals with a **Poor** credit score have the lowest average monthly salary (4731.83).

This indicates a positive correlation between higher in-hand salary and better credit scores.

'Type_of_Loan'

The **Type_of_Loan** column represents the specific type(s) of loan a customer has taken. It can include various loan categories, such as **Personal Loan**, **Auto Loan**, **Home Equity Loan**, etc. This information helps to categorize the different loan products a customer may have and can be used to understand their financial behavior or preferences.

- The **Type_of_Loan** column was transformed to separate multiple loan types into individual binary columns (0 and 1).

- Each new column indicates whether a specific loan type is present in that row.
- This helps the model better understand the different loan types.
- **True/False** values were converted to **0/1**, and the original **Type_of_Loan** column was deleted.

In [30]: `# Check column unique values and percentage
get_value_count(train, 'Type_of_Loan')`

Out[30]:

	Type_of_Loan	counts	percent
0	Not Specified	1408	1.59%
1	Credit-Builder Loan	1280	1.44%
2	Personal Loan	1272	1.44%
3	Debt Consolidation Loan	1264	1.43%
4	Student Loan	1240	1.40%
...
6255	Not Specified, Mortgage Loan, Auto Loan, and P...	8	0.01%
6256	Payday Loan, Mortgage Loan, Debt Consolidation...	8	0.01%
6257	Debt Consolidation Loan, Auto Loan, Personal L...	8	0.01%
6258	Student Loan, Auto Loan, Student Loan, Credit-...	8	0.01%
6259	Personal Loan, Auto Loan, Mortgage Loan, Stude...	8	0.01%

6260 rows × 3 columns

In [31]: `# Check missing values and dtype`

```
print('Remaining missing values in Train:', train['Type_of_Loan'].isna().sum())
print('Remaining missing values in Test:', test['Type_of_Loan'].isna().sum())
print('dtype: ', train['Type_of_Loan'].dtypes)
```

```
Remaining missing values in Train: 11408
Remaining missing values in Test: 5704
dtype: object
```

In [32]: `# Unique values`

```
# Split all the Unique Loan Types to see
unique_loan = df['Type_of_Loan'].dropna().str.split(',').explode().str.strip().unique()
# clean
unique_loan = [loan.replace('and ', '') for loan in unique_loan]
unique_loan
```

Out[32]:

```
['Auto Loan',
 'Credit-Builder Loan',
 'Personal Loan',
 'Home Equity Loan',
 'Not Specified',
 'Not Specified',
 'Mortgage Loan',
 'Student Loan',
 'Debt Consolidation Loan',
 'Auto Loan',
 'Payday Loan',
 'Payday Loan',
 'Student Loan',
 'Personal Loan',
 'Home Equity Loan',
 'Mortgage Loan',
 'Debt Consolidation Loan',
 'Credit-Builder Loan']
```

In [33]: `# TRAIN DATA`

```
# For each of the most common Loan types (excluding the first one) in the train dataset
for i in train['Type_of_Loan'].value_counts().head(9).index[1:]:

    # Create a new column for each loan type in the train dataset
    # The new column will be 1 if the loan type is present in 'Type_of_Loan', 0 otherwise
    train[i] = train['Type_of_Loan'].str.contains(i, na=False).astype(int)
```

```
# Delete the original 'Type_of_Loan' column after creating binary columns in the train dataset
del train['Type_of_Loan']

# Display the first few rows of the modified train dataframe
train.head(3)
```

Out[33]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_B
0	0x1602	CUS_0xd40	January	Aaron Maashoh	23	821-00-0265	Scientist	19114.120	1824.843	
1	0x1603	CUS_0xd40	February	Aaron Maashoh	23	821-00-0265	Scientist	19114.120	1592.843	
2	0x1604	CUS_0xd40	March	Aaron Maashoh	-500	821-00-0265	Scientist	19114.120	1592.843	

3 rows × 35 columns

In [34]:

```
# TEST DATA
# For each of the most common Loan types (excluding the first one) in the test dataset
for i in test['Type_of_Loan'].value_counts().head(9).index[1:]:

    # Create a new column for each loan type in the test dataset
    # The new column will be 1 if the loan type is present in 'Type_of_Loan', 0 otherwise
    test[i] = test['Type_of_Loan'].str.contains(i, na=False).astype(int)

# Delete the original 'Type_of_Loan' column after creating binary columns in the test dataset
del test['Type_of_Loan']

# Display the first few rows of the modified test dataframe
test.head(3)
```

Out[34]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_
0	0x160a	CUS_0xd40	September	Aaron Maashoh	23	821-00-0265	Scientist	19114.120	1824.843	
1	0x160b	CUS_0xd40	October	Aaron Maashoh	24	821-00-0265	Scientist	19114.120	1824.843	
2	0x160c	CUS_0xd40	November	Aaron Maashoh	24	821-00-0265	Scientist	19114.120	1824.843	

3 rows × 34 columns

The following columns have been created from the **Type_of_Loan** column:

- Credit-Builder Loan ,
- Personal Loan ,
- Debt Consolidation Loan ,
- Student Loan ,
- Payday Loan ,
- Mortgage Loan ,
- Auto Loan ,
- Home Equity Loan

In [35]:

```
# List of loan-related columns
loan_columns = ['Credit-Builder Loan', 'Personal Loan', 'Debt Consolidation Loan', 'Student Loan',
```

```
'Payday Loan', 'Mortgage Loan', 'Auto Loan', 'Home Equity Loan']
```

```
# Check for missing values in each of the Loan-related columns
train[loan_columns].isna().sum()
```

```
Out[35]: Credit-Builder Loan      0
Personal Loan                  0
Debt Consolidation Loan       0
Student Loan                   0
Payday Loan                    0
Mortgage Loan                  0
Auto Loan                      0
Home Equity Loan                0
dtype: int64
```

```
In [36]: # Distribution of Loan Types by Credit Score
```

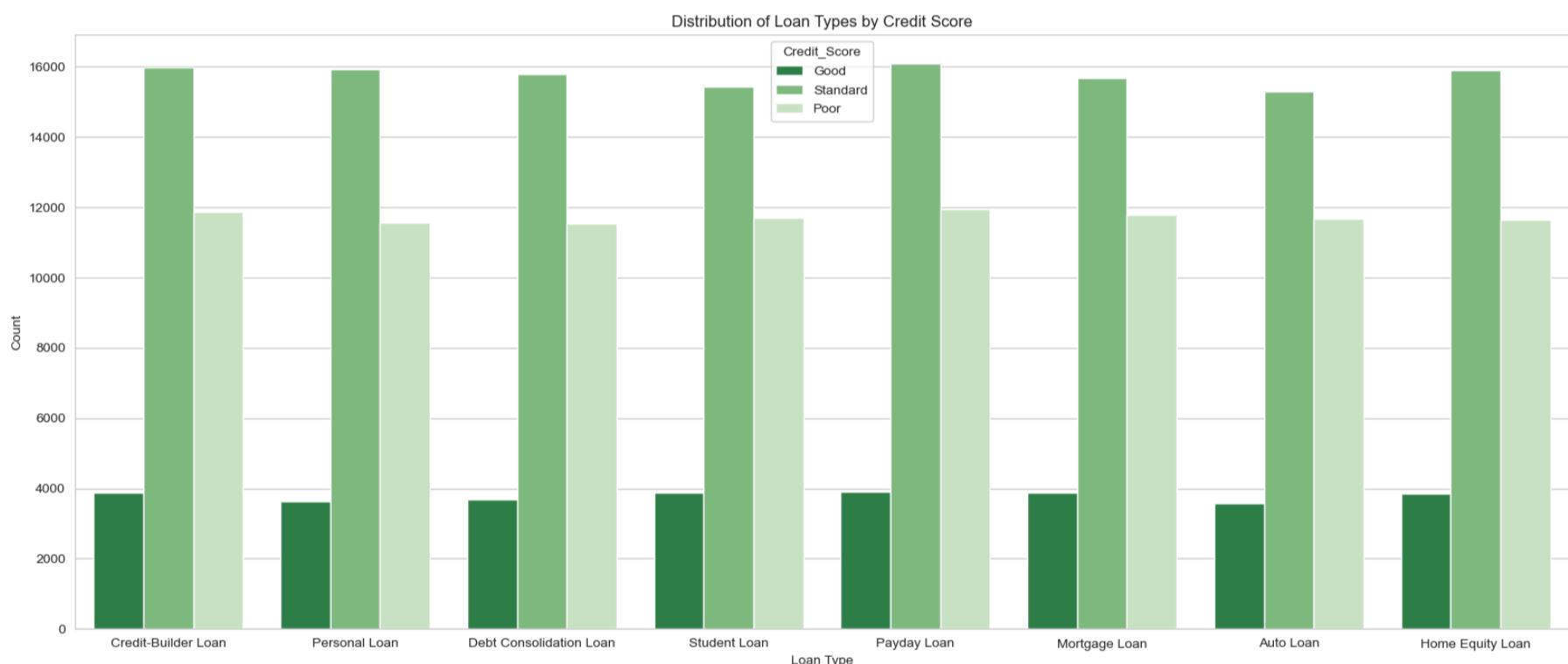
```
loan_columns = ['Credit-Builder Loan', 'Personal Loan', 'Debt Consolidation Loan', 'Student Loan',
                 'Payday Loan', 'Mortgage Loan', 'Auto Loan', 'Home Equity Loan']

df_melted = train.melt(id_vars='Credit_Score', value_vars=loan_columns,
                       var_name='Loan Type', value_name='Has Loan')

plt.figure(figsize=(20,8))
sns.countplot(x='Loan Type', hue='Credit_Score', data=df_melted[df_melted['Has Loan'] == 1], palette='Greens')

plt.title('Distribution of Loan Types by Credit Score')
plt.xlabel('Loan Type')
plt.ylabel('Count')

plt.show()
```



- The plot shows that individuals with **Standard** and **Poor** credit scores tend to have more **Credit-Builder Loans**, **Personal Loans**, and **Debt Consolidation Loans**.
- Good** credit holders generally have fewer loans, particularly **Auto Loans** and **Home Equity Loans**.

This suggests that certain loan types (`Payday Loan` and `Credit-Builder Loan`) are more common among people with lower credit scores, while those with good credit have fewer loans overall.

'Num_of_Delayed_Payment'

The **Num_of_Delayed_Payment** column represents the number of times a customer has delayed their payment beyond the due date.

This typically refers to the total count of late payments a customer has made on loans, credit cards, or other financial obligations.

It can be an important feature for credit scoring models, as a higher number of delayed payments usually indicates a higher risk of default or financial instability.

Anomalies:

- A value of **-1** likely represents missing or incorrect data. Since the number of delayed payments cannot logically be negative, this value might have been used as a placeholder for unavailable or invalid data.
- **NaN** values indicate missing data, which need to be addressed during data cleaning, either by imputing appropriate values (e.g., the median or mode) or leaving them as **NaN** if suitable.
- Other non-numeric anomalies, such as non-digit characters, should also be considered errors and cleaned appropriately.
- Additionally, the column's data type is currently **object**, which is incorrect for a numerical feature. After cleaning, the data type should be converted to an appropriate numerical format, such as **int** or **float**, to ensure proper analysis and calculations.

In [37]: `# Check the column' unique values and percentage
get_value_count(train, 'Num_of_Delayed_Payment')`

Out[37]:

	Num_of_Delayed_Payment	counts	percent
0	19	5327	5.73%
1	17	5261	5.66%
2	16	5173	5.56%
3	10	5153	5.54%
4	18	5083	5.47%
...
744	848_-	1	0.00%
745	4134	1	0.00%
746	1530	1	0.00%
747	1502	1	0.00%
748	2047	1	0.00%

749 rows × 3 columns

In [38]: `# Check missing values and dtype`

```
print('Remaining missing values in Train:', train['Num_of_Delayed_Payment'].isna().sum())
print('Remaining missing values in Test:', test['Num_of_Delayed_Payment'].isna().sum())
print('dtype: ', train['Num_of_Delayed_Payment'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Num_of_Delayed_Payment')
```

Remaining missing values in Train: 7002
 Remaining missing values in Test: 3498
 dtype: object

Out[38]:

```
{' ',  
 ' -',  
 ' nan ',  
 ' nan -',  
 ' nan nan ',  
 ' nan nan -',  
 ' nan nan nan ',  
 ' nan nan nan -',  
 ' nan nan nan nan ',  
 ' nan nan nan nan nan ',  
 '_ ',  
 '_ -',  
 '_ nan ',  
 '_ nan -',  
 '_ nan nan '}
```

Num_of_Delayed_Payment column:

contains anomalies like negative values **-1**, Non-numeric characters **_** or **-** and **NaN** values that need to be addressed, possibly through imputation.

Additionally, the column's data type is currently **object**, which should be converted to a numerical format (**int** or **float**).

Clean Anomalies in the Column 'Num_of_Delayed_Payment':

```
In [39]: # TRAIN DATA

# 1. Remove non-numeric characters from the column in train dataset
train['Num_of_Delayed_Payment'] = train['Num_of_Delayed_Payment'].apply(lambda x: re.sub(r'^[^\d.]+$', '', str(x)))
# Replace empty strings with NaN
train['Num_of_Delayed_Payment'].replace('', np.nan, inplace=True)

# 2. Convert dtype to float in train dataset
train['Num_of_Delayed_Payment'] = train['Num_of_Delayed_Payment'].astype(float)

# 3. Convert negatives like '-1' to positives
train['Num_of_Delayed_Payment'] = train['Num_of_Delayed_Payment'].abs()
```

```
In [40]: # TEST DATA

# 1. Remove non-numeric characters from 'Num_of_Delayed_Payment' column in the test dataset
test['Num_of_Delayed_Payment'] = test['Num_of_Delayed_Payment'].apply(lambda x: re.sub(r'^[^\d.]+$', '', str(x)))
# Replace empty strings with NaN
test['Num_of_Delayed_Payment'].replace('', np.nan, inplace=True)

# 2. Convert 'Num_of_Delayed_Payment' to float in the test dataset
test['Num_of_Delayed_Payment'] = test['Num_of_Delayed_Payment'].astype(float)

# 3. Convert negatives like '-1' to positives
test['Num_of_Delayed_Payment'] = test['Num_of_Delayed_Payment'].abs()
```

Filling the Missing Values in 'Num_of_Delayed_Payment':

```
In [41]: # ===== User-Defined-Fonction =====

from sklearn.impute import KNNImputer

def knn_impute_column(df, column, n_neighbors=5):
    """
    Impute missing values in the specified column using KNN.
    Args: df (DataFrame), column (str), n_neighbors (int): Number of neighbors (Default is 5).
    Returns: DataFrame: DataFrame with imputed column.
    """
    # Apply KNN imputation to the specified column
    imputer = KNNImputer(n_neighbors=n_neighbors)
    df[[column]] = imputer.fit_transform(df[[column]])

    return df
# =====
```

```
In [42]: # Apply KNN imputation to 'Num_of_Delayed_Payment' column in both train and test datasets
knn_impute_column(train, 'Num_of_Delayed_Payment') #Train
knn_impute_column(test, 'Num_of_Delayed_Payment') #Test

# Check Missing values
print('Remaining missing values in Train:', train['Num_of_Delayed_Payment'].isna().sum())
print('Remaining missing values in Test:', test['Num_of_Delayed_Payment'].isna().sum())
```

Remaining missing values in Train: 0
 Remaining missing values in Test: 0

Note:

KNN Imputer is been used to fill missing values in the specific column because it leverages the similarity between neighboring data points. Instead of filling missing values with a fixed statistic (like mean or median), KNN Imputer looks at the closest **n** neighbors (based on other available features) and imputes the missing values based on the average of those neighbors. This method is useful when the data exhibits patterns or relationships that simple imputation methods may overlook, providing a more informed estimate.

```
In [43]: train['Num_of_Delayed_Payment'].describe()
```

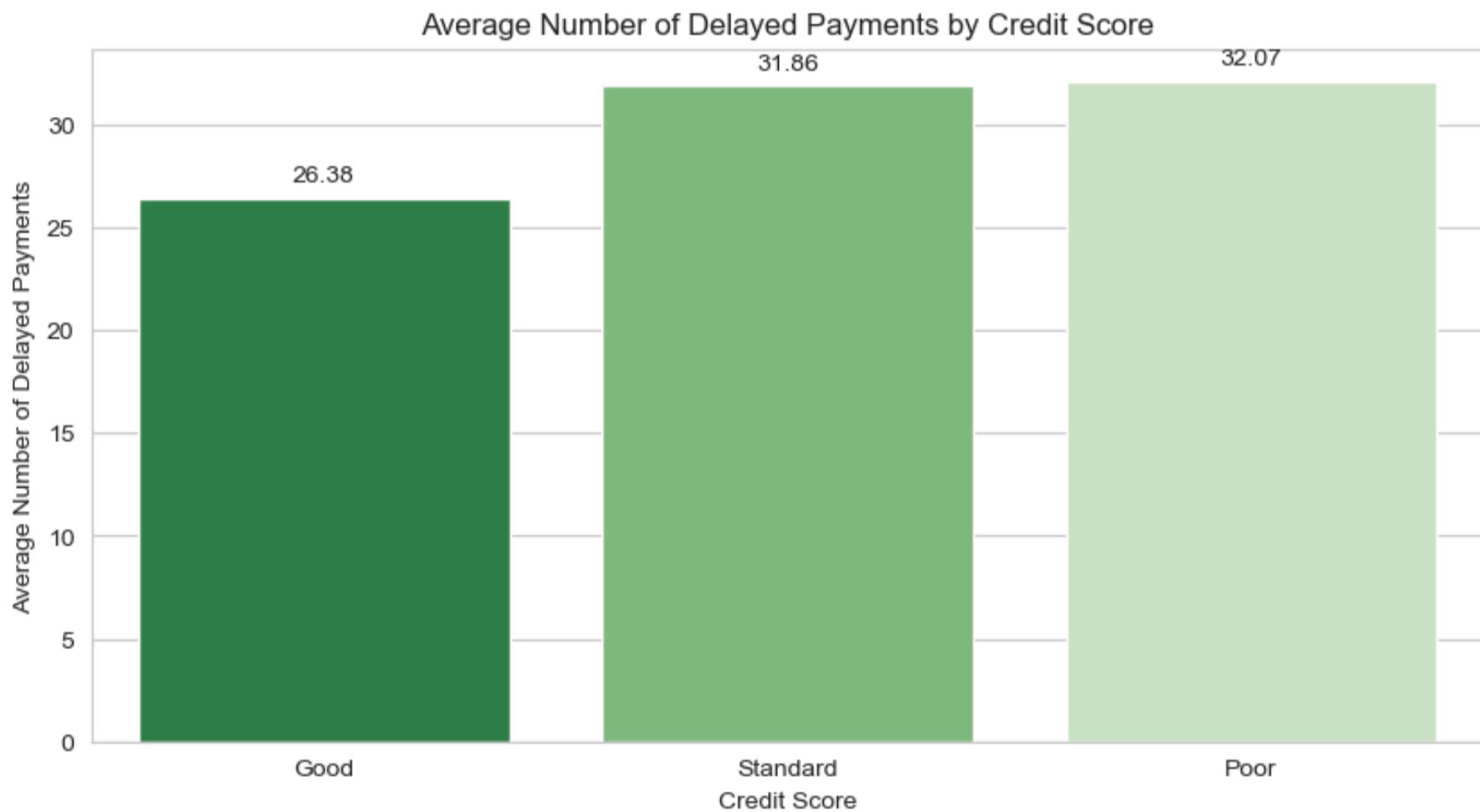
```
Out[43]: count    1000000.000
mean        30.946
std         217.972
min         0.000
25%         9.000
50%        15.000
75%        19.000
max        4397.000
Name: Num_of_Delayed_Payment, dtype: float64
```

```
In [44]: # Average Num_of_Delayed_Payment by Credit Score
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Credit_Score', y='Num_of_Delayed_Payment', data=train, ci=None, palette='Greens_r')

# Add values on top of the bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 9), textcoords='offset points')

# Add Labels and title
plt.title('Average Number of Delayed Payments by Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Average Number of Delayed Payments')

plt.show()
```



- Customers with a **Poor** credit score have the highest average number of delayed payments (32.07), followed closely by those with a **Standard** credit score (31.86).

In contrast, customers with a **Good** credit score have a significantly lower average number of delayed payments (26.38). This indicates that individuals with poorer credit scores are more likely to delay their payments, contributing to their lower credit ratings.

'Num_Credit_Inquiries'

The **Num_Credit_Inquiries** column represents the number of credit inquiries made on a customer's credit report. Each inquiry typically occurs when a customer applies for credit, such as a loan or credit card, and lenders request to check the customer's credit history. A higher number of credit inquiries could indicate a higher risk of financial instability, as frequent inquiries may suggest the customer is seeking multiple sources of credit.

The column contained unusual values such as ' ', ' nan ', ' nan nan ', ' nan nan nan ', which were treated as missing data.

The missing values were then imputed using the **KNN imputation** method, which fills in the missing entries by considering the nearest neighbors, ensuring the data remains consistent for analysis.

```
In [45]: # Check the column' unique values and percentage
get_value_count(train, 'Num_Credit_Inquiries')
```

	Num_Credit_Inquiries	counts	percent
0	4.000	11271	11.50%
1	3.000	8890	9.07%
2	6.000	8111	8.27%
3	7.000	8058	8.22%
4	2.000	8028	8.19%
...
1218	1721.000	1	0.00%
1219	1750.000	1	0.00%
1220	2397.000	1	0.00%
1221	621.000	1	0.00%
1222	74.000	1	0.00%

1223 rows × 3 columns

```
In [46]: # Check missing values and dtype
print('Remaining missing values in Train:', train['Num_Credit_Inquiries'].isna().sum())
print('Remaining missing values in Test:', test['Num_Credit_Inquiries'].isna().sum())
print('dtype: ', train['Num_Credit_Inquiries'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Num_Credit_Inquiries')
```

Remaining missing values in Train: 1965
 Remaining missing values in Test: 1035
 dtype: float64

```
Out[46]: {' ': 'nan', 'nan': 'nan nan', 'nan nan': 'nan nan nan', 'nan nan nan': '.'}
```

```
In [47]: # Apply KNN imputation to 'Num_Credit_Inquiries' column in both train and test datasets
knn_impute_column(train, 'Num_Credit_Inquiries') # Train
knn_impute_column(test, 'Num_Credit_Inquiries') # Test

# Check Missing values
print('Remaining missing values in Train:', train['Num_Credit_Inquiries'].isna().sum())
print('Remaining missing values in Test:', test['Num_Credit_Inquiries'].isna().sum())
```

Remaining missing values in Train: 0
 Remaining missing values in Test: 0

```
In [48]: train['Num_Credit_Inquiries'].describe()
```

```
Out[48]: count    1000000.000
mean        27.754
std       191.270
min         0.000
25%        3.000
50%        6.000
75%        9.000
max      2597.000
Name: Num_Credit_Inquiries, dtype: float64
```

```
In [49]: # Bar plot showing the average number of credit inquiries by credit score category
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Credit_Score', y='Num_Credit_Inquiries', data=train, ci=None, palette='Greens_r')

# Add values on top of the bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
```

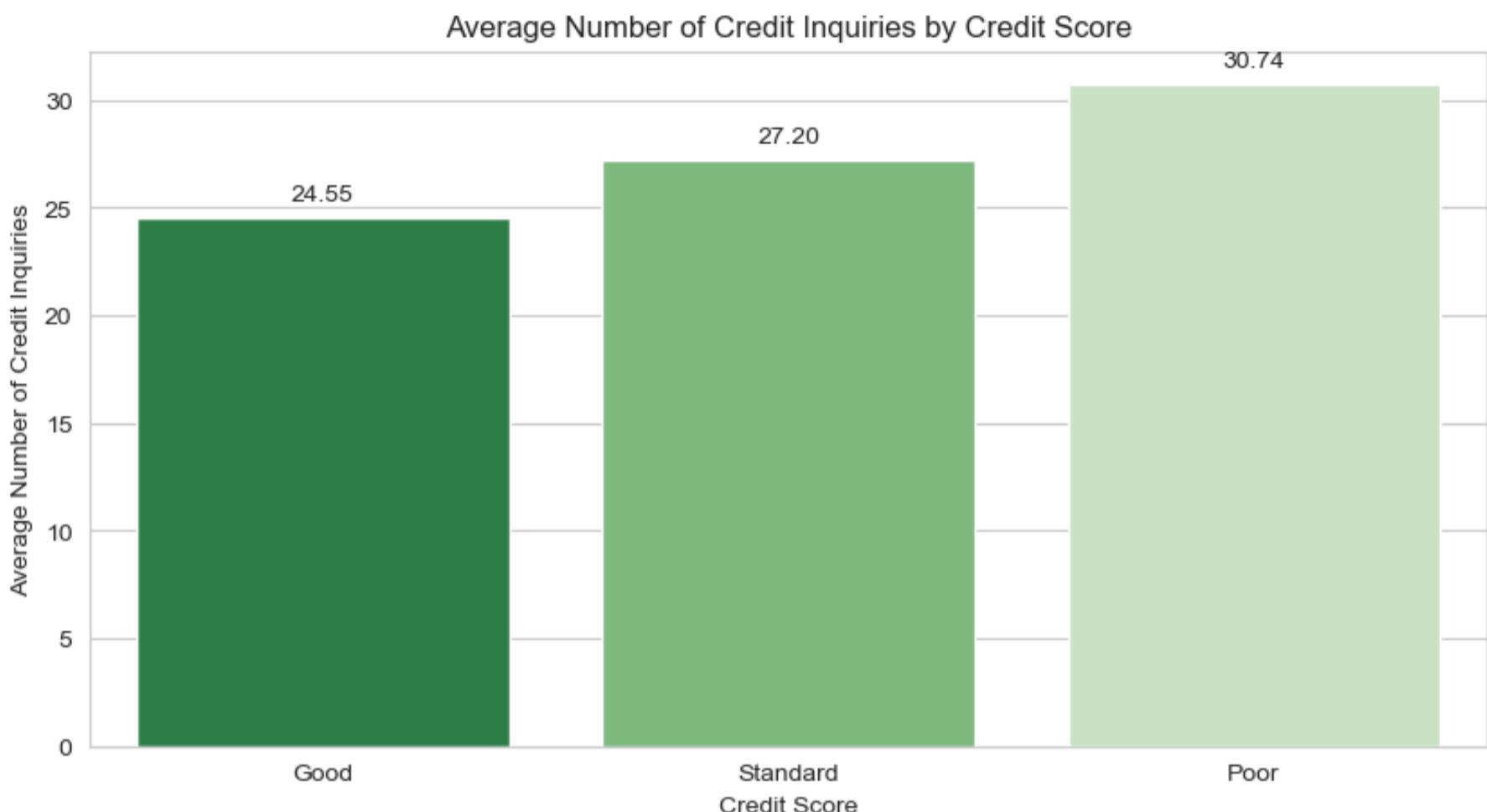
```

ha='center', va='center',
xytext=(0, 9), textcoords='offset points')

# Add labels and title
plt.title('Average Number of Credit Inquiries by Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Average Number of Credit Inquiries')

plt.show()

```



- Customers with a **Poor** credit score have the highest average number of credit inquiries (30.74), while those with a **Good** credit score have the lowest (24.55). **

This suggests that individuals with poorer credit scores may be applying for credit more frequently, possibly indicating financial difficulties or instability.

'Credit_History_Age'

The **Credit_History_Age** column represents the total duration of a customer's credit history, originally provided in a string format like "X Years and Y Months."

To standardize this feature for analysis, a custom function was applied to convert the credit history from a string format (e.g., "X Years and Y Months") into a total number of months.

No unusual values were detected except for the missing data.

These missing values were handled using **KNN imputation**, ensuring consistency in the dataset for further analysis.

```
In [50]: # Check the column's unique values and percentage
get_value_count(train, 'Credit_History_Age')
```

Out[50]:

	Credit_History_Age	counts	percent
0	15 Years and 11 Months	446	0.49%
1	19 Years and 4 Months	445	0.49%
2	19 Years and 5 Months	444	0.49%
3	17 Years and 11 Months	443	0.49%
4	19 Years and 3 Months	441	0.48%
...
399	0 Years and 3 Months	20	0.02%
400	0 Years and 2 Months	15	0.02%
401	33 Years and 7 Months	14	0.02%
402	33 Years and 8 Months	12	0.01%
403	0 Years and 1 Months	2	0.00%

404 rows × 3 columns

In [51]:

```
# Check missing values and dtype
print('Remaining missing values in Train:', train['Credit_History_Age'].isna().sum())
print('Remaining missing values in Test:', test['Credit_History_Age'].isna().sum())
print('dtype: ', train['Credit_History_Age'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Credit_History_Age')
```

Remaining missing values in Train: 9030
 Remaining missing values in Test: 4470
 dtype: object

Out[51]:

```
{'Months',
 'Months ',
 'Months nan ',
 'Months nan nan ',
 'Months nan nan nan ',
 'Months nan nan nan nan ',
 'Months nan nan nan nan nan ',
 'Years and '}
```

In [52]:

```
===== User-Defined-Fonction =====
def history_age_month(age):
    """
    This function converts a credit history age string (in the format 'X Years and Y Months')
    into a total number of months.
    """
    try:
        # Extract the years part (before "and")
        years = int(re.findall(r'\d+', age.split("and")[0])[0])

        # Extract the months part (after "and")
        months = int(re.findall(r'\d+', age.split("and")[1])[0])

        # Convert years to months and add the months part
        return int(years * 12 + months)

    except (IndexError, ValueError, AttributeError):
        # Return NaN if the input string is not in the correct format or an error occurs
        return np.nan
=====
```

In [53]:

```
# Apply the fonction the the Credit_History_Age Column in Train and Test
train['Credit_History_Age'] = train['Credit_History_Age'].apply(history_age_month)
test['Credit_History_Age'] = test['Credit_History_Age'].apply(history_age_month)
```

In [54]:

```
print('dtype: ', train['Credit_History_Age'].dtypes)
get_value_count(train, 'Credit_History_Age').head()
```

dtype: float64

Out[54]:

	Credit_History_Age	counts	percent
0	191.000	446	0.49%
1	232.000	445	0.49%
2	233.000	444	0.49%
3	215.000	443	0.49%
4	231.000	441	0.48%

In [55]:

```
# Apply KNN imputation to the column in both train and test datasets
knn_impute_column(train, 'Credit_History_Age') # Train
knn_impute_column(test, 'Credit_History_Age') # Test

# Check Missing values
print('Remaining missing values in Train:', train['Credit_History_Age'].isna().sum())
print('Remaining missing values in Test:', test['Credit_History_Age'].isna().sum())
```

Remaining missing values in Train: 0

Remaining missing values in Test: 0

In [56]:

```
train['Credit_History_Age'].describe()
```

	count	mean	std	min	25%	50%	75%	max
Name: Credit_History_Age, dtype:	100000.000	221.195	95.131	1.000	154.000	221.195	292.000	404.000

In [57]:

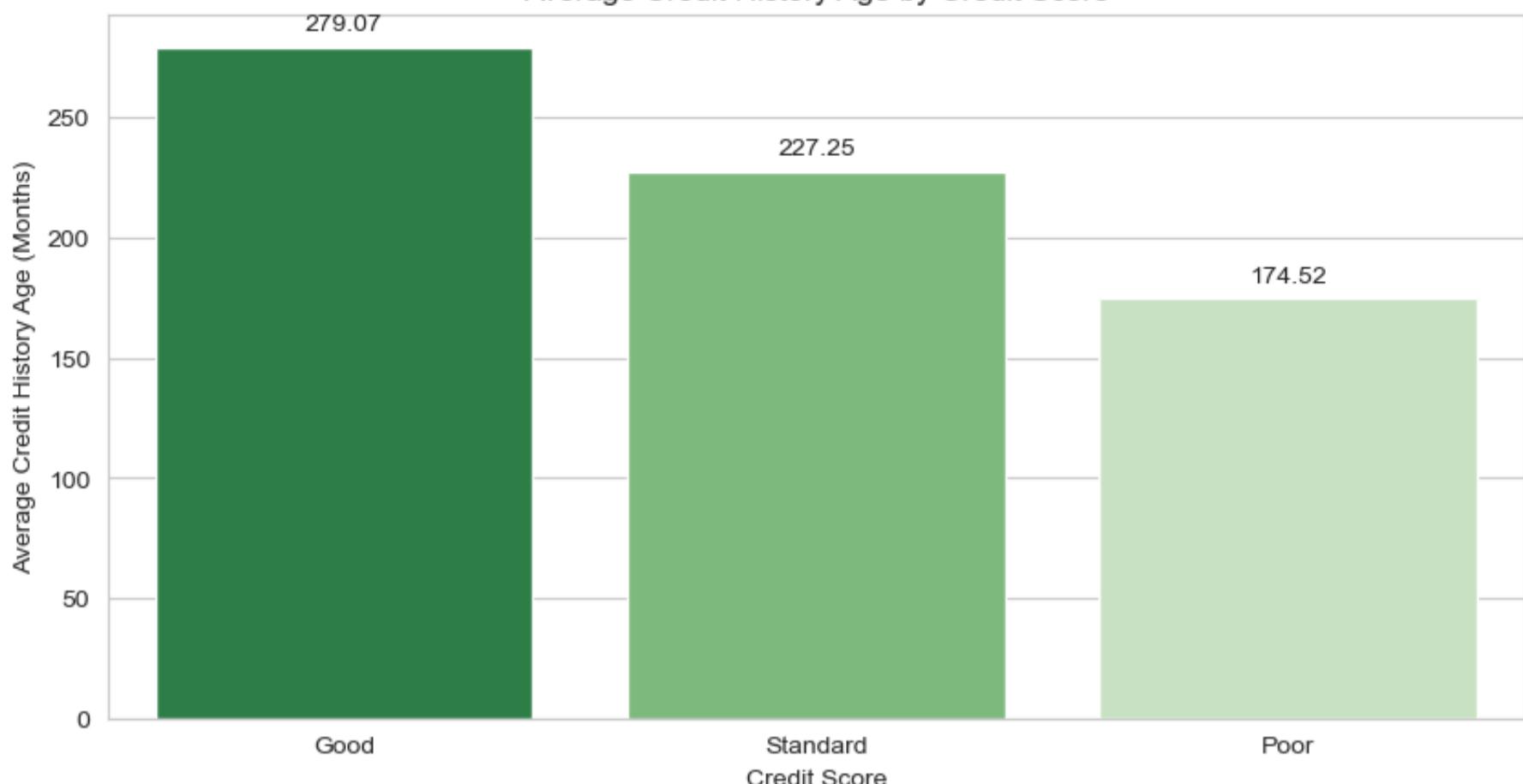
```
# Barplot showing the average Credit History Age by Credit Score category
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Credit_Score', y='Credit_History_Age', data=train, ci=None, palette='Greens_r')

# Add values on top of the bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 9), textcoords='offset points')

plt.title('Average Credit History Age by Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Average Credit History Age (Months)')

plt.show()
```

Average Credit History Age by Credit Score



- Customers with a **Good Credit Score** have a significantly longer average credit history age (279 months) compared to those with a **Standard** (227 months) or **Poor Credit Score** (174 months).

This suggests that a longer credit history may be associated with a better credit score.

'Amount_invested_monthly'

The **Amount_invested_monthly** column, represents the monthly investment amount, contained non-numeric characters and missing values.

These were cleaned, converted to **float**, and missing values were imputed using **KNN** to ensure consistency in the data.

```
In [58]: # Check the column's unique values and percentage
get_value_count(train, 'Amount_invested_monthly')
```

```
Out[58]:
```

	Amount_invested_monthly	counts	percent
0	_10000_	4305	4.51%
1	0.0	169	0.18%
2	80.41529543900253	1	0.00%
3	36.66235139442514	1	0.00%
4	89.7384893604547	1	0.00%
...
91044	36.541908593249026	1	0.00%
91045	93.45116318631192	1	0.00%
91046	140.80972223052834	1	0.00%
91047	38.73937670100975	1	0.00%
91048	167.1638651610451	1	0.00%

91049 rows × 3 columns

```
In [59]: # Check missing values and dtype
print('Remaining missing values in Train:', train['Amount_invested_monthly'].isna().sum())
print('Remaining missing values in Test:', test['Amount_invested_monthly'].isna().sum())
print('dtype: ', train['Amount_invested_monthly'].dtypes)
```

```
# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Amount_invested_monthly')
```

Remaining missing values in Train: 4479
 Remaining missing values in Test: 2271
 dtype: object

```
Out[59]: {' ',  
         ' __ ',  
         ' nan ',  
         ' nan __ ',  
         ' nan nan ',  
         ' nan nan __ ',  
         ' nan nan nan ',  
         ' nan nan nan nan ',  
         '.',  
         '__ ',  
         '__ __ ',  
         '__ nan ',  
         '__ nan __ ',  
         '__ nan nan ',  
         '__ nan nan __ '}
```

In [60]: # CLEANING UNUSUAL VALUES

```
#TRAIN
# 1. Remove non-numeric characters from the column in train dataset
train['Amount_invested_monthly'] = train['Amount_invested_monthly'].apply(lambda x: re.sub(r'^[^\d.]', '',  
# Replace empty strings with NaN
train['Amount_invested_monthly'].replace('', np.nan, inplace=True))

# 2. Convert dtype to float in train dataset
train['Amount_invested_monthly'] = train['Amount_invested_monthly'].astype(float)

# 3. Convert negatives like '-1' to positives
train['Amount_invested_monthly'] = train['Amount_invested_monthly'].abs()
```

In [61]: #TEST

```
# 1. Remove non-numeric characters from the column in test dataset
test['Amount_invested_monthly'] = test['Amount_invested_monthly'].apply(lambda x: re.sub(r'^[^\d.]', '', st
# Replace empty strings with NaN
test['Amount_invested_monthly'].replace('', np.nan, inplace=True)

# 2. Convert dtype to float in test dataset
test['Amount_invested_monthly'] = test['Amount_invested_monthly'].astype(float)

# 3. Convert negatives like '-1' to positives
test['Amount_invested_monthly'] = test['Amount_invested_monthly'].abs()
```

In [62]: # FILLING MISSING VALUES

```
# Apply KNN imputation to the column in both train and test datasets
knn_impute_column(train, 'Amount_invested_monthly') # Train
knn_impute_column(test, 'Amount_invested_monthly') # Test

# Check Missing values
print('Remaining missing values in Train:', train['Amount_invested_monthly'].isna().sum())
print('Remaining missing values in Test:', test['Amount_invested_monthly'].isna().sum())
```

Remaining missing values in Train: 0
 Remaining missing values in Test: 0

In [63]: train['Amount_invested_monthly'].describe()

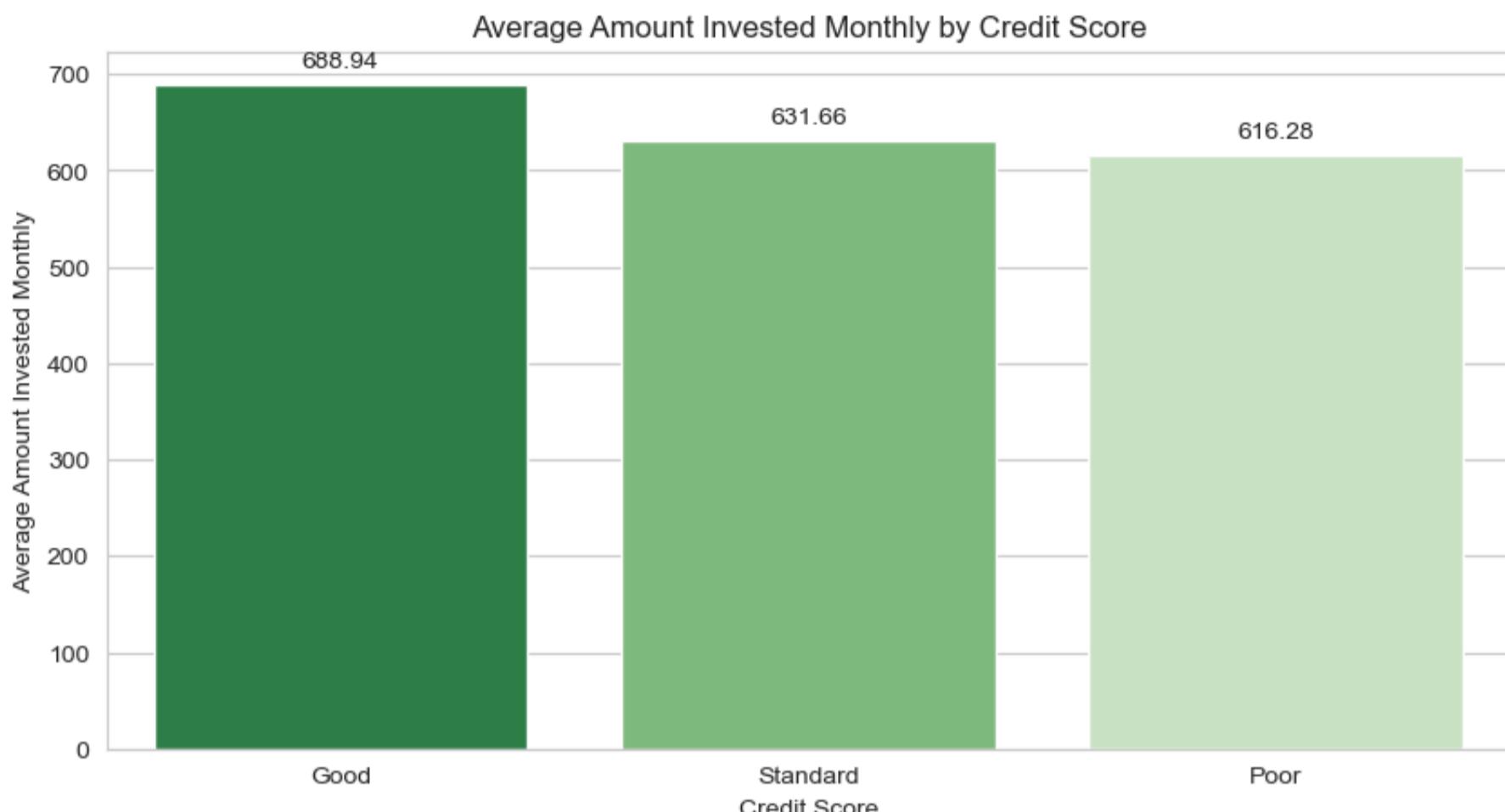
```
Out[63]: count    100000.000
mean      637.413
std       1997.035
min       0.000
25%      77.017
50%     143.128
75%     304.766
max     10000.000
Name: Amount_invested_monthly, dtype: float64
```

In [64]: # Barplot showing the average Amount Invested Monthly by Credit Score category
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Credit_Score', y='Amount_invested_monthly', data=train, ci=None, palette='Greens_r')

```
# Add values on top of the bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 9), textcoords='offset points')

plt.title('Average Amount Invested Monthly by Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Average Amount Invested Monthly')

plt.show()
```



- The plot shows that individuals with a "Good" credit score tend to invest more on a monthly basis, averaging **688.94** units.
- Those with "Standard" credit scores invest slightly less, at **631.66** units, and people with "Poor" credit scores invest the least, averaging **616.28** units.

This suggests a potential positive correlation between higher monthly investments and better credit scores.

'Monthly_Balance'

The **Monthly_Balance** column reflects the remaining balance in a customer's account at the end of each month after accounting for all expenses and income. It provides insights into a customer's financial stability and management.

- We cleaned the column by removing unusual values, such as excessively large or erroneous entries, and replaced them with `Nan` for appropriate handling.
- Missing values were handled by grouping the data based on relevant financial factors such as `Delay_from_due_date`, ensuring a logical imputation of missing values with the group's average balance.

This process ensured that the `Monthly_Balance` data remained consistent and reflective of actual customer financial behavior.

```
In [65]: # Check the column' unique values and percentage
get_value_count(train, 'Monthly_Balance')
```

Out[65]:

98792 rows × 3 columns

- The value 3333333333333333.0 in the Monthly_Balance column seems unusually large and likely erroneous compared to other values.

It has been replaced with `Nan` for proper handling later.

```
In [67]: # Check missing values and dtype
print('Remaining missing values in Train:', train['Monthly_Balance'].isna().sum())
print('Remaining missing values in Test:', test['Monthly_Balance'].isna().sum())
print('dtype: ', train['Monthly_Balance'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Monthly_Balance')
```

```
Remaining missing values in Train: 1209  
Remaining missing values in Test: 568  
dtype: object
```

```
Out[67]: {' ', ' nan ', ' nan nan ', ' nan nan nan ', ' nan nan nan nan ', '.'}
```

Cleaning Unusual Values:

```
In [68]: #TRAIN  
# 1. Identify and convert hexadecimal values to decimal  
train['Monthly_Balance'] = train['Monthly_Balance'].apply(lambda x: int(x, 16) if isinstance(x, str) and x[0] == '0x' else x)  
  
# 2. Remove non-numeric characters (keeping only digits and decimals)  
train['Monthly_Balance'] = train['Monthly_Balance'].apply(lambda x: re.sub(r'[^\d.]', '', str(x)))  
  
# 3. Replace empty strings with NaN  
train['Monthly_Balance'].replace('', np.nan, inplace=True)  
  
# 4. Convert dtype to float and coerce errors to NaN (invalid strings to NaN)  
train['Monthly_Balance'] = pd.to_numeric(train['Monthly_Balance'], errors='coerce')  
  
# 5. Convert negative values to positives if needed  
train['Monthly_Balance'] = train['Monthly_Balance'].abs()
```

```
In [69]: #TEST
# 1. Identify and convert hexadecimal values to decimal
test['Monthly_Balance'] = test['Monthly_Balance'].apply(lambda x: int(x, 16) if isinstance(x, str) and x.startswith('0x') else x)

# 2. Remove non-numeric characters (keeping only digits and decimals)
test['Monthly_Balance'] = test['Monthly_Balance'].apply(lambda x: re.sub(r'^0-9.]+', '', str(x)))

# 3. Replace empty strings with NaN
test['Monthly Balance'].replace('', np.nan, inplace=True)
```

```
# 4. Convert dtype to float and coerce errors to NaN (invalid strings to NaN)
test['Monthly_Balance'] = pd.to_numeric(test['Monthly_Balance'], errors='coerce')

# 5. Convert negative values to positives if needed
test['Monthly_Balance'] = test['Monthly_Balance'].abs()
```

In [70]: `get_value_count(train, 'Monthly_Balance').head()`

Out[70]:

	Monthly_Balance	counts	percent
0	312.494	1	0.00%
1	270.911	1	0.00%
2	254.971	1	0.00%
3	250.093	1	0.00%
4	289.755	1	0.00%

In [71]: `train['Monthly_Balance'].describe()`

Out[71]:

	count	mean	std	min	25%	50%	75%	max
	98791.000	402.551	213.925	0.008	270.107	336.731	470.263	1602.041
Name:	Monthly_Balance	dtype:	float64					

Filling Missing Values on Monthly_Balance:

The `Monthly_Balance` column shows how much money a customer has left in their account at the end of the month, after income and expenses.

- The `Delay_from_due_date` column represents how many days the customer delayed payment after the due date. This feature could be related to `Monthly_Balance` because consistent delays in payments may reflect financial stress or poor cash flow management, which might result in a lower monthly balance.

If a customer frequently delays payments, it's possible that they have less money left at the end of the month.

Therefore, it might be insightful to group by `Delay_from_due_date` to examine patterns and potentially use it for filling missing values in `Monthly_Balance`.

In [72]:

```
# numeric columns
numeric_columns = train.select_dtypes(include=[np.number])

# Calculate and sort the absolute correlation matrix
df_corr = numeric_columns.corr().abs().unstack().sort_values(kind="quicksort", ascending=False).reset_index

# Filter the columns that are related to the 'Monthly_Balance' column
df_corr[df_corr['level_0'] == 'Monthly_Balance'].head()
```

Out[72]:

	level_0	level_1	0
9	Monthly_Balance	Monthly_Balance	1.000
25	Monthly_Balance	Credit_History_Age	0.310
28	Monthly_Balance	Delay_from_due_date	0.279
30	Monthly_Balance	Credit_Utilization_Ratio	0.251
62	Monthly_Balance	Credit-Builder Loan	0.187

In [73]:

```
# Group by 'Delay_from_due_date' and fill missing values in 'Monthly_Balance' with the group's mean
#Train
train['Monthly_Balance'] = train.groupby('Delay_from_due_date')['Monthly_Balance'].transform(lambda x: x.fillna(x.mean()))

#Test
test['Monthly_Balance'] = train.groupby('Delay_from_due_date')['Monthly_Balance'].transform(lambda x: x.fillna(x.mean()))
```

```
# Check Missing values
print('Remaining missing values in Train:', train['Monthly_Balance'].isna().sum())
print('Remaining missing values in Test:', test['Monthly_Balance'].isna().sum())
```

Remaining missing values in Train: 0
 Remaining missing values in Test: 0

```
In [74]: # Barplot showing the average Monthly Balance by Credit Score category
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Credit_Score', y='Monthly_Balance', data=train, ci=None, palette='Greens_r')

# Add values on top of the bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 9), textcoords='offset points')

plt.title('Average Monthly Balance by Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Average Monthly Balance')

plt.show()
```



- The graph shows that customers with a **Good** credit score tend to have a significantly higher **average monthly balance** compared to those with **Standard** or **Poor** credit scores.
- Specifically, customers with good credit maintain an average balance of around 475, while those with poor credit have a lower average balance of around 345.

This suggests that higher credit scores may be associated with better financial management, leading to higher retained balances.

'Month'

The **Month** column in the dataset represents the month in which specific transactions or financial activities occurred. It could have provided important time-related context, allowing us to analyze patterns and trends over different months.

However, considering that all values in the **Month** column are identical across months, this feature doesn't provide any variation or useful information for the ANN model to learn from.

Since there's no distinction in behavior or patterns across different months, this column doesn't contribute to the model's ability to make better predictions and **will be dropped at the end of the cleaning process in this notebook!**

```
In [75]: # Check the column' unique values and percentage
get_value_count(train, 'Month')
```

Out[75]:

	Month	counts	percent
0	January	12500	12.50%
1	February	12500	12.50%
2	March	12500	12.50%
3	April	12500	12.50%
4	May	12500	12.50%
5	June	12500	12.50%
6	July	12500	12.50%
7	August	12500	12.50%

In [76]:

```
# Check the column' unique values and percentage
get_value_count(test, 'Month')
```

Out[76]:

	Month	counts	percent
0	September	12500	25.00%
1	October	12500	25.00%
2	November	12500	25.00%
3	December	12500	25.00%

In [77]:

```
# Check missing values and dtype
print('Remaining missing values in Train:', train['Month'].isna().sum())
print('Remaining missing values in Test:', test['Month'].isna().sum())
print('dtype: ', train['Month'].dtypes)
```

Remaining missing values in Train: 0

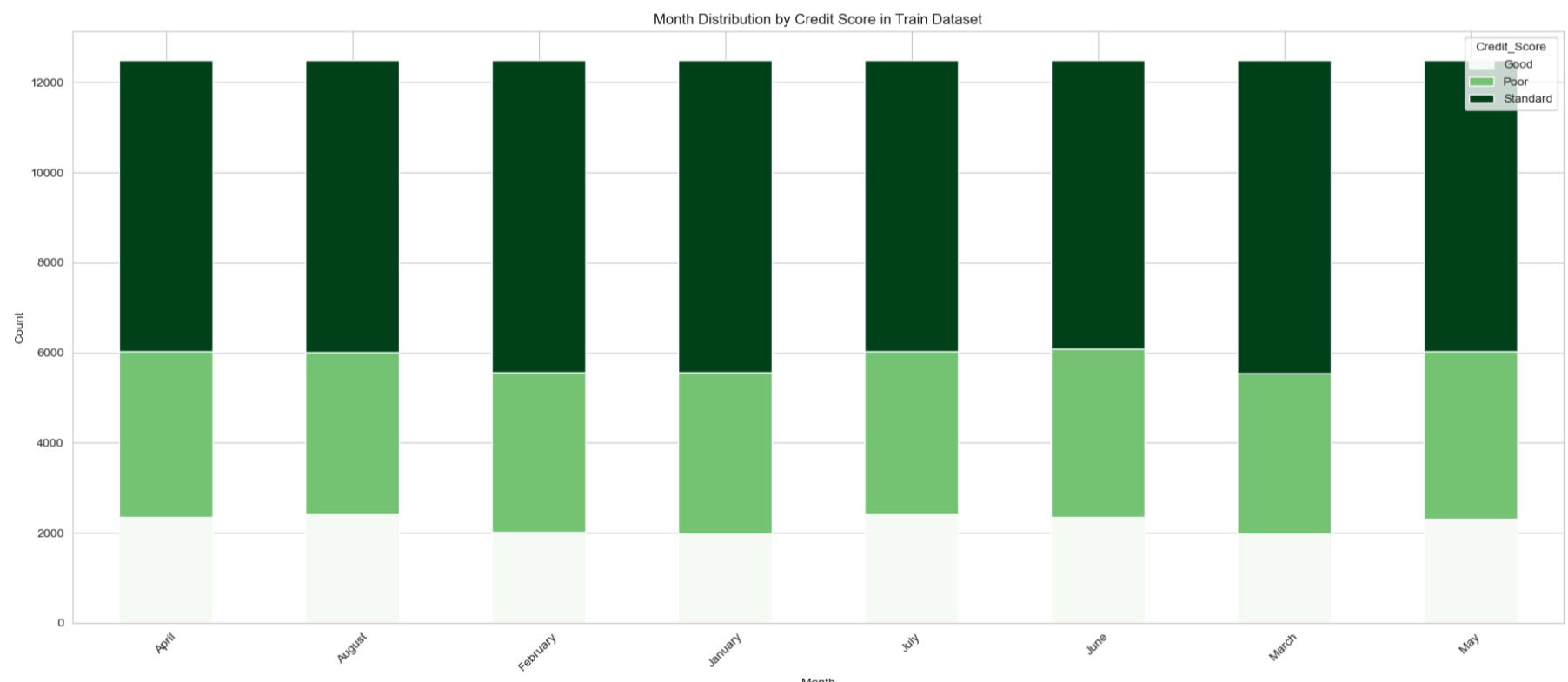
Remaining missing values in Test: 0

dtype: object

In [78]:

```
train.groupby(['Month', 'Credit_Score']).size().unstack().plot(kind='bar', stacked=True, figsize=(18,8), cm

# Add Labels and title
plt.xlabel('Month')
plt.ylabel('Count')
plt.title('Month Distribution by Credit Score in Train Dataset')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



- The chart shows that the distribution of credit scores (Good, Standard, and Poor) remains consistent across all months in the train dataset.

The proportions of each credit score category don't vary significantly month by month, indicating that the `Credit_Score` feature doesn't show a noticeable seasonal trend.

'Age'

The `Age` column represents the age of the customer, providing important demographic information that may influence financial behavior and creditworthiness.

- While there is no fixed "age cutoff," banks generally become stricter with lending after 65-70 years old, particularly for long-term loans.
- For this specific dataset, considering it focuses on financial activities such as loans, credit, and other banking services, a reasonable age limit could be around 65-70 years. This is in line with general financial industry practices, where banks and financial institutions tend to be more cautious in lending to individuals who are nearing or have surpassed typical retirement age.
- Age values below 18 should also be considered unrealistic for most types of loans, as people typically start borrowing after they reach legal adulthood.

To avoid losing data, age values outside a reasonable limit were replaced with `Nan` instead of being dropped. The data was then grouped by `Monthly_Balance`, `Credit_History_Age`, and `Delay_from_due_date` to fill the missing ages with the group's average. These columns were selected because individuals with similar income, credit history, and payment behavior tend to exhibit similar age patterns.

```
In [79]: # Check the column's unique values and percentage
get_value_count(train, 'Age')
```

```
Out[79]:
```

Age	counts	percent
0	38	2833 2.83%
1	28	2829 2.83%
2	31	2806 2.81%
3	26	2792 2.79%
4	32	2749 2.75%
...
1783	471	1 0.00%
1784	1520	1 0.00%
1785	8663	1 0.00%
1786	3363	1 0.00%
1787	1342	1 0.00%

1788 rows × 3 columns

```
In [80]: # Check missing values and dtype
print('Remaining missing values in Train:', train['Age'].isna().sum())
print('Remaining missing values in Test:', test['Age'].isna().sum())
print('dtype: ', train['Age'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Age')
```

```
Remaining missing values in Train: 0
Remaining missing values in Test: 0
dtype: object
```

```
Out[80]: {' ', '- ', '_ ', '_ - '}
```

Cleaning Unusual Values on 'Age':

```
In [81]: #TRAIN
# Remove non-numeric characters (keeping only digits and decimals)
train['Age'] = train['Age'].apply(lambda x: re.sub(r'^[^\d.]+', '', str(x)))
```

```
# Convert dtype to float and coerce errors to NaN (invalid strings to NaN)
train['Age'] = pd.to_numeric(train['Age'], errors='coerce')

# Replace unrealistic Age values with NaN (Age > 100 or Age < 0)
train.loc[(train['Age'] > 70) | (train['Age'] < 18), 'Age'] = np.nan
```

In [82]:

```
#TEST
# Remove non-numeric characters (keeping only digits and decimals)
test['Age'] = test['Age'].apply(lambda x: re.sub(r'^[0-9.]+', '', str(x)))

# Convert dtype to float and coerce errors to NaN (invalid strings to NaN)
test['Age'] = pd.to_numeric(test['Age'], errors='coerce')

# Replace unrealistic Age values with NaN (Age > 100 or Age < 0)
test.loc[(test['Age'] > 70) | (test['Age'] < 18), 'Age'] = np.nan
```

In [83]:

```
get_value_count(train, 'Age').head()
```

Out[83]:

	Age	counts	percent
0	38.000	2994	3.27%
1	28.000	2968	3.24%
2	31.000	2955	3.23%
3	26.000	2945	3.22%
4	32.000	2884	3.15%

In [84]:

```
train['Age'].describe()
```

Out[84]:

	count	mean	std	min	25%	50%	75%	max	Name: Age, dtype: float64
	91513.000	34.426	10.114	18.000	26.000	34.000	42.000	56.000	

Filling Missing Values on 'Age':

To avoid losing data, age values outside a reasonable limit (18-70) were replaced with NaN instead of being dropped.

In [85]:

```
print('Remaining missing values in Train:', train['Age'].isna().sum())
print('Remaining missing values in Test:', test['Age'].isna().sum())
```

Remaining missing values in Train: 8487
 Remaining missing values in Test: 3907

In [86]:

```
# Check the correlation between Age and other Features
numeric_columns = train.select_dtypes(include=[np.number])
# Calculate and sort the absolute correlation matrix
df_corr = numeric_columns.corr().abs().unstack().sort_values(kind="quicksort", ascending=False).reset_index
# Filter the columns that are related to the 'Monthly_Balance' column
df_corr[df_corr['level_0'] == 'Age'].head()
```

Out[86]:

	level_0	level_1	0
0	Age		1.000
66	Age	Credit_History_Age	0.178
82	Age	Delay_from_due_date	0.138
94	Age	Monthly_Balance	0.086
97	Age	Personal Loan	0.084

In [87]:

```
train.groupby(['Credit_History_Age', 'Delay_from_due_date'])['Age'].mean()
```

```
Out[87]: Credit_History_Age  Delay_from_due_date
1.000           33            31.000
2.000           27            35.000
                   33            29.000
                   36             NaN
                   37            36.000
                           ...
404.000          9             32.000
                   11            25.000
                   12            22.500
                   26            54.000
                   30            41.000
Name: Age, Length: 18345, dtype: float64
```

```
In [88]: # Fill the missing values based on the train dataset
#Train
train['Age'] = train.groupby(['Credit_History_Age', 'Delay_from_due_date'])['Age'].transform(lambda x: x.fillna(x.mean()))
# For remaining missing values, fill with the global mean
train['Age'].fillna(train['Age'].mean(), inplace=True)

#Test
test['Age'] = train.groupby(['Credit_History_Age', 'Delay_from_due_date'])['Age'].transform(lambda x: x.fillna(x.mean()))
# For remaining missing values, fill with the global mean
test['Age'].fillna(train['Age'].mean(), inplace=True)

# Check the results
train['Age'].describe()
```

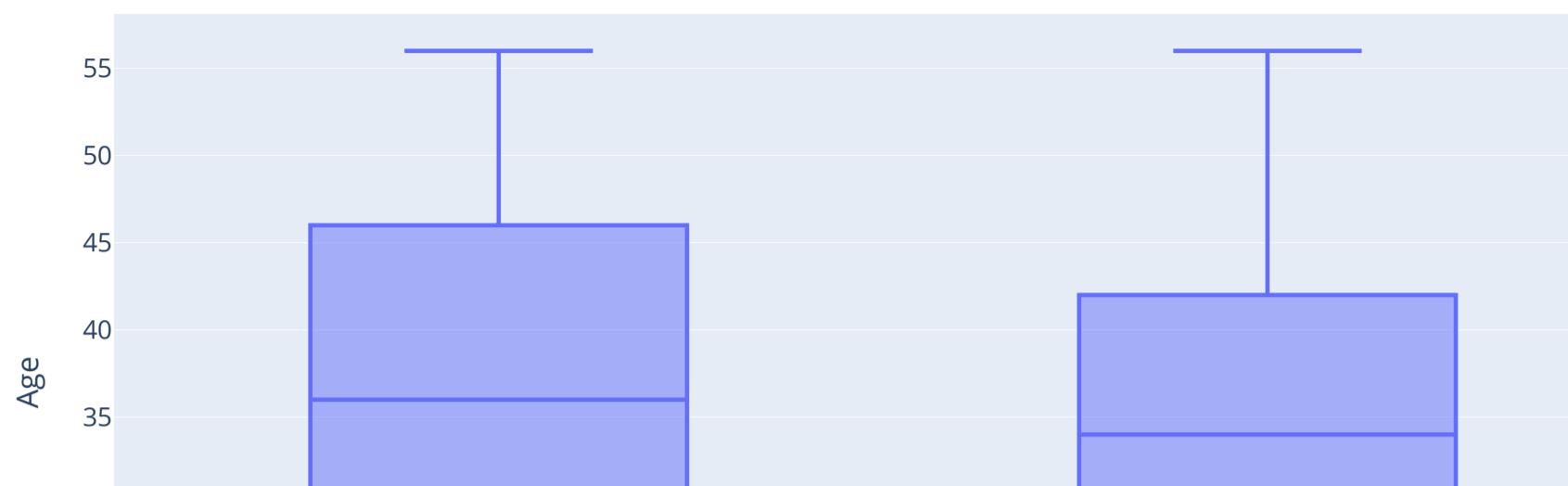
```
Out[88]: count    100000.000
mean        34.335
std         9.784
min         18.000
25%        26.000
50%        34.000
75%        42.000
max         56.000
Name: Age, dtype: float64
```

```
In [89]: # Check missing values and dtype
print('Remaining missing values in Train:', train['Age'].isna().sum())
print('Remaining missing values in Test:', test['Age'].isna().sum())
print('dtype: ', train['Age'].dtype)
```

Remaining missing values in Train: 0
 Remaining missing values in Test: 0
 dtype: float64

```
In [90]: import plotly.express as px
# Creating a boxplot to visualize the distribution of 'Age' based on 'Credit_Score'
fig = px.box(train, x='Credit_Score', y='Age', title='Age Distribution by Credit Score')
fig.show()
```

Age Distribution by Credit Score



'Occupation'

The `Occupation` column represents the type of job or profession of the customer. It provides important demographic and financial context as different occupations often have different levels of income, spending habits, and risk profiles, which are crucial for credit scoring and other financial analyses.

```
In [91]: # Check the column's unique values and percentage
get_value_count(train, 'Occupation')
```

```
Out[91]:
```

	Occupation	counts	percent
0	_____	7062	7.06%
1	Lawyer	6575	6.58%
2	Architect	6355	6.35%
3	Engineer	6350	6.35%
4	Scientist	6299	6.30%
5	Mechanic	6291	6.29%
6	Accountant	6271	6.27%
7	Developer	6235	6.24%
8	Media_Manager	6232	6.23%
9	Teacher	6215	6.21%
10	Entrepreneur	6174	6.17%
11	Doctor	6087	6.09%
12	Journalist	6085	6.08%
13	Manager	5973	5.97%
14	Musician	5911	5.91%
15	Writer	5885	5.88%

```
In [92]: # Check missing values and dtype
print('Remaining missing values in Train:', train['Occupation'].isna().sum())
print('Remaining missing values in Test:', test['Occupation'].isna().sum())
print('dtype: ', train['Occupation'].dtypes)
```

Remaining missing values in Train: 0
Remaining missing values in Test: 0
dtype: object

Cleaning Unusual Values:

```
In [93]: # Replace '---' with a new category 'Unknown'
train['Occupation'].replace('_____', 'Unknown', inplace=True)
test['Occupation'].replace('_____', 'Unknown', inplace=True)
```

```
In [94]: # Check the column's unique values and percentage
get_value_count(train, 'Occupation')
```

Out[94]:

	Occupation	counts	percent
0	Unknown	7062	7.06%
1	Lawyer	6575	6.58%
2	Architect	6355	6.35%
3	Engineer	6350	6.35%
4	Scientist	6299	6.30%
5	Mechanic	6291	6.29%
6	Accountant	6271	6.27%
7	Developer	6235	6.24%
8	Media_Manager	6232	6.23%
9	Teacher	6215	6.21%
10	Entrepreneur	6174	6.17%
11	Doctor	6087	6.09%
12	Journalist	6085	6.08%
13	Manager	5973	5.97%
14	Musician	5911	5.91%
15	Writer	5885	5.88%

```
In [95]: get_value_count(test, 'Occupation')
```

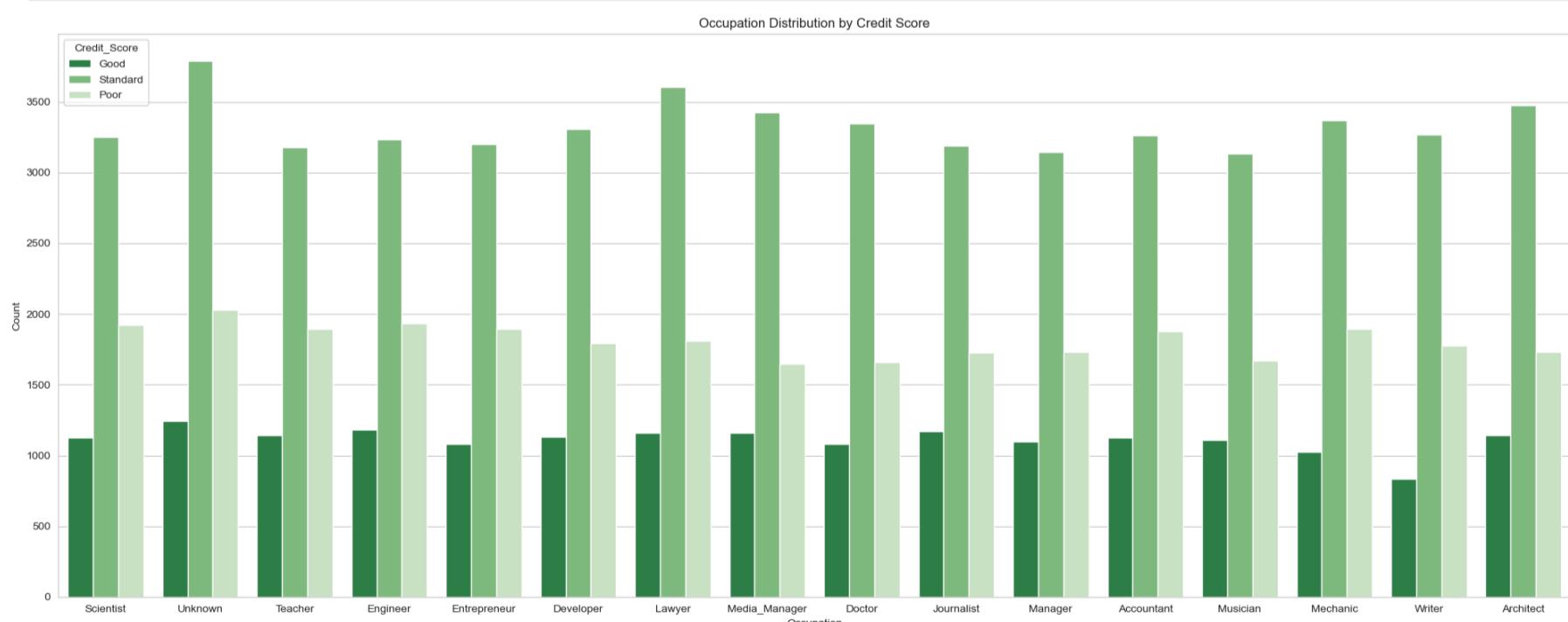
Out[95]:

	Occupation	counts	percent
0	Unknown	3438	6.88%
1	Lawyer	3324	6.65%
2	Engineer	3212	6.42%
3	Architect	3195	6.39%
4	Mechanic	3168	6.34%
5	Developer	3146	6.29%
6	Accountant	3133	6.27%
7	Media_Manager	3130	6.26%
8	Scientist	3104	6.21%
9	Teacher	3103	6.21%
10	Entrepreneur	3103	6.21%
11	Journalist	3037	6.07%
12	Doctor	3027	6.05%
13	Manager	3000	6.00%
14	Musician	2947	5.89%
15	Writer	2933	5.87%

In [96]:

```
# Create a count plot to visualize 'Occupation' distribution based on 'Credit_Score' (target feature)
plt.figure(figsize=(20, 8))
sns.countplot(x='Occupation', hue='Credit_Score', data=train, palette='Greens_r')

# Add Labels and title
plt.xlabel('Occupation')
plt.ylabel('Count')
plt.title('Occupation Distribution by Credit Score')
# plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



- The graph shows that across most occupations, the majority of individuals fall into the **Standard** credit score category.
- Occupations like **Media_Manager**, **Lawyer**, and **Unknown** have the highest counts in the **Standard** category.
- There are fewer individuals with a **Good** credit score across all occupations, while the **Poor** credit score group is relatively small, except for certain roles like **Mechanic** and **Musician**, which have a more notable share of individuals with poor credit.

'Num_Bank_Accounts'

The `Num_Bank_Accounts` column represents the number of bank accounts a customer holds. Upon reviewing the data, an unusual value of `-1` was identified. Since it is not possible to have a negative number of bank accounts, this value likely represents missing or unknown data.

- Replace `-1` with `0`, as it is reasonable to assume these customers do not have any bank accounts.

This approach keeps the data clean and accurate without introducing missing values (`NaN`).

By replacing `-1` with `0`, the dataset better reflects the customers' financial status and ensures consistency for analysis.

```
In [97]: # Check the column's unique values and percentage
get_value_count(train, 'Num_Bank_Accounts')
```

```
Out[97]:
```

	<code>Num_Bank_Accounts</code>	<code>counts</code>	<code>percent</code>
0	6	13001	13.00%
1	7	12823	12.82%
2	8	12765	12.77%
3	4	12186	12.19%
4	5	12118	12.12%
...
938	1626	1	0.00%
939	1470	1	0.00%
940	887	1	0.00%
941	211	1	0.00%
942	697	1	0.00%

943 rows × 3 columns

```
In [98]: # Check missing values and dtype
print('Remaining missing values in Train:', train['Num_Bank_Accounts'].isna().sum())
print('Remaining missing values in Test:', test['Num_Bank_Accounts'].isna().sum())
print('dtype: ', train['Num_Bank_Accounts'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Num_Bank_Accounts')
```

```
Remaining missing values in Train: 0
Remaining missing values in Test: 0
dtype: int64
```

```
Out[98]: {' ', '-'}  

```

```
In [99]: # Print the list of unique negative values
print('List of unique negative values:', train[train['Num_Bank_Accounts'] < 0]['Num_Bank_Accounts'].unique())
# Print the count of rows with negative values
print('Number of unique negative values:', train[train['Num_Bank_Accounts'] < 0]['Num_Bank_Accounts'].count)
```

```
List of unique negative values: [-1]
Number of unique negative values: 21
```

```
In [100...]: # Replace -1 values with 0, assuming no bank accounts
train['Num_Bank_Accounts'].replace(-1, 0, inplace=True) #Train
test['Num_Bank_Accounts'].replace(-1, 0, inplace=True) #Test
```

```
In [101...]: # Check unusual values
print('List of unique negative values:', train[train['Num_Bank_Accounts'] < 0]['Num_Bank_Accounts'].unique())
print('Number of unique negative values:', train[train['Num_Bank_Accounts'] < 0]['Num_Bank_Accounts'].count)
```

```
List of unique negative values: []
Number of unique negative values: 0
```

```
In [102...]: # Barplot showing the average Number of Bank Accounts by Credit Score category
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Credit_Score', y='Num_Bank_Accounts', data=train, ci=None, palette='Greens_r')

# Add values on top of the bars
```

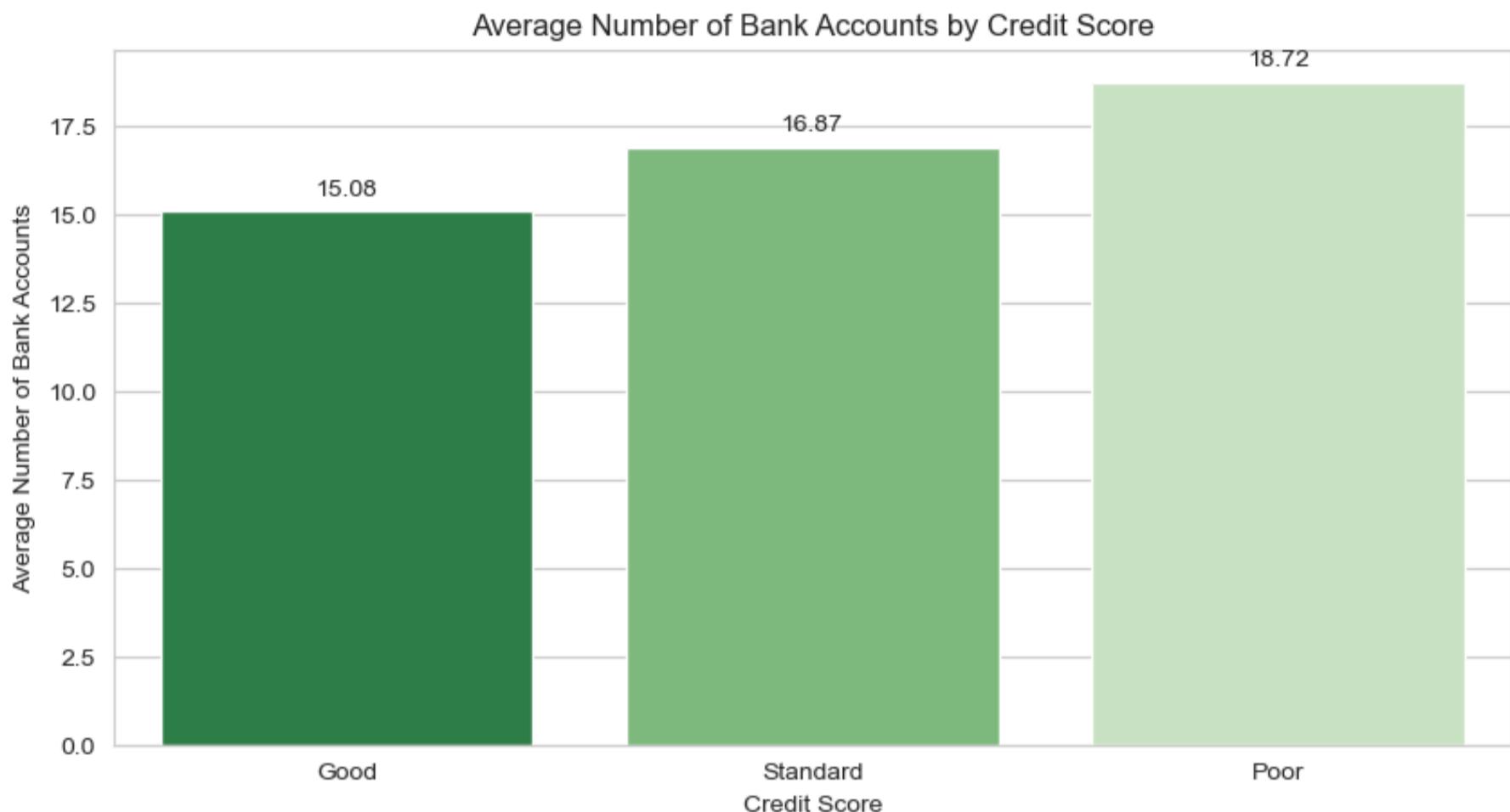
```

for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 9), textcoords='offset points')

plt.title('Average Number of Bank Accounts by Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Average Number of Bank Accounts')

plt.show()

```



- The graph shows that customers with a **Poor** credit score tend to have a higher average number of bank accounts (18.72) compared to those with **Standard** (16.87) and **Good** (15.08) credit scores.

This suggests a possible correlation between having more bank accounts and a poorer credit score.

'Num_Credit_Card'

```
In [103...]: # Check the column's unique values and percentage
get_value_count(train, 'Num_Credit_Card')
```

```
Out[103...]:
```

Num_Credit_Card	counts	percent
0	5	18459 18.46%
1	7	16615 16.61%
2	6	16559 16.56%
3	4	14030 14.03%
4	3	13277 13.28%
...
1174	791	1 0.00%
1175	1118	1 0.00%
1176	657	1 0.00%
1177	640	1 0.00%
1178	679	1 0.00%

1179 rows × 3 columns

```
In [104...]: # Check missing values and dtype
print('Remaining missing values in Train:', train['Num_Credit_Card'].isna().sum())
print('Remaining missing values in Test:', test['Num_Credit_Card'].isna().sum())
print('dtype: ', train['Num_Credit_Card'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Num_Credit_Card')
```

Remaining missing values in Train: 0
 Remaining missing values in Test: 0
 dtype: int64

Out[104...]: {' '}

```
In [105...]: # Print the list of unique negative values
print('List of unique negative values:', train[train['Num_Credit_Card'] < 0]['Num_Credit_Card'].unique())
# Print the count of rows with negative values
print('Number of unique negative values:', train[train['Num_Credit_Card'] < 0]['Num_Credit_Card'].count())
```

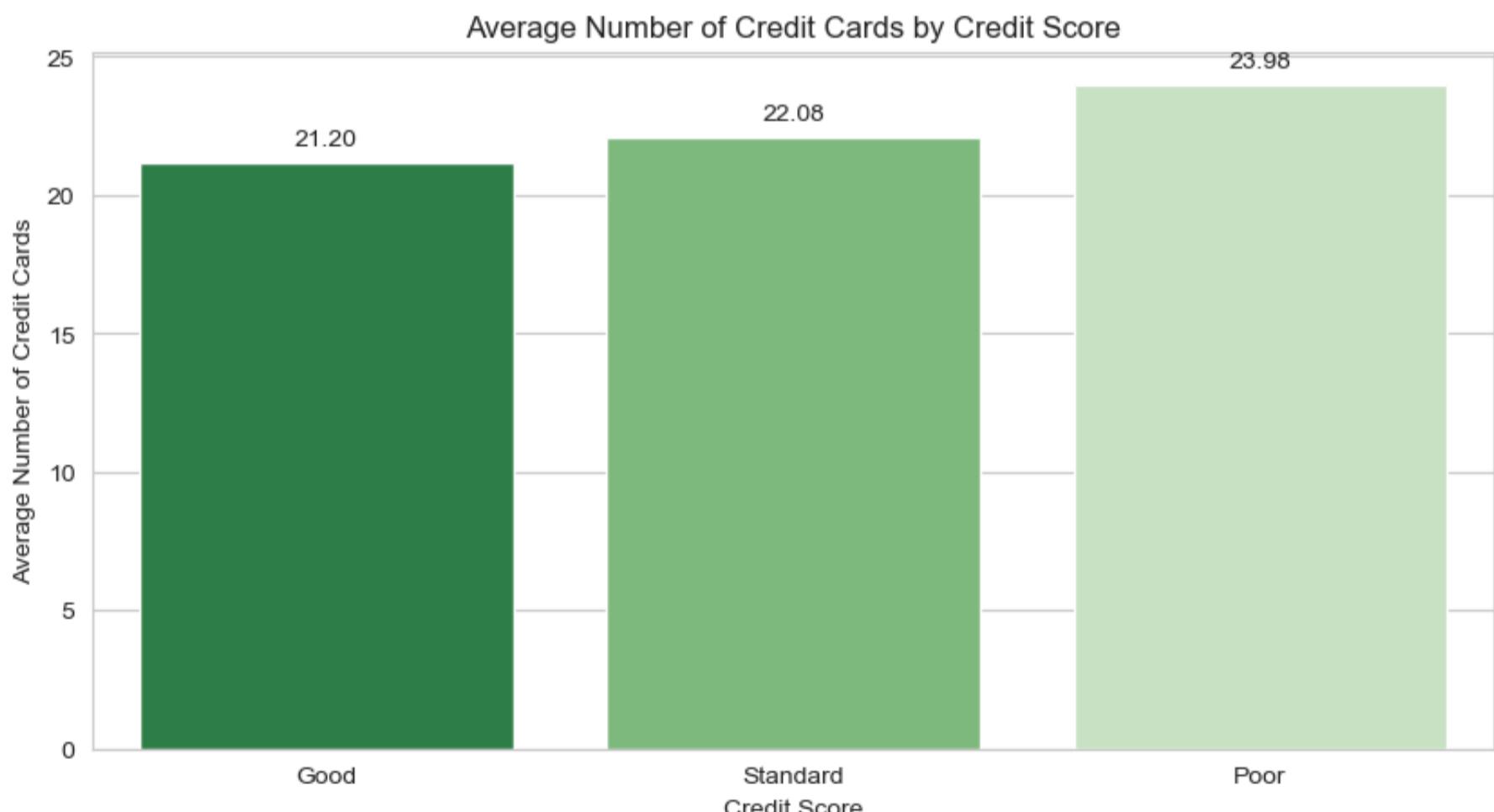
List of unique negative values: []
 Number of unique negative values: 0

```
In [106...]: # Barplot showing the average Number of Credit Cards by Credit Score category
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Credit_Score', y='Num_Credit_Card', data=train, ci=None, palette='Greens_r')

# Add values on top of the bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 9), textcoords='offset points')

plt.title('Average Number of Credit Cards by Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Average Number of Credit Cards')

plt.show()
```



- There is a possible correlation between having more credit cards and a poorer credit score, similar to the behavior observed with the number of bank accounts.

'Num_of_Loan'

```
In [107...]: # Check the column' unique values and percentage
get_value_count(train, 'Num_of_Loan')
```

Out[107...]

	Num_of_Loan	counts	percent
0	3	14386	14.39%
1	2	14250	14.25%
2	4	14016	14.02%
3	0	10380	10.38%
4	1	10083	10.08%
...
429	1320	1	0.00%
430	103	1	0.00%
431	1444	1	0.00%
432	392	1	0.00%
433	966	1	0.00%

434 rows × 3 columns

In [108...]

```
# Check missing values and dtype
print('Remaining missing values in Train:', train['Num_of_Loan'].isna().sum())
print('Remaining missing values in Test:', test['Num_of_Loan'].isna().sum())
print('dtype: ', train['Num_of_Loan'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Num_of_Loan')
```

Remaining missing values in Train: 0
 Remaining missing values in Test: 0
 dtype: object

Out[108...]

{' ', '- ', '_ ', '_ - '}

In [109...]

```
#TRAIN
# Remove non-numeric characters (keeping only digits and decimals)
train['Num_of_Loan'] = train['Num_of_Loan'].apply(lambda x: re.sub(r'^[^\d.]+' , '' , str(x)))

# Replace empty strings with NaN
train['Num_of_Loan'].replace('' , np.nan , inplace=True)

# Convert dtype to float and coerce errors to NaN (invalid strings to NaN)
train['Num_of_Loan'] = pd.to_numeric(train['Num_of_Loan'] , errors='coerce')

# Convert negative values to positives if needed
train['Num_of_Loan'] = train['Num_of_Loan'].abs()
```

In [110...]

```
#TEST
# Remove non-numeric characters (keeping only digits and decimals)
test['Num_of_Loan'] = test['Num_of_Loan'].apply(lambda x: re.sub(r'^[^\d.]+' , '' , str(x)))

# Replace empty strings with NaN
test['Num_of_Loan'].replace('' , np.nan , inplace=True)

# Convert dtype to float and coerce errors to NaN (invalid strings to NaN)
test['Num_of_Loan'] = pd.to_numeric(test['Num_of_Loan'] , errors='coerce')

# Convert negative values to positives if needed
test['Num_of_Loan'] = test['Num_of_Loan'].abs()
```

In [111...]

```
# Check missing values and dtype
print('Remaining missing values in Train:' , train['Num_of_Loan'].isna().sum())
print('Remaining missing values in Test:' , test['Num_of_Loan'].isna().sum())
print('dtype: ' , train['Num_of_Loan'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Num_of_Loan')
```

Remaining missing values in Train: 0
 Remaining missing values in Test: 0
 dtype: int64

Out[111... {' '}

```
In [112... # Barplot showing the average Number of Loans by Credit Score category
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Credit_Score', y='Num_of_Loan', data=train, ci=None, palette='Greens_r')

# Add values on top of the bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 9), textcoords='offset points')

plt.title('Average Number of Loans by Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Average Number of Loans')

plt.show()
```



- Similar correlation between having more Loans and a poorer credit score, similar to the behavior observed with the number of bank accounts and number of credit cards.

'Interest_Rate'

The **Interest_Rate** column represents the interest rate applied to loans or credit taken by the customer.

It provides critical information about the cost of borrowing for each customer.

Upon reviewing the data, no unusual or missing values were found in this column, indicating that all entries are valid and complete.

This ensures that the interest rate data can be used reliably in further analysis.

```
In [113... # Check the column's unique values and percentage
get_value_count(train, 'Interest_Rate')
```

Out[113...]

	Interest_Rate	counts	percent
0	8	5012	5.01%
1	5	4979	4.98%
2	6	4721	4.72%
3	12	4540	4.54%
4	10	4540	4.54%
...
1745	4995	1	0.00%
1746	1899	1	0.00%
1747	2120	1	0.00%
1748	5762	1	0.00%
1749	5729	1	0.00%

1750 rows × 3 columns

In [114...]

```
# Check missing values and dtype
print('Remaining missing values in Train:', train['Interest_Rate'].isna().sum())
print('Remaining missing values in Test:', test['Interest_Rate'].isna().sum())
print('dtype: ', train['Interest_Rate'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Interest_Rate')
```

Remaining missing values in Train: 0
 Remaining missing values in Test: 0
 dtype: int64

Out[114...]

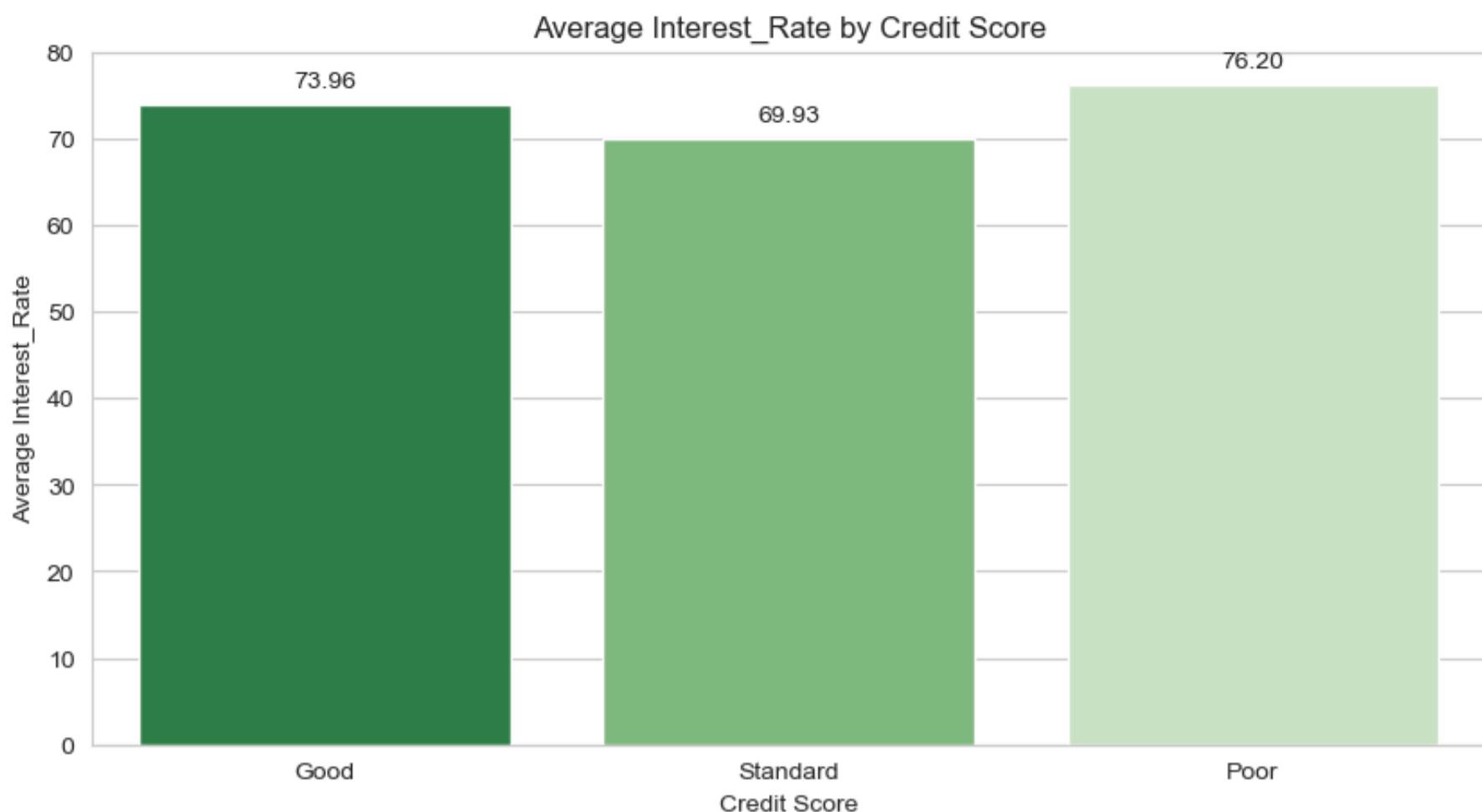
```
{' '}
```

Barplot showing the average Interest_Rate by Credit Score category
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Credit_Score', y='Interest_Rate', data=train, ci=None, palette='Greens_r')

Add values on top of the bars
for p in ax.patches:
 ax.annotate(format(p.get_height(), '.2f'),
 (p.get_x() + p.get_width() / 2., p.get_height()),
 ha='center', va='center',
 xytext=(0, 9), textcoords='offset points')

plt.title('Average Interest_Rate by Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Average Interest_Rate')

plt.show()



- Individuals with a **Poor** credit score have the highest average interest rate (76.20), while those with a **Standard** credit score have the lowest (69.93).
- Interestingly, customers with a **Good** credit score still face a relatively high interest rate (73.96), suggesting possible inconsistencies in how interest rates are assigned across credit score categories.

'Delay_from_due_date'

The `Delay_from_due_date` column represents the number of days a payment is overdue, which is a critical indicator of a customer's payment behavior and financial responsibility.

This feature is important for analysis and modeling as it directly reflects the risk of default, making it highly relevant for credit scoring and risk assessment models. After reviewing the data, only unusual negative values are present and there are no missing values.

- It's possible that the negative values were entered by mistake, especially since the `Delay_from_due_date` column should represent the number of days a payment is overdue, which is inherently a positive or zero value.
- Since different negative values are present (e.g., -1, -2, -3, etc.), this could indicate that they were intended to be positive values but were incorrectly entered as negative.

```
In [116]: # Check the column's unique values and percentage
get_value_count(train, 'Delay_from_due_date').head(10)
```

	Delay_from_due_date	counts	percent
0	15	3596	3.60%
1	13	3424	3.42%
2	8	3324	3.32%
3	14	3313	3.31%
4	10	3281	3.28%
5	7	3234	3.23%
6	9	3233	3.23%
7	11	3182	3.18%
8	12	3141	3.14%
9	6	3137	3.14%

```
In [117...]: # Check missing values and dtype
print('Remaining missing values in Train:', train['Delay_from_due_date'].isna().sum())
print('Remaining missing values in Test:', test['Delay_from_due_date'].isna().sum())
print('dtype: ', train['Delay_from_due_date'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Delay_from_due_date')

Remaining missing values in Train: 0
Remaining missing values in Test: 0
dtype: int64

Out[117...]: {' ', ' -'}
```

```
In [118...]: # Print the list of unique negative values
print('List of unique negative values:', train[train['Delay_from_due_date'] < 0]['Delay_from_due_date'].unique())
# Print the count of rows with negative values
print('Number of unique negative values:', train[train['Delay_from_due_date'] < 0]['Delay_from_due_date'].count())

List of unique negative values: [-1 -2 -3 -5 -4]
Number of unique negative values: 591
```

```
In [119...]: # Convert negative values to positives if needed
train['Delay_from_due_date'] = train['Delay_from_due_date'].abs() # Train
test['Delay_from_due_date'] = test['Delay_from_due_date'].abs() # Test
```

```
In [120...]: # Print the list of unique negative values
print('List of unique negative values:', train[train['Delay_from_due_date'] < 0]['Delay_from_due_date'].unique())
# Print the count of rows with negative values
print('Number of unique negative values:', train[train['Delay_from_due_date'] < 0]['Delay_from_due_date'].count())

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Delay_from_due_date')

List of unique negative values: []
Number of unique negative values: 0

Out[120...]: {' '}
```

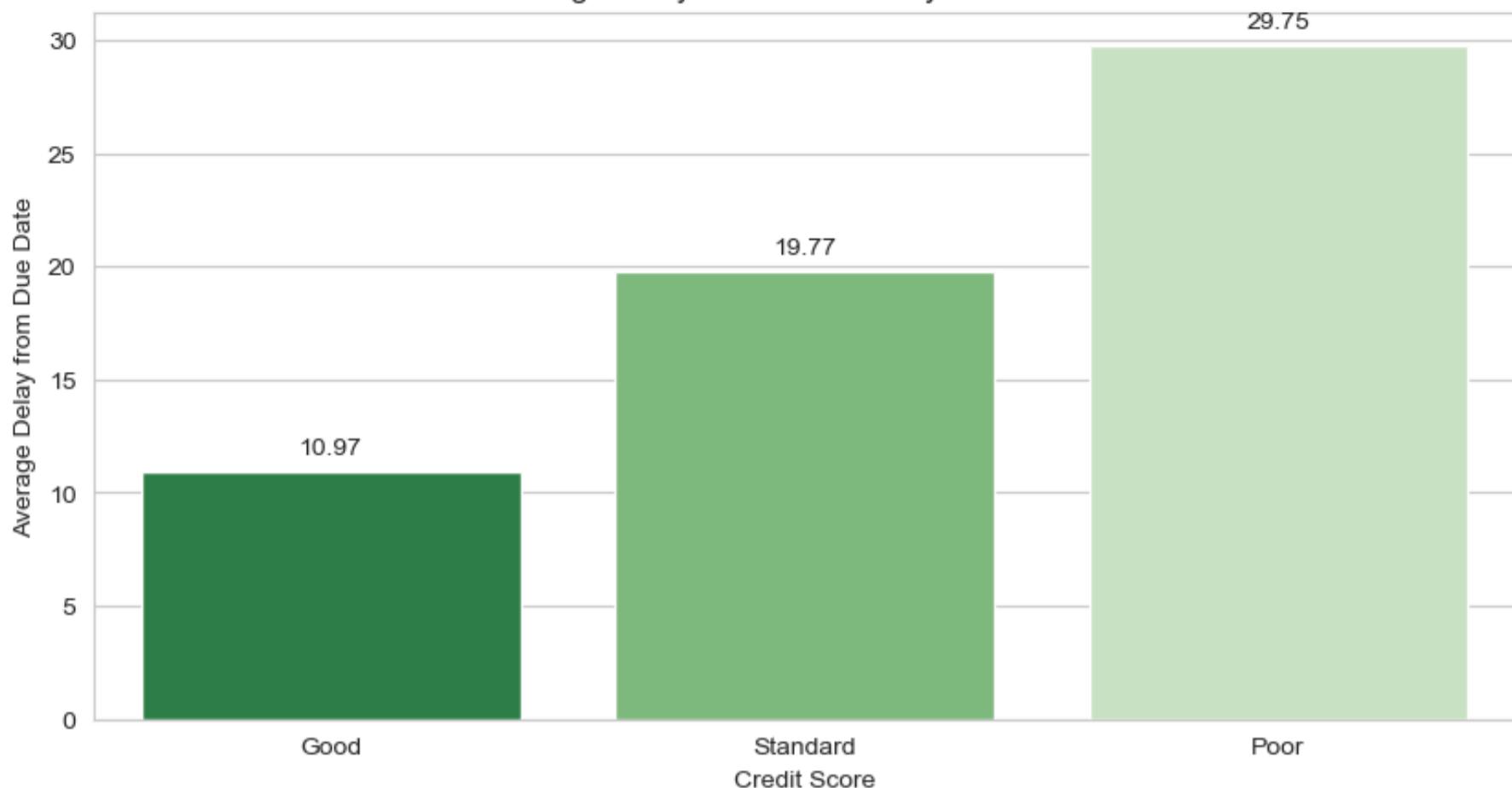
```
In [121...]: # Barplot showing the average Delay from Due Date by Credit Score category
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Credit_Score', y='Delay_from_due_date', data=train, ci=None, palette='Greens_r')

# Add values on top of the bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 9), textcoords='offset points')

plt.title('Average Delay from Due Date by Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Average Delay from Due Date')

plt.show()
```

Average Delay from Due Date by Credit Score



- The graph shows that customers with a **Poor** credit score have the longest average delay in payments (29.75 days), while those with a **Good** credit score have the shortest delay (10.97 days).

This suggests that late payments are strongly correlated with a poorer credit score.

'Changed_Credit_Limit'

The **Changed_Credit_Limit** column tracks changes in a customer's credit limit, with positive values indicating an increase and negative values indicating a decrease.

- Unusual characters like ' ', '- ', '_ ', '_ - ', '_ _ ' were found and cleaned.
- The `_` values were replaced with `0`, assuming they represent **no change** in the credit limit. This keeps the data numeric and clean for analysis.
- Negative values were not changed, as they are valid and show a **reduction** in the credit limit, which is important for understanding customer risk.

In [122...]

```
# Check the column's unique values and percentage
get_value_count(train, 'Changed_Credit_Limit')
```

Out[122...]

	Changed_Credit_Limit	counts	percent
0	-	2091	2.09%
1	8.22	133	0.13%
2	11.5	127	0.13%
3	11.32	126	0.13%
4	7.35	121	0.12%
...
4379	-1.84	1	0.00%
4380	0.8899999999999999	1	0.00%
4381	28.06	1	0.00%
4382	1.5599999999999996	1	0.00%
4383	21.17	1	0.00%

4384 rows × 3 columns

In [123...]

```
# Check missing values and dtype
print('Remaining missing values in Train:', train['Changed_Credit_Limit'].isna().sum())
print('Remaining missing values in Test:', test['Changed_Credit_Limit'].isna().sum())
print('dtype: ', train['Changed_Credit_Limit'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Changed_Credit_Limit')
```

Remaining missing values in Train: 0

Remaining missing values in Test: 0

dtype: object

Out[123...]: {' ', ' -', ' _', ' _ -', ' _ _', '.'}

In [124...]

```
#TRAIN: Cleaning
# Remove non-numeric characters (keeping only digits and decimals)
train['Changed_Credit_Limit'] = train['Changed_Credit_Limit'].apply(lambda x: re.sub(r'^0-9.-]+', '', str(x))

# Replace empty strings with 0 !!!!!!!!
train['Changed_Credit_Limit'].replace('', 0, inplace=True)

# Convert dtype to float and coerce errors to NaN (invalid strings to NaN)
train['Changed_Credit_Limit'] = pd.to_numeric(train['Changed_Credit_Limit'], errors='coerce')
```

In [125...]

```
#TEST: Cleaning
# Remove non-numeric characters (keeping only digits and decimals)
test['Changed_Credit_Limit'] = test['Changed_Credit_Limit'].apply(lambda x: re.sub(r'^0-9.-]+', '', str(x))

# Replace empty strings with 0 !!!!!!!!
test['Changed_Credit_Limit'].replace('', 0, inplace=True)

# Convert dtype to float and coerce errors to NaN (invalid strings to NaN)
test['Changed_Credit_Limit'] = pd.to_numeric(test['Changed_Credit_Limit'], errors='coerce')
```

In [126...]

```
# Check the column
get_value_count(train, 'Changed_Credit_Limit')
```

Out[126...]:

	Changed_Credit_Limit	counts	percent
0	0.000	2095	2.10%
1	8.220	133	0.13%
2	11.500	127	0.13%
3	11.320	126	0.13%
4	7.350	121	0.12%
...
4370	26.380	1	0.00%
4371	-1.220	1	0.00%
4372	30.450	1	0.00%
4373	31.940	1	0.00%
4374	21.170	1	0.00%

4375 rows × 3 columns

In [127...]

```
# Check missing values and dtype
print('Remaining missing values in Train:', train['Changed_Credit_Limit'].isna().sum())
print('Remaining missing values in Test:', test['Changed_Credit_Limit'].isna().sum())
print('dtype: ', train['Changed_Credit_Limit'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Changed_Credit_Limit')
```

Remaining missing values in Train: 0

Remaining missing values in Test: 0

dtype: float64

Out[127...]: {' ', ' -', '.'}

```
In [128...]: # Print the List of unique negative values
print('List of unique negative values:', train[train['Changed_Credit_Limit'] < 0]['Changed_Credit_Limit'].unique())
# Print the count of rows with negative values
print('Number of unique negative values:', len(train[train['Changed_Credit_Limit'] < 0]['Changed_Credit_Limit'].unique()))

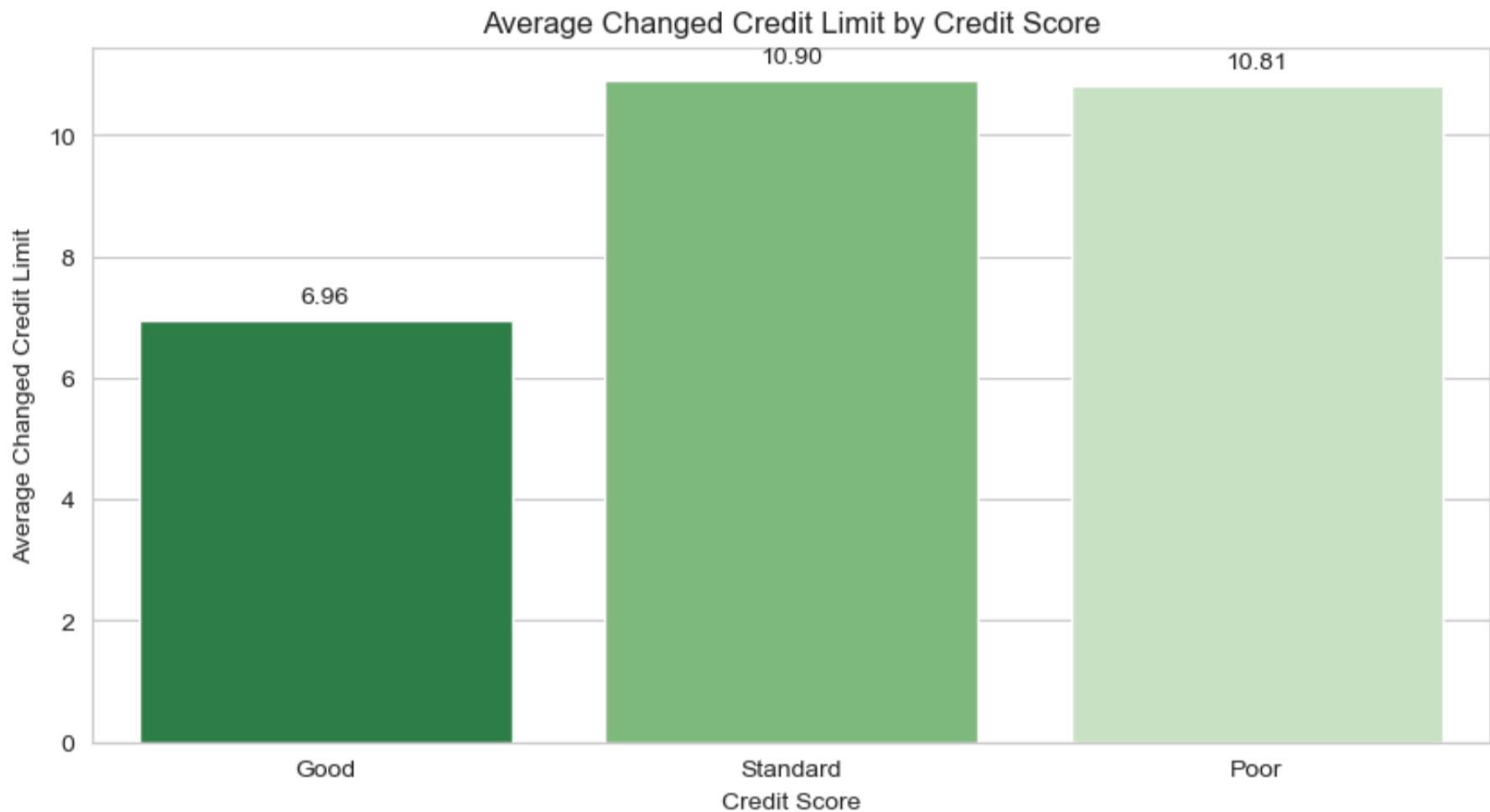
List of unique negative values: [-2.01 -1.01 -3.01 -1.24 -4.14 -0.14 -2.87 -2.46 -4.26 -1.26]
Number of unique negative values: 1586

In [129...]: # Barplot showing the average Changed_Credit_Limit by Credit Score category
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Credit_Score', y='Changed_Credit_Limit', data=train, ci=None, palette='Greens_r')

# Add values on top of the bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 9), textcoords='offset points')

plt.title('Average Changed Credit Limit by Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Average Changed Credit Limit')

plt.show()
```



- The graph shows that customers with **Standard** and **Poor** credit scores have experienced similar and higher average changes to their credit limit (around 10.9 and 10.8), while those with a **Good** credit score have a lower average change (6.96).

This could indicate that credit limit changes are more common or significant for customers with lower credit scores.

'Credit_Mix'

The **Credit_Mix** column represents the variety of credit types a customer uses, such as credit cards, loans, and mortgages. A balanced credit mix can positively affect a customer's credit score by demonstrating their ability to manage different types of credit.

- Values marked with `_` likely indicate missing or unknown data. To address this, the `_` values were replaced with 'Unknown', signaling that the information was not provided.

This approach avoids making assumptions about the customer's credit mix while keeping the dataset complete and ready for analysis without introducing missing values.

In [130...]

```
# Check the column
get_value_count(train, 'Credit_Mix')
```

Out[130...]

	Credit_Mix	counts	percent
0	Standard	36479	36.48%
1	Good	24337	24.34%
2	_	20195	20.20%
3	Bad	18989	18.99%

In [131...]

```
# Check missing values and dtype
print('Remaining missing values in Train:', train['Credit_Mix'].isna().sum())
print('Remaining missing values in Test:', test['Credit_Mix'].isna().sum())
print('dtype: ', train['Credit_Mix'].dtypes)
```

Remaining missing values in Train: 0

Remaining missing values in Test: 0

dtype: object

In [132...]

```
# Replace '_' with 'Unknown'
train['Credit_Mix'].replace('_', 'Unknown', inplace=True) #train
test['Credit_Mix'].replace('_', 'Unknown', inplace=True) #test

# Check the column
get_value_count(train, 'Credit_Mix')
```

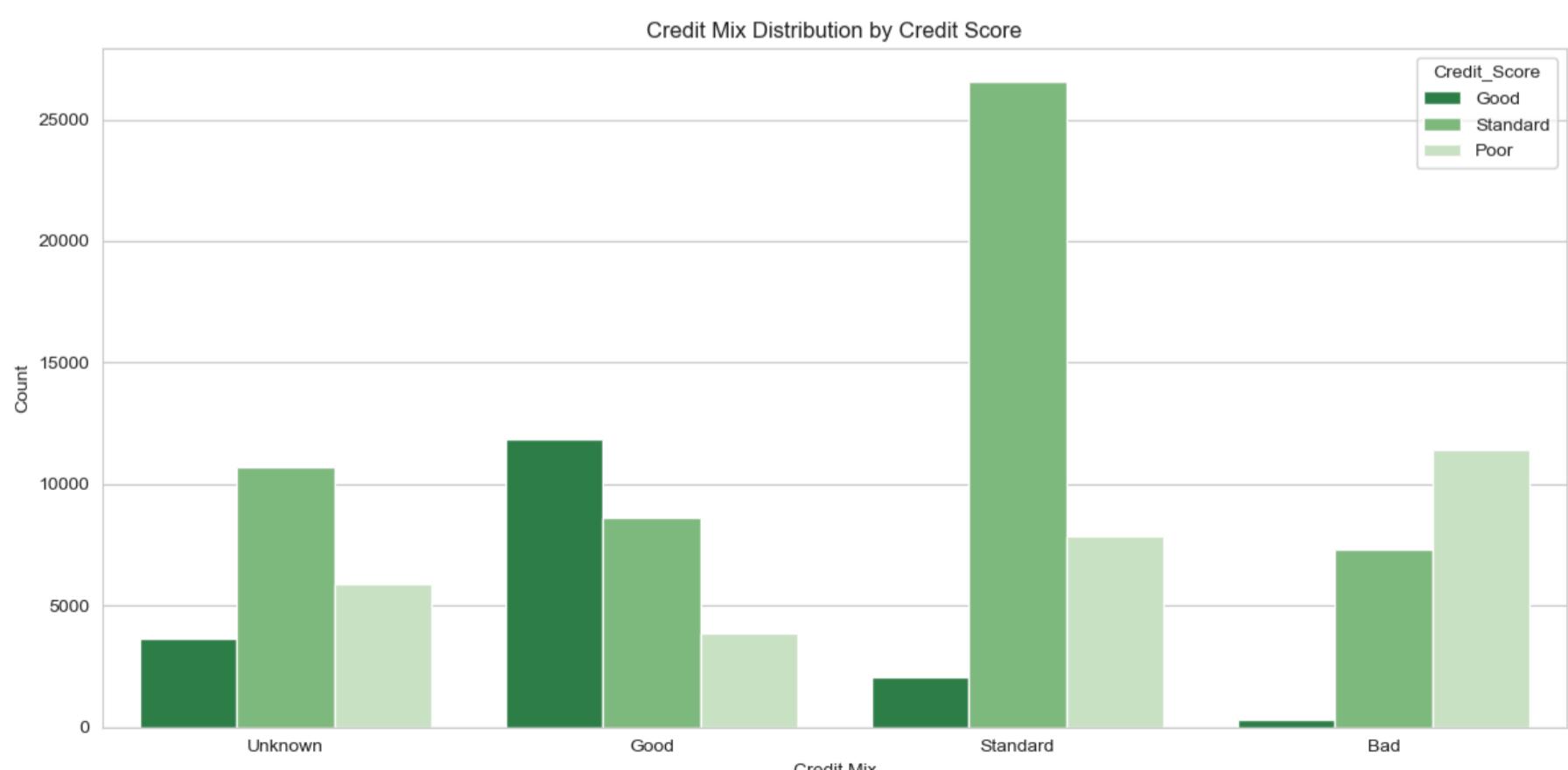
Out[132...]

	Credit_Mix	counts	percent
0	Standard	36479	36.48%
1	Good	24337	24.34%
2	Unknown	20195	20.20%
3	Bad	18989	18.99%

In [133...]

```
# Barplot showing the distribution of Credit_Mix by Credit Score category
plt.figure(figsize=(12, 6))
ax = sns.countplot(x='Credit_Mix', hue='Credit_Score', data=train, palette='Greens_r')

# Add Labels and title
plt.title('Credit Mix Distribution by Credit Score')
plt.xlabel('Credit Mix')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```



- **Good Credit Mix** is strongly associated with **Good Credit Scores**,

- **Bad Credit Mix** is more common among those with **Poor Credit Scores**.
- **Standard Credit Mix** is most frequent in both **Standard** and **Poor Credit Scores**, showing a balanced but less optimized credit type.
- **Unknown Credit Mix** has no strong correlation with any specific credit score.

'Outstanding_Debt'

The **Outstanding_Debt** column reflects the total unpaid debt that a customer owes, including balances from loans, credit cards, and other credit lines. It provides a clear view of the customer's current financial obligations, which is key for assessing credit risk and determining the customer's ability to manage more debt.

- Upon reviewing the data, there were no missing values, but `_` characters were found in place of numeric values.
- These characters were cleaned, and the data type was converted to **numeric** to ensure that the column is properly formatted for analysis and modeling.

This ensures that the debt values are correctly handled and can be used in financial calculations.

In [134...]

```
# Check the column
get_value_count(train, 'Outstanding_Debt')
```

Out[134...]

	Outstanding_Debt	counts	percent
0	1360.45	24	0.02%
1	460.46	23	0.02%
2	1151.7	23	0.02%
3	1109.03	23	0.02%
4	467.7	16	0.02%
...
13173	245.46_	1	0.00%
13174	645.77_	1	0.00%
13175	174.79_	1	0.00%
13176	1181.13_	1	0.00%
13177	1013.53_	1	0.00%

13178 rows × 3 columns

In [135...]

```
# Check missing values and dtype
print('Remaining missing values in Train:', train['Outstanding_Debt'].isna().sum())
print('Remaining missing values in Test:', test['Outstanding_Debt'].isna().sum())
print('dtype: ', train['Outstanding_Debt'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Outstanding_Debt')
```

Remaining missing values in Train: 0
 Remaining missing values in Test: 0
 dtype: object

Out[135...]

```
{' ', '.', '_ '}
```

In [136...]

```
#TRAIN: Cleaning
# Remove non-numeric characters (keeping only digits and decimals)
train['Outstanding_Debt'] = train['Outstanding_Debt'].apply(lambda x: re.sub(r'^0-9.-]+', '', str(x)))

# Convert dtype to float and coerce errors to NaN (invalid strings to NaN)
train['Outstanding_Debt'] = pd.to_numeric(train['Outstanding_Debt'], errors='coerce')
```

In [137...]

```
#TEST: Cleaning
# Remove non-numeric characters (keeping only digits and decimals)
test['Outstanding_Debt'] = test['Outstanding_Debt'].apply(lambda x: re.sub(r'^0-9.-]+', '', str(x)))
```

```
# Convert dtype to float and coerce errors to NaN (invalid strings to NaN)
test['Outstanding_Debt'] = pd.to_numeric(test['Outstanding_Debt'], errors='coerce')
```

In [138...]

```
# Check the list of unique negative values
print('List of unique negative values:', train[train['Outstanding_Debt'] < 0]['Outstanding_Debt'].unique())
# Print the count of rows with negative values
print('Number of unique negative values:', train[train['Outstanding_Debt'] < 0]['Outstanding_Debt'].count())
```

List of unique negative values: []
Number of unique negative values: 0

In [139...]

```
# Check missing values and dtype
print('Remaining missing values in Train:', train['Outstanding_Debt'].isna().sum())
print('Remaining missing values in Test:', test['Outstanding_Debt'].isna().sum())
print('dtype: ', train['Outstanding_Debt'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Outstanding_Debt')
```

Remaining missing values in Train: 0
Remaining missing values in Test: 0
dtype: float64

Out[139...]

```
{' ', '.'}
```

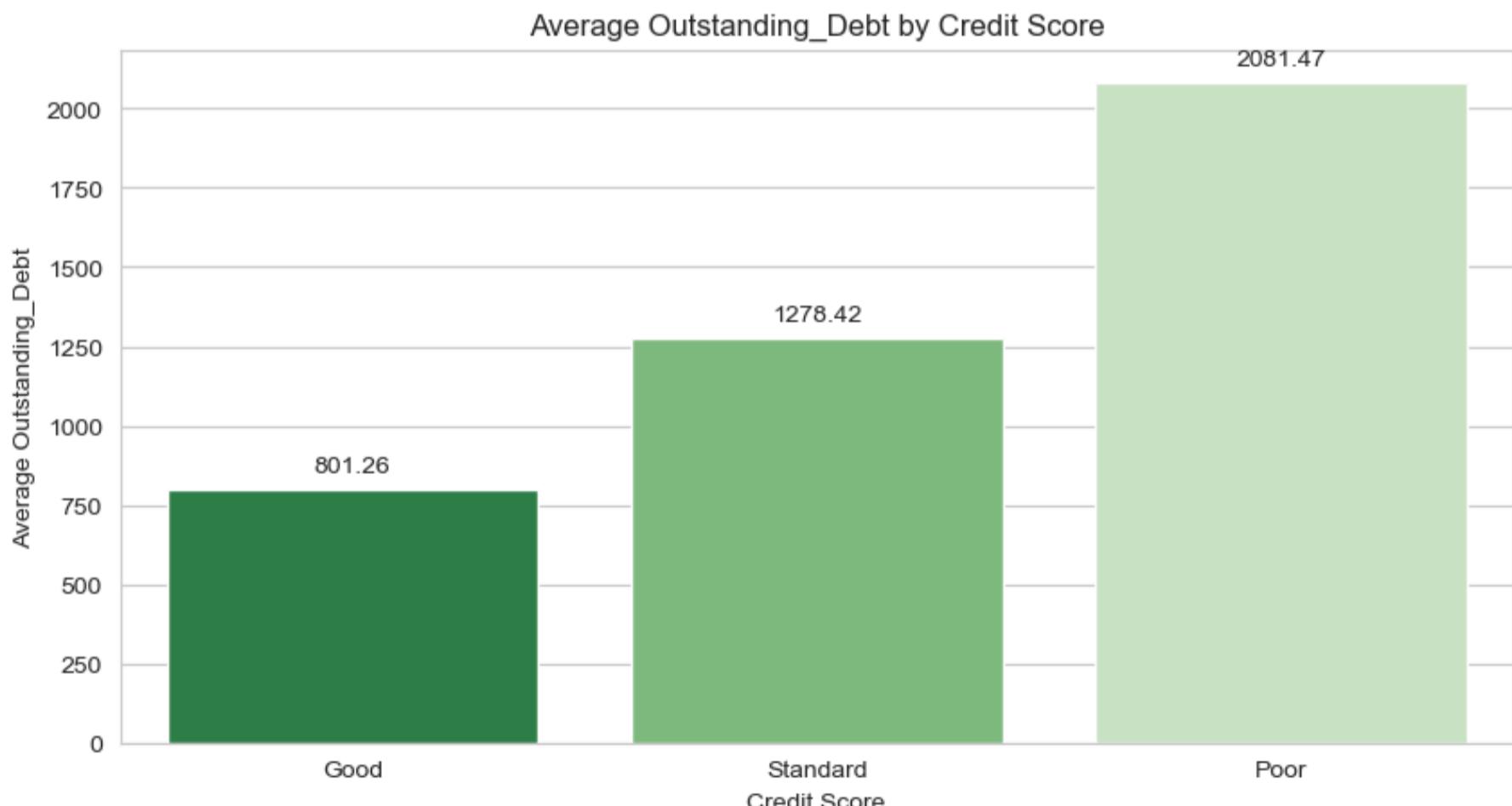
In [140...]

```
# Barplot showing the average Outstanding_Debt by Credit Score category
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Credit_Score', y='Outstanding_Debt', data=train, ci=None, palette='Greens_r')

# Add values on top of the bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 9), textcoords='offset points')

plt.title('Average Outstanding_Debt by Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Average Outstanding_Debt')

plt.show()
```



- Customers with **poor credit scores** have much higher **outstanding debt** compared to those with **standard** and **good credit scores**.
- This suggests a clear relationship between higher outstanding debt and poorer credit scores, indicating that managing debt is a challenge for those with lower credit ratings.
- In contrast, customers with **good credit scores** have much lower outstanding debt, reflecting better financial responsibility.

'Credit_Utilization_Ratio'

The **Credit_Utilization_Ratio** column represents the ratio of a customer's current outstanding debt to their total available credit.

It measures how much of the customer's available credit is being used, typically expressed as a percentage.

No missing or unusual values were found in this column.

Importance:

- **Higher Ratios:** A high credit utilization ratio (e.g., above 30%) can indicate that a customer is heavily reliant on credit and may have difficulty repaying future debts, which can negatively impact their credit score.
- **Lower Ratios:** A low credit utilization ratio suggests that the customer is managing their credit responsibly by not maxing out available credit limits, which can be a positive indicator for credit scoring.

In [141...]

```
# Check the column
get_value_count(train, 'Credit_Utilization_Ratio')
```

Out[141...]

	Credit_Utilization_Ratio	counts	percent
0	26.823	1	0.00%
1	28.328	1	0.00%
2	30.017	1	0.00%
3	25.479	1	0.00%
4	33.934	1	0.00%
...
99995	30.687	1	0.00%
99996	38.730	1	0.00%
99997	30.018	1	0.00%
99998	27.280	1	0.00%
99999	34.192	1	0.00%

100000 rows × 3 columns

In [142...]

```
# Check missing values and dtype
print('Remaining missing values in Train:', train['Credit_Utilization_Ratio'].isna().sum())
print('Remaining missing values in Test:', test['Credit_Utilization_Ratio'].isna().sum())
print('dtype: ', train['Credit_Utilization_Ratio'].dtypes)

# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Credit_Utilization_Ratio')
```

Remaining missing values in Train: 0

Remaining missing values in Test: 0

dtype: float64

Out[142...]

```
{' ', '.'}
```

In [143...]

```
# Check the List of unique negative values
print('List of unique negative values:', train[train['Credit_Utilization_Ratio'] < 0]['Credit_Utilization_Ratio'])
# Print the count of rows with negative values
print('Number of unique negative values:', train[train['Credit_Utilization_Ratio'] < 0]['Credit_Utilization_Ratio'])
```

List of unique negative values: []

Number of unique negative values: 0

In [144...]

```
# Barplot showing the average Credit Utilization Ratio by Credit Score category
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Credit_Score', y='Credit_Utilization_Ratio', data=train, ci=None, palette='Greens_r')

# Add values on top of the bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
```

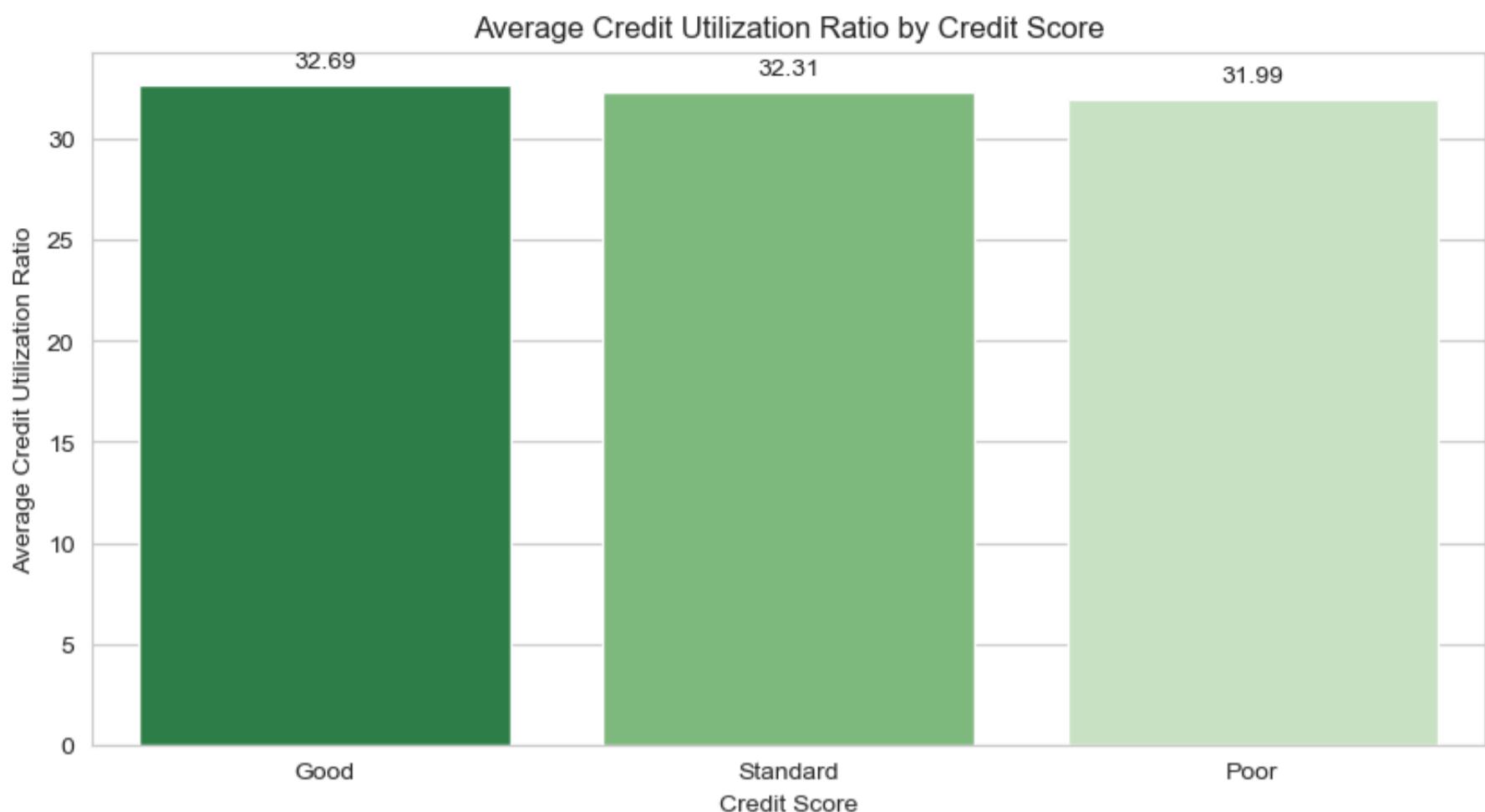
```

xytext=(0, 9), textcoords='offset points')

plt.title('Average Credit Utilization Ratio by Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Average Credit Utilization Ratio')

plt.show()

```



The **Credit_Utilization_Ratio** represents the percentage of a customer's available credit that is being used. A higher ratio indicates more reliance on credit, while a lower ratio suggests responsible credit usage.

- The average **Credit Utilization Ratio** is quite similar across all credit score categories, with **Good Credit Score** having a slightly higher ratio (32.69%) compared to **Standard** (32.31%) and **Poor** (31.99%).
- This indicates that credit utilization does not significantly vary between these groups.

'Payment_Behaviour'

The **Payment_Behaviour** column describes the customer's spending and payment patterns. It indicates whether a customer spends and makes payments in small, medium, or large amounts. This feature helps in understanding how a customer manages their finances in terms of spending and repayments.

In [145...]

```
# Check the column
get_value_count(train, 'Payment_Behaviour')
```

Out[145...]

	Payment_Behaviour	counts	percent
0	Low_spent_Small_value_payments	25513	25.51%
1	High_spent_Medium_value_payments	17540	17.54%
2	Low_spent_Medium_value_payments	13861	13.86%
3	High_spent_Large_value_payments	13721	13.72%
4	High_spent_Small_value_payments	11340	11.34%
5	Low_spent_Large_value_payments	10425	10.42%
6	!@9#%8	7600	7.60%

In [146...]

```
# Check missing values and dtype
print('Remaining missing values in Train:', train['Payment_Behaviour'].isna().sum())
print('Remaining missing values in Test:', test['Payment_Behaviour'].isna().sum())
print('dtype: ', train['Payment_Behaviour'].dtypes)
```

```
Remaining missing values in Train: 0
Remaining missing values in Test: 0
dtype: object
```

```
In [147...]: # Replace unusual values in 'Payment_Behaviour' with NaN
train['Payment_Behaviour'].replace('!@9#%8', np.nan, inplace=True)
test['Payment_Behaviour'].replace('!@9#%8', np.nan, inplace=True)
```

Filling the NaN Values after replacing !@9#%8 with NaN:

For the `Payment_Behaviour` column, the unusual value `!@9#%8` was replaced with `NaN` to handle invalid data. To accurately fill the missing values, we grouped the data using key financial features for `Payment_Behaviour`:

- `Credit_Score`: Indicates the customer's overall financial responsibility, closely tied to payment habits.
- `Outstanding_Debt`: Shows how much debt a customer has, affecting their ability to make payments.
- `Num_of_Loan`: Reflects the customer's financial commitments and payment patterns.
- `Annual_Income`: A customer's income directly influences their ability to meet payments.

These columns were selected because they provide a clear financial profile of the customer. By using `mode` to fill missing values within each group, we ensure the imputation is data-driven and reflects the most common behavior in similar financial situations. This method maintains the integrity of the data and accurately represents customer payment behavior.

```
In [148...]: # Check missing values after replacing !@9#%8 with NaN
print('Remaining missing values in Train:', train['Payment_Behaviour'].isna().sum())
print('Remaining missing values in Test:', test['Payment_Behaviour'].isna().sum())
```

```
Remaining missing values in Train: 7600
Remaining missing values in Test: 3800
```

```
In [149...]: # Handle missing values using groupby and mode, with a fallback to 'Unknown'
train['Payment_Behaviour'] = train.groupby(['Credit_Score', 'Outstanding_Debt', 'Num_of_Loan', 'Annual_Income'],
                                         lambda x: x.fillna(x.mode()[0] if not x.mode().empty else 'Unknown'))

test['Payment_Behaviour'] = test.groupby(['Outstanding_Debt', 'Num_of_Loan', 'Annual_Income'])['Payment_Behaviour']
                                         lambda x: x.fillna(x.mode()[0] if not x.mode().empty else 'Unknown'))
```

```
In [150...]: # Check missing values and dtype
print('Remaining missing values in Train:', train['Payment_Behaviour'].isna().sum())
print('Remaining missing values in Test:', test['Payment_Behaviour'].isna().sum())
```

```
Remaining missing values in Train: 0
Remaining missing values in Test: 0
```

```
In [151...]: get_value_count(train, 'Payment_Behaviour')
```

	Payment_Behaviour	counts	percent
0	Low_spent_Small_value_payments	27350	27.35%
1	High_spent_Medium_value_payments	19195	19.20%
2	High_spent_Large_value_payments	15336	15.34%
3	Low_spent_Medium_value_payments	14600	14.60%
4	High_spent_Small_value_payments	11991	11.99%
5	Low_spent_Large_value_payments	10952	10.95%
6	Unknown	576	0.58%

```
In [152...]: # Drop rows where 'Payment_Behaviour' column has 'Unknown' values ---> only 576 data points (0.5% of total)
train = train[train['Payment_Behaviour'] != 'Unknown']
test = test[test['Payment_Behaviour'] != 'Unknown']
```

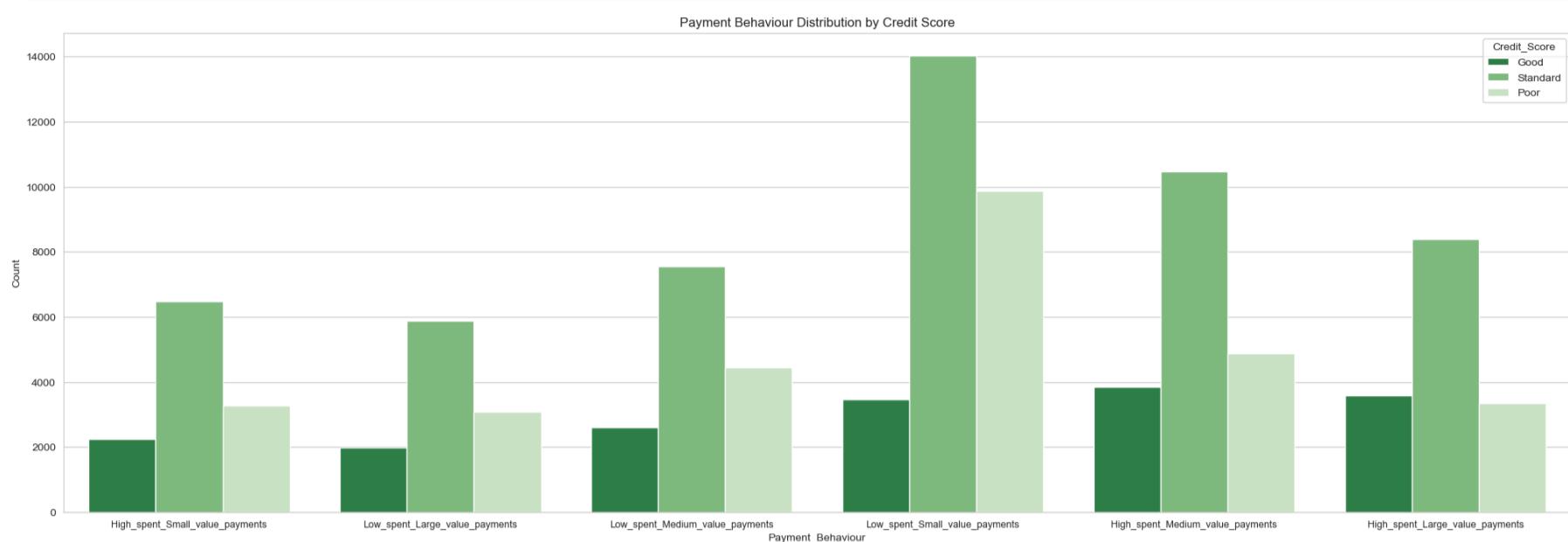
```
In [153...]: # Plotting the barplot of Payment_Behaviour by Credit_Score (target feature)
plt.figure(figsize=(25, 8))

# Since Payment_Behaviour is categorical, we use countplot instead of barplot for more accurate visualization
ax = sns.countplot(x='Payment_Behaviour', hue='Credit_Score', data=train, palette='Greens_r')

# Add title and labels
```

```
plt.title('Payment Behaviour Distribution by Credit Score')
plt.xlabel('Payment_Behaviour')
plt.ylabel('Count')
plt.xticks(fontsize=9)

# Display the plot
plt.show()
```



- **Standard credit score** customers tend to have a higher proportion of both **low-spent small value payments** and **high-spent large value payments** compared to other credit score categories.
- **Poor credit score** customers exhibit more **low-spent small value payments**, indicating potentially more frequent small transactions, but still reflecting lower overall financial responsibility.
- **Good credit score** customers generally show fewer **high-spent large value payments**, implying more disciplined or stable financial behavior.

This pattern highlights the relationship between **payment behavior** and **credit score**, where higher credit scores are associated with more responsible and consistent payment behavior.

This feature is crucial for understanding customer financial habits and predicting their creditworthiness, making it an important factor in credit risk models.

'Payment_of_Min_Amount'

The **Payment_of_Min_Amount** column shows whether a customer has paid the minimum required amount on their credit or loan payments. It contains the following values:

- **Yes** : The customer paid the minimum amount.
- **No** : The customer did not pay the minimum amount.
- **NM** : This likely indicates missing or unreported data regarding whether the customer paid the minimum amount.

In [154...]

```
# Check the column
get_value_count(train, 'Payment_of_Min_Amount')
```

Out[154...]

	Payment_of_Min_Amount	counts	percent
0	Yes	52024	52.33%
1	No	35466	35.67%
2	NM	11934	12.00%

In [155...]

```
# Check missing values and dtype
print('Remaining missing values in Train:', train['Payment_of_Min_Amount'].isna().sum())
print('Remaining missing values in Test:', test['Payment_of_Min_Amount'].isna().sum())
print('dtype: ', train['Payment_of_Min_Amount'].dtypes)
```

Remaining missing values in Train: 0
 Remaining missing values in Test: 0
 dtype: object

In [156...]

```
# Replace unusual values in 'Payment_Behaviour' with NaN
train['Payment_of_Min_Amount'].replace('NM', np.nan, inplace=True)
test['Payment_of_Min_Amount'].replace('NM', np.nan, inplace=True)
```

Filling the NaN Values after replacing NM with NaN:

The `Payment_of_Min_Amount` column indicates whether a customer has made the minimum required payment, with values like '`Yes`', '`No`', and '`NM`' (likely representing missing or unknown information).

- To handle this, the unusual '`NM`' values were replaced with `NaN` to signify missing data.
- To fill these missing values, we used the `mode` (most frequent value) based on the customer's `Credit_Score`, `Annual_Income`, `Outstanding_Debt`, and `Payment_Behaviour`.
- These features were selected because they directly relate to a customer's financial status and payment habits, ensuring that the imputed values are based on relevant financial characteristics and provide more accurate predictions.

In [157...]

```
# Check missing values after replacing !@#%8 with NaN
print('Remaining missing values in Train:', train['Payment_of_Min_Amount'].isna().sum())
print('Remaining missing values in Test:', test['Payment_of_Min_Amount'].isna().sum())
```

Remaining missing values in Train: 11934

Remaining missing values in Test: 5966

In [158...]

```
# Display the first mode of 'Payment_of_Min_Amount' for each group of 'Credit_Score' and 'Payment_Behaviour'
train.groupby(['Credit_Score', 'Outstanding_Debt', 'Num_of_Loan', 'Payment_Behaviour'])['Payment_of_Min_Amount']
```

Out[158...]

Credit_Score	Outstanding_Debt	Num_of_Loan	Payment_Behaviour	
Good	0.230	3	High_spent_Large_value_payments High_spent_Medium_value_payments Low_spent_Large_value_payments Low_spent_Medium_value_payments Low_spent_Small_value_payments	No No No No No
Standard	4997.050 4997.100 4998.070	5 6 8	Low_spent_Medium_value_payments Low_spent_Large_value_payments Low_spent_Small_value_payments High_spent_Medium_value_payments High_spent_Small_value_payments	Yes Yes Yes Yes Yes
				...

Name: Payment_of_Min_Amount, Length: 59785, dtype: object

In [159...]

```
# Filling missing values using groupby and mode, with a fallback to 'Unknown'
train['Payment_of_Min_Amount'] = train.groupby(['Credit_Score', 'Outstanding_Debt', 'Num_of_Loan', 'Payment_Behaviour'])['Payment_of_Min_Amount'].apply(lambda x: x.fillna(x.mode()[0] if not x.mode().empty else 'Unknown'))

test['Payment_of_Min_Amount'] = test.groupby(['Outstanding_Debt', 'Num_of_Loan', 'Payment_Behaviour'])['Payment_of_Min_Amount'].apply(lambda x: x.fillna(x.mode()[0] if not x.mode().empty else 'Unknown'))
```

In [160...]

```
# Check missing values and dtype
print('Remaining missing values in Train:', train['Payment_of_Min_Amount'].isna().sum())
print('Remaining missing values in Test:', test['Payment_of_Min_Amount'].isna().sum())
```

Remaining missing values in Train: 0

Remaining missing values in Test: 0

In [161...]

```
get_value_count(train, 'Payment_of_Min_Amount')
```

Out[161...]

Payment_of_Min_Amount	counts	percent
0	Yes	56272 56.60%
1	No	38241 38.46%
2	Unknown	4911 4.94%

In [162...]

```
# Drop rows where 'Payment_of_Min_Amount' column has 'Unknown' values--> only 4% of total data
train = train[train['Payment_of_Min_Amount'] != 'Unknown']
test = test[test['Payment_of_Min_Amount'] != 'Unknown']
```

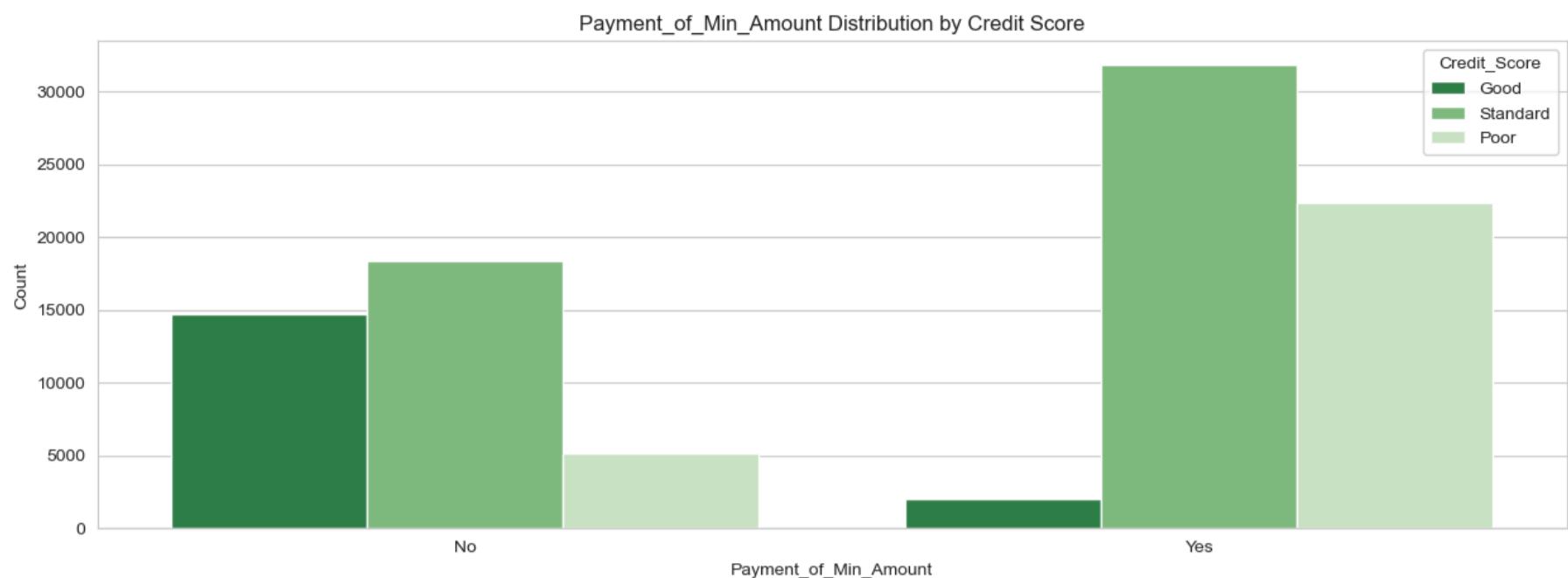
In [163...]

```
# Plotting the barplot of Payment_of_Min_Amount by Credit_Score (target feature)
plt.figure(figsize=(15, 5))

# Since Payment_Behaviour is categorical, we use countplot instead of barplot for more accurate visualization
ax = sns.countplot(x='Payment_of_Min_Amount', hue='Credit_Score', data=train, palette='Greens_r')

# Add title and labels
plt.title('Payment_of_Min_Amount Distribution by Credit Score')
```

```
plt.xlabel('Payment_of_Min_Amount')
plt.ylabel('Count')
plt.show()
```



- Customers with **standard** and **poor credit scores** are more likely to make the minimum payment ('**Yes**'), while those with **good credit scores** are more evenly split between making and not making the minimum payment.
- The '**Unknown**' category is minimal across all credit scores, indicating most customers have clear payment behaviors.

This suggests that making the minimum payment is more common among customers with lower credit scores.

'Total_EMI_per_month'

The `Total_EMI_per_month` column shows how much a customer pays in monthly loan installments, including both principal and interest.

- It's important because it reflects the customer's financial commitment to loans.
- Higher EMIs mean more debt, which could impact their ability to take on new loans or manage other expenses.
- There are no missing or unusual values in this column, so it's ready for analysis.

In [164...]

```
# Check the column
get_value_count(train, 'Total_EMI_per_month')
```

Out[164...]

	Total_EMI_per_month	counts	percent
0	0.000	10052	10.64%
1	49.575	8	0.01%
2	38.072	8	0.01%
3	6.525	8	0.01%
4	21.581	8	0.01%
...
14785	51384.000	1	0.00%
14786	64946.000	1	0.00%
14787	50947.000	1	0.00%
14788	37303.000	1	0.00%
14789	58638.000	1	0.00%

14790 rows × 3 columns

In [165...]

```
# Check missing values and dtype
print('Remaining missing values in Train:', train['Total_EMI_per_month'].isna().sum())
print('Remaining missing values in Test:', test['Total_EMI_per_month'].isna().sum())
print('dtype: ', train['Total_EMI_per_month'].dtypes)
```

```
# Check the unusual-non-numeric values
find_non_numeric_values(train, 'Total_EMI_per_month')

Remaining missing values in Train: 0
Remaining missing values in Test: 0
dtype: float64

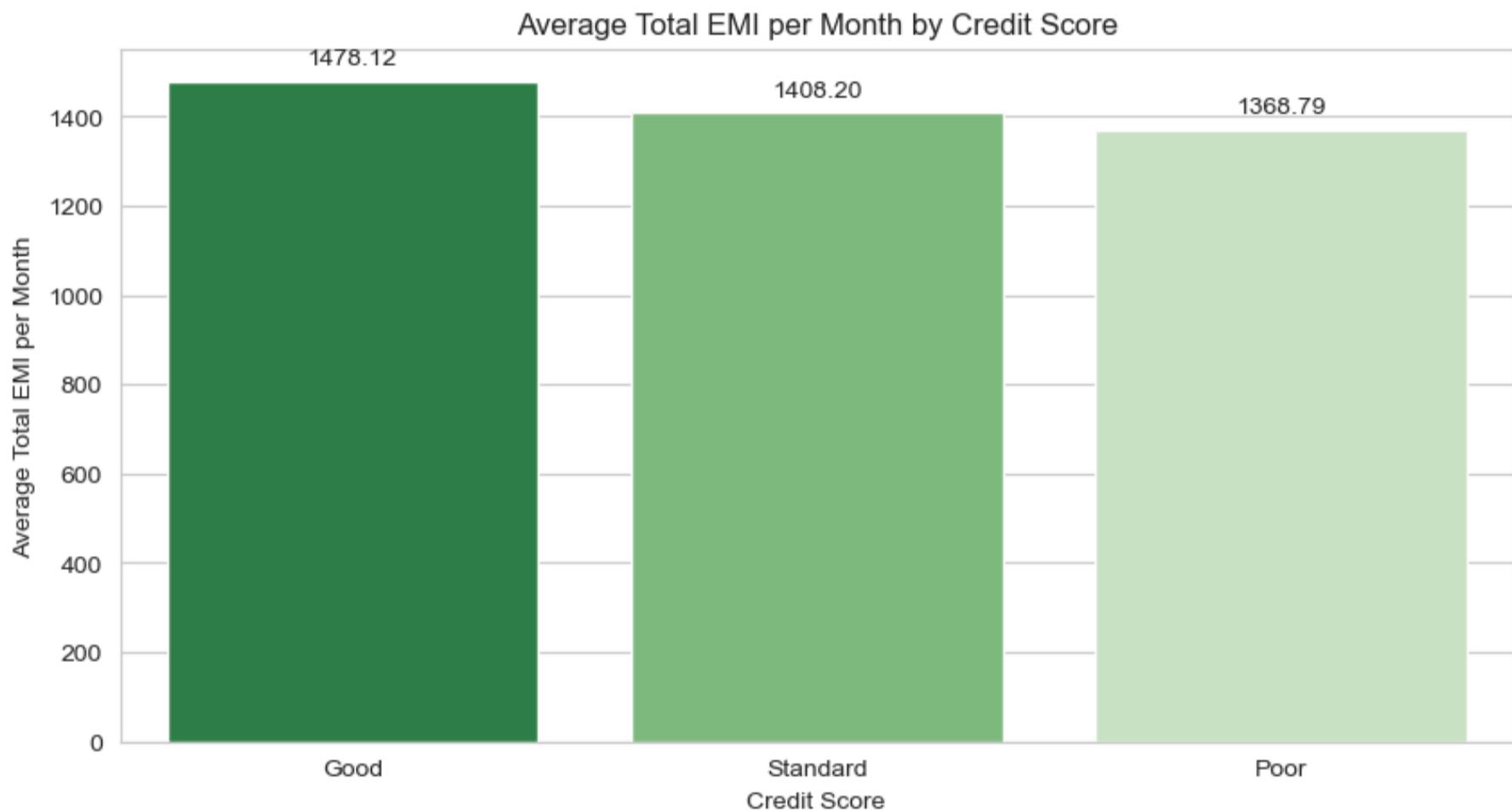
Out[165... {' ', '.'}

In [166... # Barplot showing the average Total EMI per Month by Credit Score category
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Credit_Score', y='Total_EMI_per_month', data=train, ci=None, palette='Greens_r')

# Add values on top of the bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 9), textcoords='offset points')

plt.title('Average Total EMI per Month by Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Average Total EMI per Month')

plt.show()
```



- Customers with a **good credit score** tend to have slightly higher **Total EMI per month** compared to those with **standard** and **poor credit scores**.
- This suggests that higher credit score customers are likely managing larger loans or have more financial commitments, but they handle their debt better, leading to a higher credit score.

Target: 'Credit_Score'

The **Credit_Score** feature categorizes customers into **Good**, **Standard**, and **Poor** based on their creditworthiness. In real-world finance, this helps banks assess risk, approve loans, and set interest rates. Higher credit scores indicate lower risk, leading to better loan terms, while lower scores suggest higher risk.

For the **dataset** and **ANN model**, the **Credit_Score** is the target variable the model aims to predict. By analyzing features like **income**, **debt**, and **payment behavior**, the model learns to classify customers into these categories. This allows for better risk assessment and decision-making, helping financial institutions make informed credit-related decisions.

```
In [167... # Check the column
get_value_count(train, 'Credit_Score')
```

Out[167...]

	Credit_Score	counts	percent
0	Standard	50262	53.18%
1	Poor	27539	29.14%
2	Good	16712	17.68%

In [168...]

```
# Check missing values and dtype
print('Remaining missing values in Train:', train['Credit_Score'].isna().sum())
print('dtype: ', train['Credit_Score'].dtypes)
```

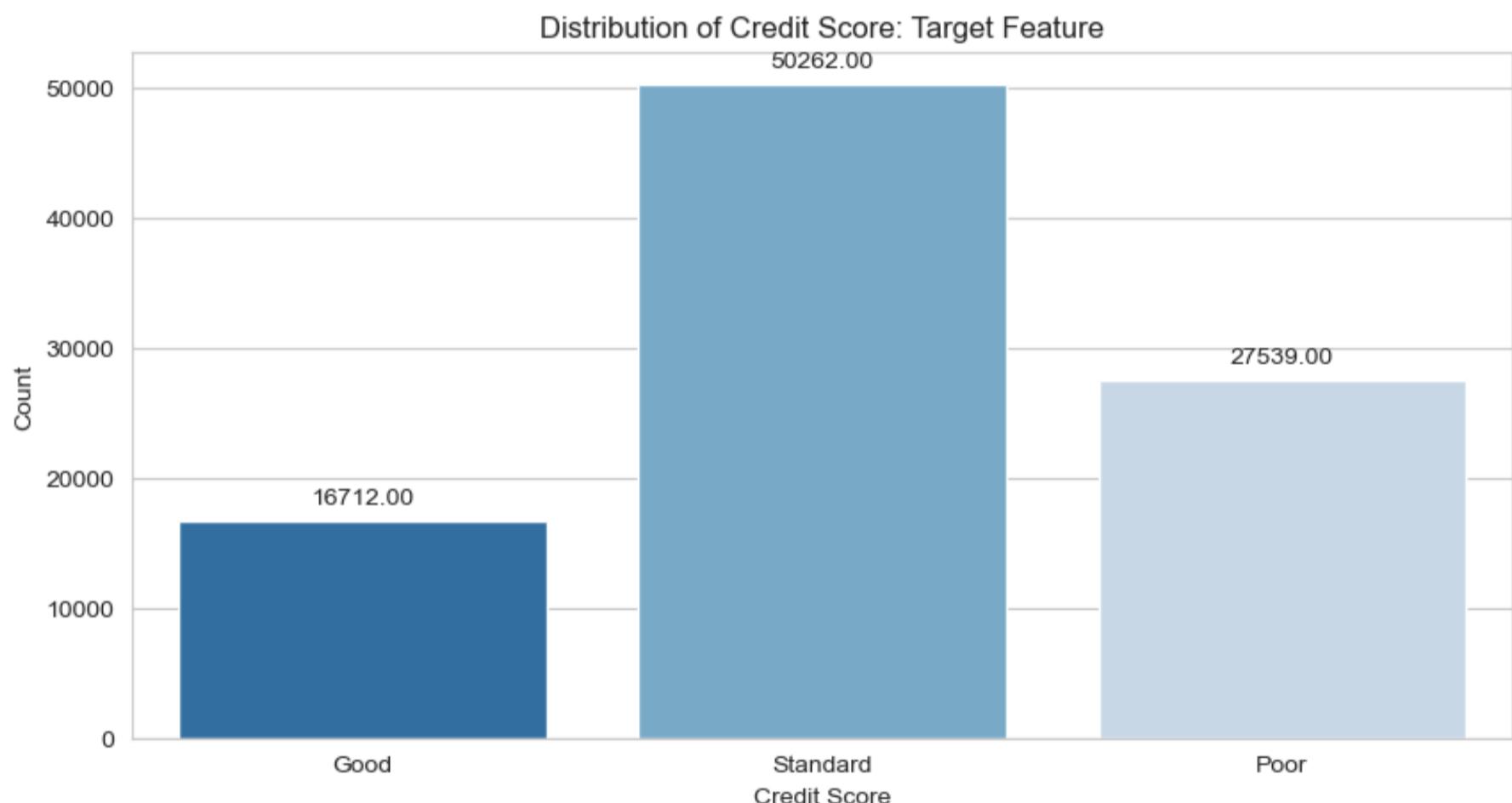
Remaining missing values in Train: 0
dtype: object

In [169...]

```
# Distribution of target feature: Credit_Score
plt.figure(figsize=(10, 5))
ax = sns.countplot(x='Credit_Score', data=train, palette='Blues_r')

# Add values on top of the bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 9), textcoords='offset points')

plt.title('Distribution of Credit Score: Target Feature')
plt.xlabel('Credit Score')
plt.ylabel('Count')
plt.show()
```



- Majority of customers fall into the **Standard** credit score category, with **53,174** records.
- The **Poor** credit score group is the next largest, with **28,998** records, while the **Good** credit score category has the smallest count at **17,828**.
- This distribution indicates that most customers have an average credit score, and there are fewer customers with excellent credit scores in this dataset.

Imbalanced Data Distribution:

The **Credit_Score** target feature is imbalanced, with many more examples in the **Standard** category than in the **Good** and **Poor** categories.

This imbalance could lead the model to focus too much on the majority class and make less accurate predictions for the minority classes.

- To deal with this, techniques like **SMOTE** can be used to generate more samples for the smaller classes, balancing the data.

- Another option is to use **Stratified K-Fold Cross-Validation**, which ensures that each fold in the training process has the same proportion of each class.

These methods will help the model learn better from all classes, leading to more accurate predictions.

Dropping Unnecessary Features

- ID ,
- Customer_ID ,
- Name ,
- SSN
- Month

The **ID**, **Customer_ID**, **Name**, and **SSN** columns are not useful for classification tasks and can add unnecessary complexity to the model. These identification features don't provide any patterns or information for the **ANN** to learn from and may even lead to overfitting, where the model memorizes unique values rather than learning useful patterns.

Similarly, the **Month** column, where all values are identical, doesn't offer any variation or predictive power for the model. Since it doesn't reflect any meaningful behavior, it won't help the ANN improve its predictions.

```
In [170...]: # List of columns to drop
columns_to_drop = ['ID', 'Customer_ID', 'Name', 'SSN', 'Month']

# Dropping the columns in both train and test datasets
train.drop(columns=columns_to_drop, inplace=True)
test.drop(columns=columns_to_drop, inplace=True)

print('Dropped Features: 1.ID, 2.Customer_ID, 3.Name, 4.SSN, 5.Month.')
```

Dropped Features: 1.ID, 2.Customer_ID, 3.Name, 4.SSN, 5.Month.

Outliers

```
In [171...]: train.describe(include='number').T
```

Out[171...]

		count	mean	std	min	25%	50%	75%	max
	Age	94513.000	34.343	9.786	18.000	26.000	34.000	42.000	56.00
	Annual_Income	94513.000	168536.136	1386456.264	7005.930	19379.710	37353.580	72572.460	24198062.00
	Monthly_Inhand_Salary	94513.000	5664.906	44994.056	303.645	1621.118	3085.853	5957.715	1990379.58
	Num_Bank_Accounts	94513.000	17.124	117.652	0.000	3.000	6.000	7.000	1798.00
	Num_Credit_Card	94513.000	22.576	129.431	0.000	4.000	5.000	7.000	1499.00
	Interest_Rate	94513.000	72.293	466.108	1.000	8.000	13.000	20.000	5797.00
	Num_of_Loan	94513.000	9.716	56.183	0.000	2.000	3.000	6.000	1496.00
	Delay_from_due_date	94513.000	21.093	14.807	0.000	10.000	18.000	28.000	67.00
	Num_of_Delayed_Payment	94513.000	31.104	219.451	0.000	9.000	15.000	19.000	4397.00
	Changed_Credit_Limit	94513.000	10.186	6.878	-6.490	4.980	9.270	14.700	36.97
	Num_Credit_Inquiries	94513.000	27.969	192.267	0.000	3.000	6.000	9.000	2597.00
	Outstanding_Debt	94513.000	1425.042	1155.306	0.230	565.340	1163.470	1945.030	4998.07
	Credit_Utilization_Ratio	94513.000	32.291	5.116	20.000	28.061	32.314	36.501	50.00
	Credit_History_Age	94513.000	221.124	95.161	1.000	154.000	221.195	292.000	404.00
	Total_EMI_per_month	94513.000	1409.080	8323.330	0.000	30.174	68.807	160.769	82331.00
	Amount_invested_monthly	94513.000	637.794	2000.177	0.000	77.061	142.572	302.729	10000.00
	Monthly_Balance	94513.000	402.432	212.956	0.008	270.690	337.202	468.732	1602.04
	Credit-Builder Loan	94513.000	0.318	0.466	0.000	0.000	0.000	1.000	1.00
	Personal Loan	94513.000	0.311	0.463	0.000	0.000	0.000	1.000	1.00
	Debt Consolidation Loan	94513.000	0.311	0.463	0.000	0.000	0.000	1.000	1.00
	Student Loan	94513.000	0.310	0.463	0.000	0.000	0.000	1.000	1.00
	Payday Loan	94513.000	0.319	0.466	0.000	0.000	0.000	1.000	1.00
	Mortgage Loan	94513.000	0.314	0.464	0.000	0.000	0.000	1.000	1.00
	Auto Loan	94513.000	0.306	0.461	0.000	0.000	0.000	1.000	1.00
	Home Equity Loan	94513.000	0.314	0.464	0.000	0.000	0.000	1.000	1.00

◀ ▶

In [172...]

train.describe(include='object').T

Out[172...]

	count	unique	top	freq
Occupation	94513	16	Unknown	6657
Credit_Mix	94513	4	Standard	34607
Payment_of_Min_Amount	94513	2	Yes	56272
Payment_Behaviour	94513	6	Low_spent_Small_value_payments	26464
Credit_Score	94513	3	Standard	50262

In [173...]

```
# BOXPLOT
import cufflinks as cf
cf.go_offline()

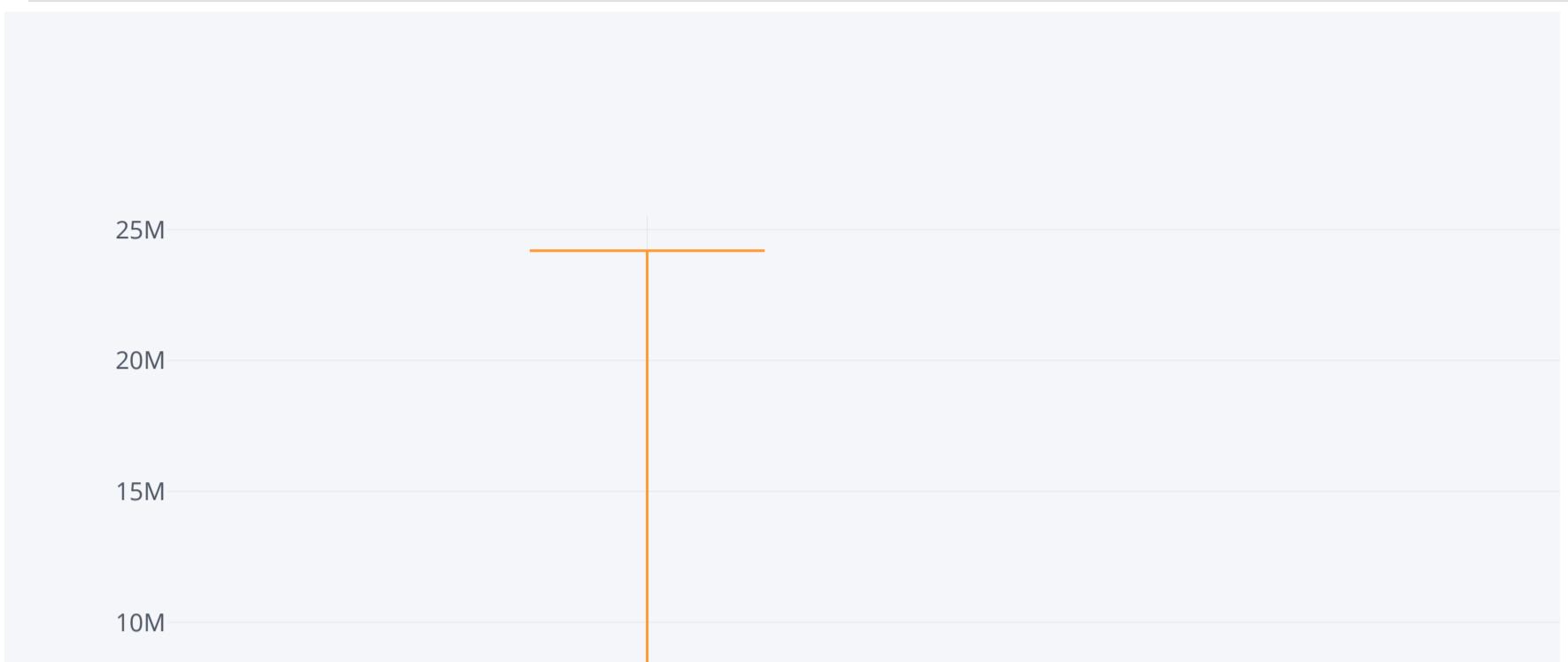
# Selecting only numerical columns
df_num = train.select_dtypes(include='number')

# Calculate IQR for each numerical column
iqr_values = df_num.apply(lambda x: x.quantile(0.75) - x.quantile(0.25))

# Set a threshold for IQR to filter columns with a significant range
iqr_threshold = 2000
selected_columns = iqr_values[iqr_values > iqr_threshold].index
```

```
# Filter the DataFrame with the selected columns
df_filtered = df_num[selected_columns]

# Plotting the boxplot for the filtered numerical columns
df_filtered.iplot(kind="box")
```



In [174...]

```
# # Calculate skewness for numeric features
num_cols = df.select_dtypes('number').columns
skew_limit = 1 # define a limit for highly skewed data
skew_cols = df[num_cols].skew().loc[lambda x: (x > skew_limit) | (x < -skew_limit)].sort_values(ascending=False)

skew_cols
```

Out[174...]

	Skew
Num_Bank_Accounts	11.219
Num_Credit_Inquiries	9.718
Interest_Rate	9.123
Num_Credit_Card	8.401
Total_EMI_per_month	7.050
Monthly_Inhand_Salary	1.129

In [175...]

```
# Handling Outliers
# replace Outliers with median

# for col in Numericals :
#     outliers_indecies = detect_outliers(df,0,[col])
#     median = df[col].median()
#     df[col].iloc[outliers_indecies] = median
```

Outliers:

In the dataset, there are visible outliers in features such as **Annual Income** and **Monthly Inhand Salary**.

Given the financial nature of the data, these outliers may represent legitimate variations in customer financial profiles. Therefore, no immediate action will be taken on these outliers.

If necessary, after evaluating the model's performance, adjustments may be considered to further improve results.

In addition to outliers, the dataset also contains features with **high skewness**, such as **Num_Bank_Accounts**, **Num_Credit_Inquiries**, and **Interest_Rate**.

These skewed distributions suggest that the data is not evenly distributed, with many values clustered on one side. While skewness can affect the performance of certain models, it might not necessarily hinder the learning process for an ANN, which is more robust to such irregularities compared to linear models.

For now, skewed features will not be transformed, as handling them might not be crucial for an ANN. However, if skewness is found to impact the model's performance, transformations such as log scaling may be considered.

Save Cleaned DataSets

```
In [176...]: # Save the cleaned training data
train.to_csv('clean_train.csv', index=False)

# Save the cleaned test data
test.to_csv('clean_test.csv', index=False)
```

```
In [177...]: # download Links
from IPython.display import FileLink, display

# Create and display download links for the cleaned datasets
train_file = FileLink('clean_train.csv', result_html_prefix="Click here to download: ")
test_file = FileLink('clean_test.csv', result_html_prefix="Click here to download: ")

display(train_file)
display(test_file)
```

Click here to download: [clean_train.csv](#)

Click here to download: [clean_test.csv](#)

Data Preprocessing

Load the Cleaned Data

```
In [178...]: # Load the cleaned training data
df_train = pd.read_csv('clean_train.csv')

# Load the cleaned test data
df_test = pd.read_csv('clean_test.csv')
```

Encode Categoricals

```
In [179...]: # select columns of type 'object'
df_train.select_dtypes(include=['object']).head()
```

	Occupation	Credit_Mix	Payment_of_Min_Amount	Payment_Behaviour	Credit_Score
0	Scientist	Unknown	No	High_spent_Small_value_payments	Good
1	Scientist	Good	No	Low_spent_Large_value_payments	Good
2	Scientist	Good	No	Low_spent_Medium_value_payments	Good
3	Scientist	Good	No	Low_spent_Small_value_payments	Good
4	Scientist	Good	No	High_spent_Medium_value_payments	Good

Ordinal Encoding

Following columns are suitable for **Ordinal Encoding**:

1. **Credit_Score** :

- Categories like **Good**, **Standard**, and **Poor** have a natural order in terms of creditworthiness, making them ideal for ordinal encoding.

2. Credit_Mix :

- Categories like **Poor**, **Standard**, and **Good** indicate credit risk levels, making it ideal for ordinal encoding.

3. Payment_of_Min_Amount :

- This can be encoded as **No = 0**, **Yes = 1** since paying the minimum amount regularly may indicate different levels of financial behavior.

4. Payment_Behaviour :

- The column represents a clear progression in terms of financial responsibility. Moving from **Low_spent_Small_value_payments** to **High_spent_Large_value_payments** reflects increasing levels of financial commitment and responsible payment behavior. Therefore, this feature should be ordinally encoded, as the order is significant. Customers who make **high-value payments** demonstrate a stronger financial capacity and responsibility compared to those with **low-value payments**, making this column ideal for ordinal encoding.

One-Hot Encoding

Occupation column not inherently ordinal as categories don't have a specific rank or order, so it would be better suited for **One-Hot Encoding**.

Ordinal Encoding the Target Feature: Credit_Score

```
In [180...]: df_train['Credit_Score'].value_counts()
```

```
Out[180...]: Credit_Score
Standard      50262
Poor          27539
Good          16712
Name: count, dtype: int64
```

```
In [181...]: # Define the order of categories for Credit_Score
credit_score_order = [['Poor', 'Standard', 'Good']] # Ordered from lowest to highest

# Create the OrdinalEncoder for Credit_Score
ordinal_encoder = OrdinalEncoder(categories=credit_score_order, dtype=int)

# Apply ordinal encoding only to the 'Credit_Score' column in df_train
df_train['Credit_Score'] = ordinal_encoder.fit_transform(df_train[['Credit_Score']])

# Check the encoded 'Credit_Score' values
df_train['Credit_Score'].value_counts()
```

```
Out[181...]: Credit_Score
1      50262
0      27539
2      16712
Name: count, dtype: int64
```

Train-Test Split

```
In [182...]: #Drop 'Credit_Score' from df_train before applying the column transformer
```

```
X = df_train.drop(columns ="Credit_Score")
y = df_train["Credit_Score"]
```

```
In [183...]: seed = 101
X_train, X_val, y_train, y_val = train_test_split(X,
                                                    y,
                                                    test_size=0.2,
                                                    stratify=y,
                                                    random_state=seed)
```

```
In [184...]: print('-----')
print("X_train Data rows-columns: ", X_train.shape)
print("y_train Target Feature rows: ", y_train.shape)
```

```

print("X_test Validation Data rows-columns: ", X_val.shape)
print("y_test Validation Data rows-columns: ", y_val.shape)
print('-----')
# Original Test Dataset for Submission
print("Test Dataset rows-columns: ", df_test.shape)
print('-----')

```

```

-----
X_train Data rows-columns: (75610, 29)
y_train Target Feature rows: (75610,)
X_test Validation Data rows-columns: (18903, 29)
y_test Validation Data rows-columns: (18903,)

-----
Test Dataset rows-columns: (46609, 29)
-----
```

Ordinal and One-Hot Encoding the all Categoricals

In [185...]: df_train['Credit_Mix'].value_counts()

Out[185...]:

Credit_Mix	Count
Standard	34607
Good	22859
Unknown	19147
Bad	17900

Name: count, dtype: int64

In [186...]: df_train['Payment_Behaviour'].value_counts()

Out[186...]:

Payment_Behaviour	Count
Low_spent_Small_value_payments	26464
High_spent_Medium_value_payments	18353
High_spent_Large_value_payments	14612
Low_spent_Medium_value_payments	13698
High_spent_Small_value_payments	11150
Low_spent_Large_value_payments	10236

Name: count, dtype: int64

In [187...]: df_train['Payment_of_Min_Amount'].value_counts()

Out[187...]:

Payment_of_Min_Amount	Count
Yes	56272
No	38241

Name: count, dtype: int64

In [188...]:

```

from sklearn.compose import make_column_transformer
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder

# Features to be ordinaly encoded
ordinal_features = ['Credit_Mix', 'Payment_of_Min_Amount', 'Payment_Behaviour']

# Features to be one-hot encoded
onehot_features = ['Occupation']

# Define the ordinal categories for each feature
credit_mix_order = ['Bad', 'Standard', 'Good', 'Unknown'] # Adjusted based on value counts
payment_min_order = ['No', 'Yes'] # Binary (No < Yes)
payment_behaviour_order = ['Low_spent_Small_value_payments',
                           'Low_spent_Medium_value_payments',
                           'Low_spent_Large_value_payments',
                           'High_spent_Small_value_payments',
                           'High_spent_Medium_value_payments',
                           'High_spent_Large_value_payments']

# Specify the order for these features
ordinal_categories = [credit_mix_order, payment_min_order, payment_behaviour_order]

# Define the column transformer with OrdinalEncoder and OneHotEncoder
column_transformed = make_column_transformer(
    (OrdinalEncoder(categories=ordinal_categories, dtype=int, handle_unknown="use_encoded_value", unknown_v
    (OneHotEncoder(handle_unknown="ignore", dtype=int), onehot_features), # One-hot encoding
    remainder='passthrough', # Leave other features as is
    verbose_feature_names_out=False)

```

```
In [189... # Apply the column transformer to X-train and X_val!
X_train_transformed = column_transformed.fit_transform(X_train) # Train data
X_val_transformed = column_transformed.transform(X_val) # Validation test data
```

```
In [190... # OrdinalEncoder features and codes
ordinal_features = ['Credit_Mix', 'Payment_of_Min_Amount', 'Payment_Behaviour']
for feature, categories in zip(ordinal_features, ordinal_categories):
    print(f"\nFeature: {feature}")
    for i, category in enumerate(categories):
        print(f"  {category} -> {i}")

Feature: Credit_Mix
Bad -> 0
Standard -> 1
Good -> 2
Unknown -> 3

Feature: Payment_of_Min_Amount
No -> 0
Yes -> 1

Feature: Payment_Behaviour
Low_spent_Small_value_payments -> 0
Low_spent_Medium_value_payments -> 1
Low_spent_Large_value_payments -> 2
High_spent_Small_value_payments -> 3
High_spent_Medium_value_payments -> 4
High_spent_Large_value_payments -> 5
```

```
In [191... features = column_transformed.get_feature_names_out()
features
```

```
Out[191... array(['Credit_Mix', 'Payment_of_Min_Amount', 'Payment_Behaviour',
       'Occupation_Accountant', 'Occupation_Architect',
       'Occupation_Developer', 'Occupation_Doctor', 'Occupation_Engineer',
       'Occupation_Entrepreneur', 'Occupation_Journalist',
       'Occupation_Lawyer', 'Occupation_Manager', 'Occupation_Mechanic',
       'Occupation_Media_Manager', 'Occupation_Musician',
       'Occupation_Scientist', 'Occupation_Teacher', 'Occupation_Unknown',
       'Occupation_Writer', 'Age', 'Annual_Income',
       'Monthly_Inhand_Salary', 'Num_Bank_Accounts', 'Num_Credit_Card',
       'Interest_Rate', 'Num_of_Loan', 'Delay_from_due_date',
       'Num_of_Delayed_Payment', 'Changed_Credit_Limit',
       'Num_Credit_Inquiries', 'Outstanding_Debt',
       'Credit_Utilization_Ratio', 'Credit_History_Age',
       'Total_EMI_per_month', 'Amount_invested_monthly',
       'Monthly_Balance', 'Credit-Builder Loan', 'Personal Loan',
       'Debt Consolidation Loan', 'Student Loan', 'Payday Loan',
       'Mortgage Loan', 'Auto Loan', 'Home Equity Loan'], dtype=object)
```

```
In [192... # Reassign the transformed features to X train data
X_train = pd.DataFrame(X_train_transformed, columns=features, index=X_train.index)
X_train.head()
```

```
Out[192...
Credit_Mix  Payment_of_Min_Amount  Payment_Behaviour  Occupation_Accountant  Occupation_Architect  Occu
88708      0.000                 1.000             2.000                  0.000                  0.000
72254      1.000                 0.000             2.000                  0.000                  0.000
42173      1.000                 1.000             0.000                  0.000                  0.000
60665      0.000                 1.000             2.000                  0.000                  0.000
17746      2.000                 0.000             3.000                  1.000                  0.000
5 rows × 44 columns
```

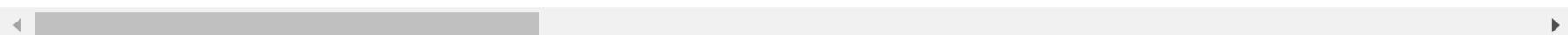
```
In [193... # Reassign the transformed features to Validation X_test
X_val = pd.DataFrame(X_val_transformed, columns=features, index=X_val.index)
X_val.head()
```

Out[193...]

	Credit_Mix	Payment_of_Min_Amount	Payment_Behaviour	Occupation_Accountant	Occupation_Architect	Occu
--	------------	-----------------------	-------------------	-----------------------	----------------------	------

29108	0.000	1.000	2.000	0.000	0.000	
73621	2.000	0.000	5.000	0.000	0.000	
25814	2.000	0.000	2.000	0.000	0.000	
59837	3.000	0.000	0.000	0.000	0.000	
87303	1.000	0.000	0.000	0.000	0.000	

5 rows × 44 columns



Scale: MinMaxScaler

In [194...]

```
# Initialize the MinMaxScaler
scaler = MinMaxScaler()

# Fit the scaler on X_train and transform both X_train and X_val
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)

# Convert scaled arrays back to DataFrame
X_train = pd.DataFrame(X_train_scaled, columns=X_train.columns)
X_val = pd.DataFrame(X_val_scaled, columns=X_val.columns)

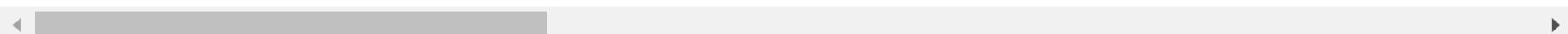
X_train.head(3)
```

Out[194...]

	Credit_Mix	Payment_of_Min_Amount	Payment_Behaviour	Occupation_Accountant	Occupation_Architect	Occupatio
--	------------	-----------------------	-------------------	-----------------------	----------------------	-----------

0	0.000	1.000	0.400	0.000	0.000	
1	0.333	0.000	0.400	0.000	0.000	
2	0.333	1.000	0.000	0.000	0.000	

3 rows × 44 columns



SMOTE for Imbalanced Data

SMOTE (Synthetic Minority Over-sampling Technique) is a technique used to address class imbalance by generating synthetic examples for the minority class. It should only be applied to the **training set**, as it creates new synthetic data that should not be used in the **validation** or **test sets**. Using SMOTE on these sets would interfere with the evaluation of the model's performance on real, unseen data.

In [195...]

```
y_train.value_counts()
```

Out[195...]

```
Credit_Score
1    40209
0    22031
2    13370
Name: count, dtype: int64
```

In [196...]

```
from imblearn.over_sampling import SMOTE

# Initialize the SMOTE object
smote = SMOTE(sampling_strategy='auto', random_state=seed)

# Apply SMOTE only to the training data
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

# X_train_resampled and y_train_resampled are now ready for model training
print("X_train shape :", X_train_smote.shape)
print("y_train shape :", y_train_smote.shape)
```

```

print("X_test shape :", X_val.shape)
print("y_test shape :", y_val.shape)

y_train_smote.value_counts()

X_train shape : (120627, 44)
y_train shape : (120627,)
X_test shape : (18903, 44)
y_test shape : (18903,)

Out[196... Credit_Score
1    40209
2    40209
0    40209
Name: count, dtype: int64

```

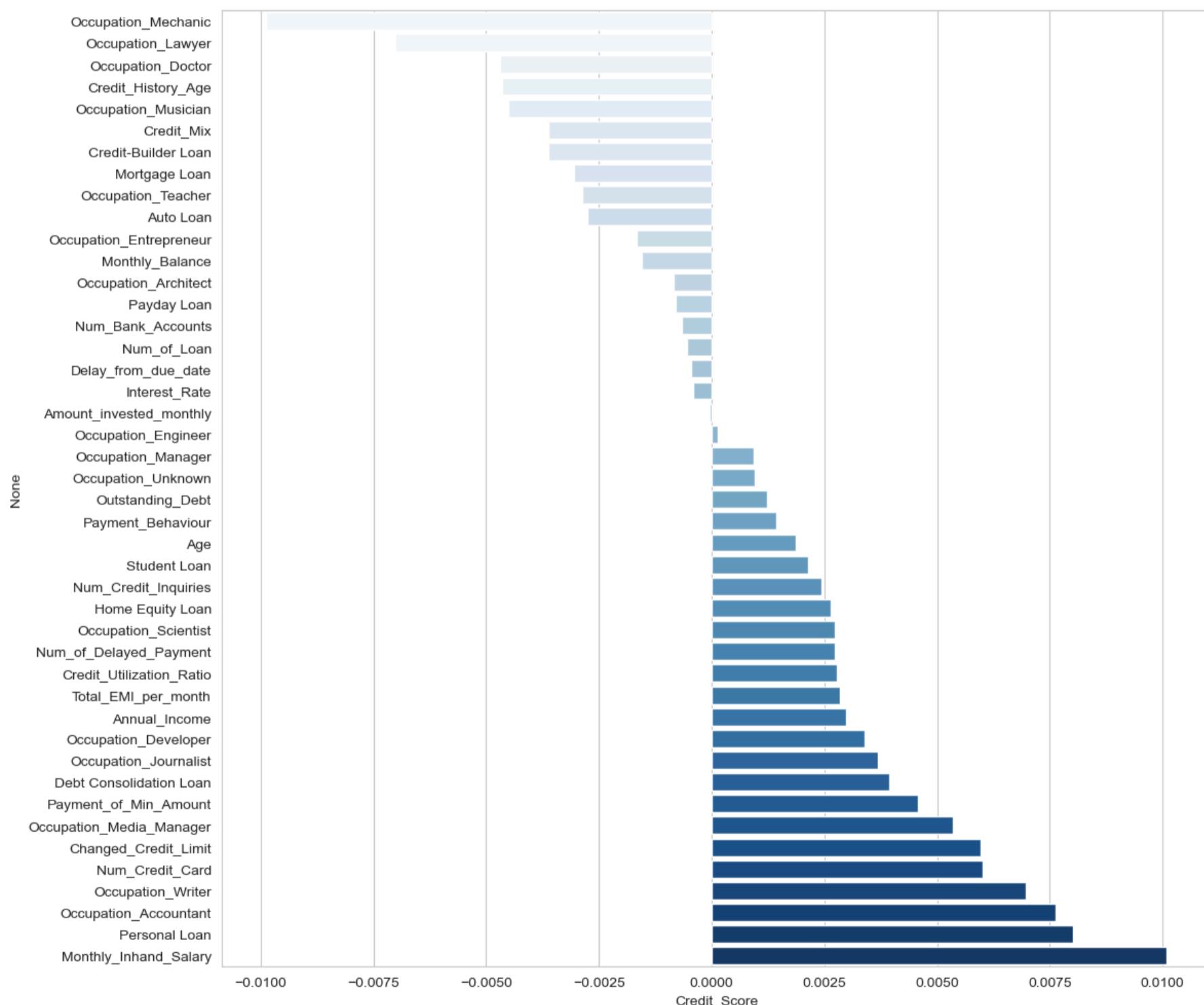
Correlation

```

In [197... # Calculate correlation with Credit_Score and sort
corr_by_target = X_train.join(y_train).corr()["Credit_Score"].sort_values()[:-1]

# Create the figure for the horizontal bar plot
plt.figure(figsize=(12, 10))
sns.barplot(x=corr_by_target, y=corr_by_target.index, palette='Blues')
plt.tight_layout()
plt.show()

```



- **Monthly Inhand Salary** and **Personal Loan** have the strongest positive impact on the **Credit Score**. This indicates that individuals with higher in-hand salaries and personal loans tend to have better credit scores.
- **Occupation Mechanic**, **Occupation Lawyer**, and **Occupation Doctor** show negative impacts, suggesting that individuals in these occupations might have lower credit scores or less influence on their credit scores compared to other features.
- Features related to financial behavior, such as **Credit Utilization Ratio**, **Num of Delayed Payments**, **Outstanding Debt**, and **Annual Income**, play a significant role in determining credit scores, with a positive effect.

In summary, financial factors and loan-related features are the most influential in determining credit scores, while some occupations seem to have a minimal or negative impact.

In [198...]

```
# ##### User-Defined-Function #####
# Function to Evaluate the Model Performans using Classification Confusion_matrix()
# Also does the prediction in the function

def eval_metric(model, X_train, y_train, X_test, y_test):
    y_train_pred_probabilities = model.predict(X_train)
    y_train_pred = y_train_pred_probabilities.argmax(axis=1)

    y_pred_probabilities = model.predict(X_test)
    y_pred = y_pred_probabilities.argmax(axis=1)

    print("Test Set:")
    print(confusion_matrix(y_test, y_pred))
    print(classification_report(y_test, y_pred))

    print("\nTrain Set:")
    print(confusion_matrix(y_train, y_train_pred))
    print(classification_report(y_train, y_train_pred))
```

In [199...]

```
# ##### User-Defined-Function #####
# Plot Feature Importance
def plot_feature_importances(model, X):
    """
    Computes and plots feature importances for the given model using the provided data (X).
    Assumes X is a pandas DataFrame with column names.

    Args:
        model: Trained TensorFlow/Keras model.
        X: Input features data (pandas DataFrame).
    """

    # Check if X is a DataFrame and get feature names
    if isinstance(X, pd.DataFrame):
        feature_names = X.columns.tolist()
    else:
        raise ValueError("X must be a pandas DataFrame with column names.")

    # Convert the input data to a TensorFlow tensor
    X_tensor = tf.convert_to_tensor(X, dtype=tf.float32)

    # Calculate gradients of the output with respect to the input features
    with tf.GradientTape() as tape:
        tape.watch(X_tensor)
        predictions = model(X_tensor)
    gradients = tape.gradient(predictions, X_tensor)

    # Compute feature importances as the absolute sum of gradients
    feature_importances = np.abs(gradients.numpy()).mean(axis=0)

    # Normalize feature importances to percentages
    feature_importances_percent = (feature_importances / feature_importances.sum()) * 100

    # Create a DataFrame to display feature importances
    importance_df = pd.DataFrame({
        'Feature': feature_names,
        'Importance (%)': feature_importances_percent.round(2)
    }).sort_values(by='Importance (%)', ascending=False)

    # Plot with feature importance values on top of each bar
    plt.figure(figsize=(18, 8))
    bars = plt.bar(importance_df['Feature'], importance_df['Importance (%)'])

    # Add text labels on top of the bars
    for bar, importance in zip(bars, importance_df['Importance (%)']):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.1,
                 f'{importance:.2f}%', ha='center', va='bottom', rotation=90, fontsize=12)

    plt.xticks(rotation=90, ha='right', fontsize=12)
    plt.xlabel('Features')
    plt.ylabel('Importance (%)')
    plt.title('Feature Importances')
```

```
plt.tight_layout()  
plt.show()
```

ANN Model

An **Artificial Neural Network (ANN)** is a machine learning model that mimics the human brain's structure to learn patterns in data. It consists of layers of interconnected nodes (neurons) that adjust weights to improve predictions.

We use ANN for this dataset because it can capture complex, non-linear relationships between features like credit score, payment history, and income. Its flexibility and ability to learn from large amounts of data make it ideal for achieving high accuracy on this dataset.

Required Libraries:

```
In [210...]  
import tensorflow as tf  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization  
from tensorflow.keras.optimizers import Adam, Nadam, AdamW  
from tensorflow.keras.callbacks import EarlyStopping  
from tensorflow.keras.metrics import Recall, Precision  
from tensorflow.keras.regularizers import l2  
from tensorflow.keras.callbacks import ReduceLROnPlateau  
from tensorflow.keras.optimizers import SGD
```

ANN-1 Model (Vanilla) (%69)

```
In [ ]: # 1) Model Architecture:
```

```
# Fully connected (Dense) Layers number start with 5
# Neurons start with = 256
ann1 = Sequential([
    Dense(256, input_dim=X_train.shape[1], activation='relu'),
    Dropout(0.3),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dropout(0.25),
    Dense(64, activation='relu'),
    Dropout(0.25),
    Dense(3, activation='softmax')
])

# 2) Compiling the Model:
# Compile the model with accuracy, (can add Precision and Recall later)
ann1.compile(optimizer=Adam(learning_rate=0.01),    # Learning_rate start with = 0.01
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 3) Early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=50,
                               mode="auto",
                               restore_best_weights=True)

# 4) Train the model
history = ann1.fit(
    x=X_train,
    y=y_train,
    validation_data=(X_val, y_val),
    batch_size= 512,    # batch_size start with = 512
    epochs=150,         # epochs start with = 150
    verbose=1,
    callbacks=[early_stopping])
```

```
In [ ]: ann1.evaluate(X_train, y_train)
```

```
In [ ]: ann1.summary()
```

```
In [ ]: eval_metric(ann1, X_train, y_train, X_val, y_val)
```

```
In [ ]: pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```

```
In [ ]: pd.DataFrame(history.history)[["accuracy"]].plot(figsize=(10, 5))
plt.title("Accuracy improvements with Epoch")
plt.show()
```

ANN-1 Model:

- (Dense) layers: 5 / Neurons: 256-128-64-64 /Dropout: 25-30% / Learning Rate: 0.01 / Batch Size: 512 / Epochs: 150 / Early Stop (val_accuracy): 50
- Accuracy: 0.69 / Val_Accuracy: 0.69 / Loss: 0.72 / Val_Loss: 0.70 / Train Recall(Class 2): 0.77 / Test Recall (Class 2): 0.75

Demonstrates moderate performance with training and validation accuracies hovering around 66-69%.

The model exhibits significant variability in performance across different classes, notably with class 2 showing high recall but low precision, indicating potential issues with class imbalance or misclassification.

Adjustments in learning rate, training epochs, and model architecture might be necessary to enhance generalization and overall performance.

ANN-1 with SMOTE (%67)

```
In [ ]: print('Credi_Score Classes before Smote: ', y_train.value_counts())
print('\nCredi_Score Classes after Smote: ', y_train_smote.value_counts())

In [ ]: # 1) Model Architecture:

# Fully connected (Dense) Layers number start with 5
# Neurons start with = 256
ann1_smote = Sequential([
    Dense(256, input_dim=X_train_smote.shape[1], activation='relu'),
    Dropout(0.3),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dropout(0.25),
    Dense(64, activation='relu'),
    Dropout(0.25),
    Dense(3, activation='softmax')
])

# 2) Compiling the Model:
# Compile the model with accuracy, (can add Precision and Recall later)
ann1_smote.compile(optimizer=Adam(learning_rate=0.01), # Learning_rate start with = 0.01
                    loss='sparse_categorical_crossentropy',
                    metrics=['accuracy'])

# 3) Early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=50,
                               mode="auto",
                               restore_best_weights=True)

# 4) Train the model
history = ann1_smote.fit(
    x=X_train_smote,
    y=y_train_smote,
    validation_data=(X_val, y_val),
    batch_size=512,      # batch_size start with = 512
    epochs=150,          # epochs start with = 150
    verbose=1,
    callbacks=[early_stopping])

In [ ]: ann1_smote.evaluate(X_train_smote, y_train_smote)

In [ ]: ann1_smote.summary()

In [ ]: eval_metric(ann1_smote, X_train_smote, y_train_smote, X_val, y_val)

In [ ]: pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()

In [ ]: pd.DataFrame(history.history)[["accuracy"]].plot(figsize=(10, 5))
plt.title("Accuracy improvements with Epoch")
plt.show()
```

ANN-1_Smote Model Summary:

- **(Dense) layers:** 5 / **Neurons:** 256-128-64-64 / **Dropout:** 25-30% / **Learning Rate:** 0.01 / **Batch Size:** 512 / **Epochs:** 150 / **Early Stop (val_accuracy):** 50
- **Accuracy:** 0.6765 / **Val_Accuracy:** 0.666 / **Loss:** 0.7417 / **Val_Loss:** 0.7498 / **Train Recall(Class 2):** 0.85 / **Test Recall (Class 2):** 0.83

The SMOTE technique has enhanced the recall for class 2 in the ANN-1 model, showing better sensitivity towards the minority class.

While this has improved class balance, it also resulted in higher validation loss, indicating a compromise on model precision.

Despite the increase in loss, overall accuracy has been maintained, suggesting the model is still learning effectively. Adjustments to the model could aim to reduce loss without sacrificing gains in recall.

ANN-2 Model: +Neurons -LearningRate -Batchsize +Epoch(%86)

```
In [ ]: # 1) Model Architecture:

# Fully connected (Dense) Layers number: 5
# Increased Neurons number from 256 to ----> 512
ann2 = Sequential([
    Dense(512, input_dim=X_train.shape[1], activation='relu'),
    Dropout(0.4),
    Dense(256, activation='relu'),
    Dropout(0.3),
    Dense(128, activation='relu'),
    Dropout(0.25),
    Dense(64, activation='relu'),
    Dropout(0.25),
    Dense(3, activation='softmax')
])

# 2) Compiling the Model:
# Decreased Learning_rate from 0.01 to---> 0.001
ann2.compile(optimizer=Adam(learning_rate=0.001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 3) Early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=50,
                               mode="auto",
                               restore_best_weights=True)

# 4) Train the model
history = ann2.fit(
    x=X_train,
    y=y_train,
    validation_data=(X_val, y_val),
    batch_size=256, # Decreased batch_size from 512 to ----> 256
    epochs=300,      # Increased Epoch from 150 to -----> 300
    verbose=1,
    callbacks=[early_stopping])
```

```
In [24]: ann2.evaluate(X_train, y_train)
```

2363/2363 ━━━━━━━━ 4s 2ms/step - accuracy: 0.8672 - loss: 0.3283

```
Out[24]: [0.3275914490222931, 0.8693029880523682]
```

```
In [25]: ann2.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	23,040
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131,328
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32,896
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 64)	8,256
dropout_3 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 3)	195

Total params: 587,147 (2.24 MB)

Trainable params: 195,715 (764.51 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 391,432 (1.49 MB)

```
In [26]: eval_metric(ann2, X_train, y_train, X_val, y_val)
```

2363/2363 ————— 4s 2ms/step
591/591 ————— 1s 2ms/step

Test Set:

```
[[4473 877 158]
 [1425 7625 1003]
 [ 71 806 2465]]
```

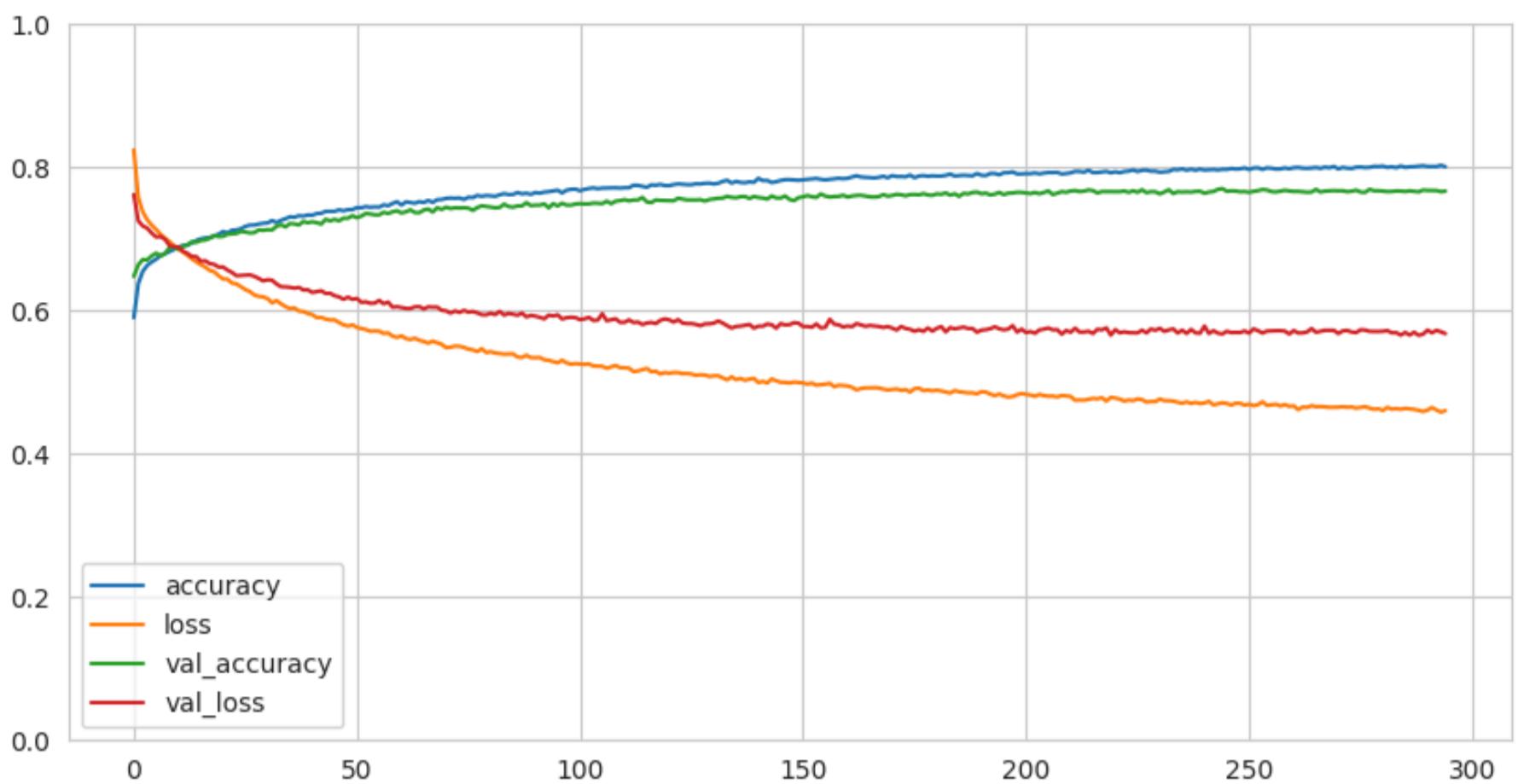
	precision	recall	f1-score	support
0	0.75	0.81	0.78	5508
1	0.82	0.76	0.79	10053
2	0.68	0.74	0.71	3342
accuracy			0.77	18903
macro avg	0.75	0.77	0.76	18903
weighted avg	0.77	0.77	0.77	18903

Train Set:

```
[[20361 1463 207]
 [ 3751 33669 2789]
 [ 38 1634 11698]]
```

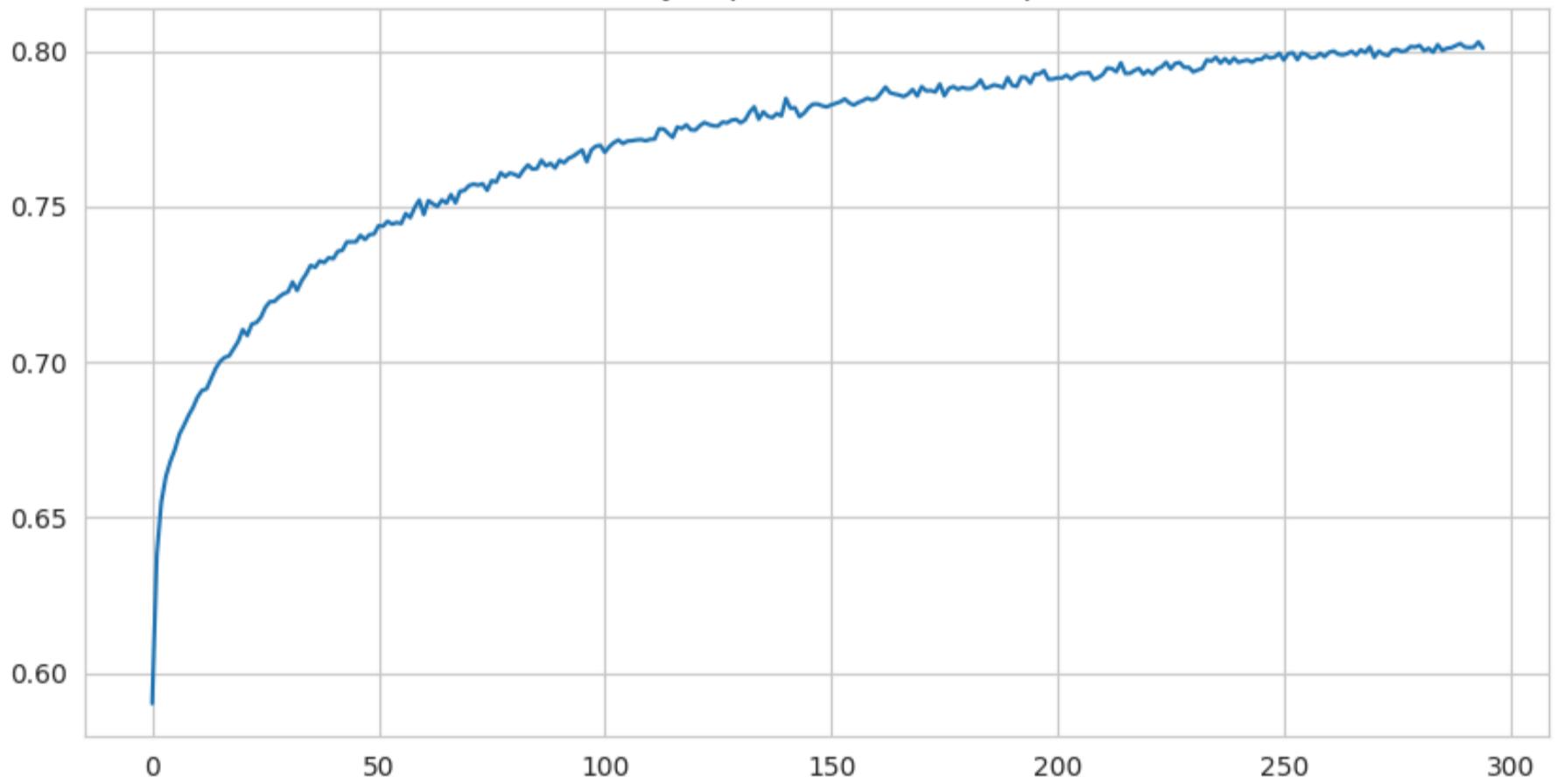
	precision	recall	f1-score	support
0	0.84	0.92	0.88	22031
1	0.92	0.84	0.87	40209
2	0.80	0.87	0.83	13370
accuracy			0.87	75610
macro avg	0.85	0.88	0.86	75610
weighted avg	0.87	0.87	0.87	75610

```
In [27]: pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



```
In [28]: pd.DataFrame(history.history)[“accuracy”].plot(figsize=(10, 5))
plt.title(“Accuracy improvements with Epoch”)
plt.show()
```

Accuracy improvements with Epoch



```
In [31]: # Save the Model

from tensorflow.keras.models import load_model

# Save the model to 'model.h5' file
ann2.save('ann2_model.h5')

# To Load the model later for further use
loaded_ann2 = load_model('ann2_model.h5')
```

ANN-2 Model Summary:

- **(Dense) layers:** 5 / **Neurons:** 512-256-128-64 / **Dropout:** 40-30-25-25% / **Learning Rate:** 0.001 / **Batch Size:** 256 / **Epochs:** 300 / **Early Stop (val_accuracy):** 50
- **Accuracy:** 0.8693 / **Val_Accuracy:** 0.7658 / **Loss:** 0.3276 / **Val_Loss:** 0.5716 / **Train Recall (Class 2):** 0.87 / **Test Recall(Class 2):** 0.71

The ANN-2 model shows substantial improvements in training accuracy and a reduction in loss, thanks to optimized neuron configurations and a lower learning rate.

However, the significant gap between training and validation performance, particularly with higher validation loss, suggests possible overfitting.

This is further indicated by the higher recall for class 2 in training, showing that while the model recognizes the minority class well during training, it doesn't perform as well on unseen data.

To mitigate this and improve model generalization, further adjustments such as increasing dropout rates, adding more regularization, or employing techniques like SMOTE or class weight adjustments could be explored in subsequent iterations.

ANN-2 Model with SMOTE (%82)

```
In [29]: print('Credi_Score Classes before Smote: ', y_train.value_counts())
print('\nCredi_Score Classes after Smote: ', y_train_smote.value_counts())
```

```
Credi_Score Classes before Smote: Credit_Score
1    40209
0    22031
2    13370
Name: count, dtype: int64
```

```
Credi_Score Classes after Smote: Credit_Score
1    40209
2    40209
0    40209
Name: count, dtype: int64
```

```
In [ ]: from sklearn.utils import class_weight

class_weights = class_weight.compute_class_weight('balanced',
                                                    classes=np.unique(y_train),
                                                    y=y_train)
class_weights_dict = {i: class_weights[i] for i in range(len(class_weights))}
class_weights_dict
```

```
In [231...]: # ANN-2 Model with SMOTE
```

```
# 1) Model Architecture:
# Fully connected (Dense) Layers number: 5

# Increased Neurons number from 256 to ----> 512
ann2_smote = Sequential([
    Dense(512, input_dim=X_train_smote.shape[1], activation='relu'),
    Dropout(0.4),
    Dense(256, activation='relu'),
    Dropout(0.3),
    Dense(128, activation='relu'),
    Dropout(0.25),
    Dense(64, activation='relu'),
    Dropout(0.25),
    Dense(3, activation='softmax')
])
```

```
# 2) Compiling the Model:
# Decreased Learning_rate from 0.01 to---> 0.001
ann2_smote.compile(optimizer=Adam(learning_rate=0.001),
                    loss='sparse_categorical_crossentropy',
                    metrics=['accuracy'])
```

```
# 3) Early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=50,
                               mode="auto",
                               restore_best_weights=True)
```

```
# 4) Train the model
history = ann2_smote.fit(
    x=X_train_smote, # SMOTE X_train and y_train
    y=y_train_smote,
```

```
validation_data=(X_val, y_val),  
batch_size=256, # Decreased batch_size from 512 to ----> 256  
epochs=300,      # Increased Epoch from 150 to -----> 300  
verbose=1,  
callbacks=[early_stopping],  
class_weight = class_weights_dict)    # + class_weight is been used
```

Epoch 1/300
472/472 4s 4ms/step - accuracy: 0.5826 - loss: 0.9203 - val_accuracy: 0.5767 - val_loss:
s: 1.0129
Epoch 2/300
472/472 2s 4ms/step - accuracy: 0.6626 - loss: 0.7811 - val_accuracy: 0.6148 - val_loss:
s: 0.9387
Epoch 3/300
472/472 2s 4ms/step - accuracy: 0.6811 - loss: 0.7433 - val_accuracy: 0.5971 - val_loss:
s: 0.9696
Epoch 4/300
472/472 2s 4ms/step - accuracy: 0.6938 - loss: 0.7074 - val_accuracy: 0.6140 - val_loss:
s: 0.9217
Epoch 5/300
472/472 2s 4ms/step - accuracy: 0.7052 - loss: 0.6852 - val_accuracy: 0.6065 - val_loss:
s: 0.9358
Epoch 6/300
472/472 2s 4ms/step - accuracy: 0.7098 - loss: 0.6712 - val_accuracy: 0.5995 - val_loss:
s: 0.9530
Epoch 7/300
472/472 2s 4ms/step - accuracy: 0.7138 - loss: 0.6551 - val_accuracy: 0.6142 - val_loss:
s: 0.9236
Epoch 8/300
472/472 2s 4ms/step - accuracy: 0.7198 - loss: 0.6391 - val_accuracy: 0.6247 - val_loss:
s: 0.9156
Epoch 9/300
472/472 2s 4ms/step - accuracy: 0.7249 - loss: 0.6239 - val_accuracy: 0.6181 - val_loss:
s: 0.9227
Epoch 10/300
472/472 2s 4ms/step - accuracy: 0.7250 - loss: 0.6189 - val_accuracy: 0.6283 - val_loss:
s: 0.8959
Epoch 11/300
472/472 2s 4ms/step - accuracy: 0.7307 - loss: 0.6070 - val_accuracy: 0.6167 - val_loss:
s: 0.9093
Epoch 12/300
472/472 2s 4ms/step - accuracy: 0.7345 - loss: 0.5971 - val_accuracy: 0.6277 - val_loss:
s: 0.8684
Epoch 13/300
472/472 2s 4ms/step - accuracy: 0.7352 - loss: 0.5871 - val_accuracy: 0.6339 - val_loss:
s: 0.8799
Epoch 14/300
472/472 2s 3ms/step - accuracy: 0.7412 - loss: 0.5785 - val_accuracy: 0.6279 - val_loss:
s: 0.8664
Epoch 15/300
472/472 2s 4ms/step - accuracy: 0.7430 - loss: 0.5719 - val_accuracy: 0.6366 - val_loss:
s: 0.8666
Epoch 16/300
472/472 2s 3ms/step - accuracy: 0.7454 - loss: 0.5620 - val_accuracy: 0.6365 - val_loss:
s: 0.8617
Epoch 17/300
472/472 2s 3ms/step - accuracy: 0.7480 - loss: 0.5616 - val_accuracy: 0.6282 - val_loss:
s: 0.8970
Epoch 18/300
472/472 2s 4ms/step - accuracy: 0.7461 - loss: 0.5562 - val_accuracy: 0.6381 - val_loss:
s: 0.8815
Epoch 19/300
472/472 2s 4ms/step - accuracy: 0.7490 - loss: 0.5506 - val_accuracy: 0.6421 - val_loss:
s: 0.8553
Epoch 20/300
472/472 2s 4ms/step - accuracy: 0.7525 - loss: 0.5457 - val_accuracy: 0.6432 - val_loss:
s: 0.8500
Epoch 21/300
472/472 2s 4ms/step - accuracy: 0.7502 - loss: 0.5431 - val_accuracy: 0.6464 - val_loss:
s: 0.8507
Epoch 22/300
472/472 2s 3ms/step - accuracy: 0.7541 - loss: 0.5400 - val_accuracy: 0.6441 - val_loss:
s: 0.8770
Epoch 23/300
472/472 2s 4ms/step - accuracy: 0.7551 - loss: 0.5342 - val_accuracy: 0.6458 - val_loss:
s: 0.8302
Epoch 24/300
472/472 2s 4ms/step - accuracy: 0.7552 - loss: 0.5302 - val_accuracy: 0.6516 - val_loss:
s: 0.8249
Epoch 25/300
472/472 2s 3ms/step - accuracy: 0.7591 - loss: 0.5239 - val_accuracy: 0.6459 - val_loss:
s: 0.8310
Epoch 26/300

472/472 2s 4ms/step - accuracy: 0.7599 - loss: 0.5213 - val_accuracy: 0.6488 - val_loss: 0.8401
Epoch 27/300
472/472 2s 3ms/step - accuracy: 0.7629 - loss: 0.5137 - val_accuracy: 0.6552 - val_loss: 0.8309
Epoch 28/300
472/472 2s 3ms/step - accuracy: 0.7620 - loss: 0.5150 - val_accuracy: 0.6431 - val_loss: 0.8402
Epoch 29/300
472/472 2s 4ms/step - accuracy: 0.7619 - loss: 0.5114 - val_accuracy: 0.6548 - val_loss: 0.8153
Epoch 30/300
472/472 2s 4ms/step - accuracy: 0.7636 - loss: 0.5085 - val_accuracy: 0.6479 - val_loss: 0.8352
Epoch 31/300
472/472 2s 4ms/step - accuracy: 0.7649 - loss: 0.5076 - val_accuracy: 0.6469 - val_loss: 0.8400
Epoch 32/300
472/472 2s 4ms/step - accuracy: 0.7664 - loss: 0.5024 - val_accuracy: 0.6608 - val_loss: 0.8120
Epoch 33/300
472/472 2s 4ms/step - accuracy: 0.7698 - loss: 0.4961 - val_accuracy: 0.6685 - val_loss: 0.7854
Epoch 34/300
472/472 2s 3ms/step - accuracy: 0.7699 - loss: 0.4977 - val_accuracy: 0.6466 - val_loss: 0.8340
Epoch 35/300
472/472 2s 3ms/step - accuracy: 0.7663 - loss: 0.4978 - val_accuracy: 0.6592 - val_loss: 0.8231
Epoch 36/300
472/472 2s 4ms/step - accuracy: 0.7730 - loss: 0.4904 - val_accuracy: 0.6579 - val_loss: 0.8095
Epoch 37/300
472/472 2s 4ms/step - accuracy: 0.7712 - loss: 0.4901 - val_accuracy: 0.6622 - val_loss: 0.8122
Epoch 38/300
472/472 2s 4ms/step - accuracy: 0.7705 - loss: 0.4906 - val_accuracy: 0.6629 - val_loss: 0.8153
Epoch 39/300
472/472 2s 4ms/step - accuracy: 0.7739 - loss: 0.4876 - val_accuracy: 0.6504 - val_loss: 0.8502
Epoch 40/300
472/472 2s 4ms/step - accuracy: 0.7738 - loss: 0.4853 - val_accuracy: 0.6569 - val_loss: 0.8297
Epoch 41/300
472/472 2s 3ms/step - accuracy: 0.7748 - loss: 0.4838 - val_accuracy: 0.6647 - val_loss: 0.8164
Epoch 42/300
472/472 2s 4ms/step - accuracy: 0.7776 - loss: 0.4791 - val_accuracy: 0.6654 - val_loss: 0.8131
Epoch 43/300
472/472 2s 4ms/step - accuracy: 0.7767 - loss: 0.4807 - val_accuracy: 0.6654 - val_loss: 0.7992
Epoch 44/300
472/472 2s 4ms/step - accuracy: 0.7766 - loss: 0.4817 - val_accuracy: 0.6710 - val_loss: 0.8031
Epoch 45/300
472/472 2s 4ms/step - accuracy: 0.7747 - loss: 0.4805 - val_accuracy: 0.6717 - val_loss: 0.7928
Epoch 46/300
472/472 2s 4ms/step - accuracy: 0.7798 - loss: 0.4759 - val_accuracy: 0.6677 - val_loss: 0.7983
Epoch 47/300
472/472 2s 4ms/step - accuracy: 0.7792 - loss: 0.4723 - val_accuracy: 0.6794 - val_loss: 0.7777
Epoch 48/300
472/472 2s 4ms/step - accuracy: 0.7754 - loss: 0.4782 - val_accuracy: 0.6747 - val_loss: 0.8023
Epoch 49/300
472/472 2s 4ms/step - accuracy: 0.7802 - loss: 0.4722 - val_accuracy: 0.6704 - val_loss: 0.7993
Epoch 50/300
472/472 2s 3ms/step - accuracy: 0.7808 - loss: 0.4685 - val_accuracy: 0.6638 - val_loss: 0.8272
Epoch 51/300
472/472 2s 3ms/step - accuracy: 0.7821 - loss: 0.4676 - val_accuracy: 0.6636 - val_loss:

s: 0.8178
Epoch 52/300
472/472 2s 3ms/step - accuracy: 0.7828 - loss: 0.4633 - val_accuracy: 0.6680 - val_loss: 0.7812
s: 0.8062
Epoch 53/300
472/472 2s 3ms/step - accuracy: 0.7817 - loss: 0.4651 - val_accuracy: 0.6697 - val_loss: 0.7806
s: 0.8086
Epoch 54/300
472/472 2s 4ms/step - accuracy: 0.7793 - loss: 0.4659 - val_accuracy: 0.6710 - val_loss: 0.7804
s: 0.8004
Epoch 55/300
472/472 2s 4ms/step - accuracy: 0.7823 - loss: 0.4610 - val_accuracy: 0.6681 - val_loss: 0.7809
s: 0.8069
Epoch 56/300
472/472 2s 4ms/step - accuracy: 0.7843 - loss: 0.4566 - val_accuracy: 0.6750 - val_loss: 0.7811
s: 0.7851
Epoch 57/300
472/472 2s 4ms/step - accuracy: 0.7819 - loss: 0.4635 - val_accuracy: 0.6754 - val_loss: 0.7813
s: 0.7873
Epoch 58/300
472/472 2s 4ms/step - accuracy: 0.7844 - loss: 0.4606 - val_accuracy: 0.6734 - val_loss: 0.7923
s: 0.7923
Epoch 59/300
472/472 2s 4ms/step - accuracy: 0.7872 - loss: 0.4526 - val_accuracy: 0.6792 - val_loss: 0.7919
s: 0.7919
Epoch 60/300
472/472 2s 4ms/step - accuracy: 0.7866 - loss: 0.4550 - val_accuracy: 0.6760 - val_loss: 0.7947
s: 0.7947
Epoch 61/300
472/472 2s 3ms/step - accuracy: 0.7875 - loss: 0.4553 - val_accuracy: 0.6760 - val_loss: 0.7780
s: 0.7780
Epoch 62/300
472/472 2s 4ms/step - accuracy: 0.7879 - loss: 0.4533 - val_accuracy: 0.6765 - val_loss: 0.7801
s: 0.7801
Epoch 63/300
472/472 2s 4ms/step - accuracy: 0.7864 - loss: 0.4561 - val_accuracy: 0.6790 - val_loss: 0.7738
s: 0.7738
Epoch 64/300
472/472 2s 4ms/step - accuracy: 0.7872 - loss: 0.4574 - val_accuracy: 0.6841 - val_loss: 0.7917
s: 0.7917
Epoch 65/300
472/472 2s 4ms/step - accuracy: 0.7887 - loss: 0.4518 - val_accuracy: 0.6695 - val_loss: 0.7996
s: 0.7996
Epoch 66/300
472/472 2s 4ms/step - accuracy: 0.7861 - loss: 0.4519 - val_accuracy: 0.6780 - val_loss: 0.7966
s: 0.7966
Epoch 67/300
472/472 2s 4ms/step - accuracy: 0.7879 - loss: 0.4518 - val_accuracy: 0.6768 - val_loss: 0.8044
s: 0.8044
Epoch 68/300
472/472 2s 4ms/step - accuracy: 0.7889 - loss: 0.4504 - val_accuracy: 0.6833 - val_loss: 0.7857
s: 0.7857
Epoch 69/300
472/472 2s 3ms/step - accuracy: 0.7905 - loss: 0.4450 - val_accuracy: 0.6821 - val_loss: 0.7821
s: 0.7821
Epoch 70/300
472/472 2s 3ms/step - accuracy: 0.7916 - loss: 0.4431 - val_accuracy: 0.6747 - val_loss: 0.7985
s: 0.7985
Epoch 71/300
472/472 2s 3ms/step - accuracy: 0.7931 - loss: 0.4423 - val_accuracy: 0.6872 - val_loss: 0.8031
s: 0.8031
Epoch 72/300
472/472 2s 4ms/step - accuracy: 0.7925 - loss: 0.4401 - val_accuracy: 0.6835 - val_loss: 0.7964
s: 0.7964
Epoch 73/300
472/472 2s 4ms/step - accuracy: 0.7961 - loss: 0.4420 - val_accuracy: 0.6766 - val_loss: 0.8029
s: 0.8029
Epoch 74/300
472/472 2s 4ms/step - accuracy: 0.7916 - loss: 0.4409 - val_accuracy: 0.6841 - val_loss: 0.7811
s: 0.7811
Epoch 75/300
472/472 2s 4ms/step - accuracy: 0.7938 - loss: 0.4394 - val_accuracy: 0.6921 - val_loss: 0.7625
s: 0.7625
Epoch 76/300
472/472 2s 4ms/step - accuracy: 0.7966 - loss: 0.4360 - val_accuracy: 0.6863 - val_loss: 0.7762
s: 0.7762

Epoch 77/300
472/472 2s 4ms/step - accuracy: 0.7952 - loss: 0.4378 - val_accuracy: 0.6914 - val_loss: 0.7438
Epoch 78/300
472/472 2s 4ms/step - accuracy: 0.7959 - loss: 0.4366 - val_accuracy: 0.6923 - val_loss: 0.7565
Epoch 79/300
472/472 2s 4ms/step - accuracy: 0.7948 - loss: 0.4364 - val_accuracy: 0.6881 - val_loss: 0.7616
Epoch 80/300
472/472 2s 4ms/step - accuracy: 0.7941 - loss: 0.4372 - val_accuracy: 0.6813 - val_loss: 0.7883
Epoch 81/300
472/472 2s 4ms/step - accuracy: 0.7981 - loss: 0.4320 - val_accuracy: 0.6891 - val_loss: 0.7766
Epoch 82/300
472/472 2s 4ms/step - accuracy: 0.7940 - loss: 0.4355 - val_accuracy: 0.6951 - val_loss: 0.7701
Epoch 83/300
472/472 2s 4ms/step - accuracy: 0.7959 - loss: 0.4382 - val_accuracy: 0.6904 - val_loss: 0.7674
Epoch 84/300
472/472 2s 4ms/step - accuracy: 0.7986 - loss: 0.4289 - val_accuracy: 0.6858 - val_loss: 0.7656
Epoch 85/300
472/472 2s 4ms/step - accuracy: 0.7947 - loss: 0.4356 - val_accuracy: 0.6857 - val_loss: 0.7695
Epoch 86/300
472/472 2s 4ms/step - accuracy: 0.7955 - loss: 0.4341 - val_accuracy: 0.6879 - val_loss: 0.7670
Epoch 87/300
472/472 2s 4ms/step - accuracy: 0.7985 - loss: 0.4319 - val_accuracy: 0.6848 - val_loss: 0.7894
Epoch 88/300
472/472 2s 4ms/step - accuracy: 0.7999 - loss: 0.4268 - val_accuracy: 0.6909 - val_loss: 0.7549
Epoch 89/300
472/472 2s 4ms/step - accuracy: 0.7990 - loss: 0.4308 - val_accuracy: 0.6906 - val_loss: 0.7699
Epoch 90/300
472/472 2s 5ms/step - accuracy: 0.7994 - loss: 0.4322 - val_accuracy: 0.6859 - val_loss: 0.7801
Epoch 91/300
472/472 2s 4ms/step - accuracy: 0.7990 - loss: 0.4284 - val_accuracy: 0.6808 - val_loss: 0.7984
Epoch 92/300
472/472 2s 4ms/step - accuracy: 0.7965 - loss: 0.4303 - val_accuracy: 0.6881 - val_loss: 0.7874
Epoch 93/300
472/472 2s 4ms/step - accuracy: 0.8008 - loss: 0.4266 - val_accuracy: 0.6889 - val_loss: 0.7661
Epoch 94/300
472/472 2s 4ms/step - accuracy: 0.7995 - loss: 0.4266 - val_accuracy: 0.6811 - val_loss: 0.8086
Epoch 95/300
472/472 2s 4ms/step - accuracy: 0.8009 - loss: 0.4246 - val_accuracy: 0.6951 - val_loss: 0.7590
Epoch 96/300
472/472 2s 4ms/step - accuracy: 0.8001 - loss: 0.4249 - val_accuracy: 0.6858 - val_loss: 0.7848
Epoch 97/300
472/472 2s 4ms/step - accuracy: 0.7990 - loss: 0.4268 - val_accuracy: 0.6923 - val_loss: 0.7651
Epoch 98/300
472/472 2s 4ms/step - accuracy: 0.7981 - loss: 0.4272 - val_accuracy: 0.6969 - val_loss: 0.7633
Epoch 99/300
472/472 2s 4ms/step - accuracy: 0.8028 - loss: 0.4201 - val_accuracy: 0.6915 - val_loss: 0.7556
Epoch 100/300
472/472 2s 4ms/step - accuracy: 0.7992 - loss: 0.4232 - val_accuracy: 0.6960 - val_loss: 0.7505
Epoch 101/300
472/472 2s 4ms/step - accuracy: 0.8038 - loss: 0.4167 - val_accuracy: 0.6947 - val_loss: 0.7626
Epoch 102/300

472/472 2s 4ms/step - accuracy: 0.8026 - loss: 0.4209 - val_accuracy: 0.6911 - val_loss: 0.7672
Epoch 103/300
472/472 2s 4ms/step - accuracy: 0.8020 - loss: 0.4203 - val_accuracy: 0.6878 - val_loss: 0.7678
Epoch 104/300
472/472 2s 4ms/step - accuracy: 0.8018 - loss: 0.4216 - val_accuracy: 0.6969 - val_loss: 0.7590
Epoch 105/300
472/472 2s 4ms/step - accuracy: 0.8042 - loss: 0.4169 - val_accuracy: 0.6934 - val_loss: 0.7631
Epoch 106/300
472/472 2s 3ms/step - accuracy: 0.8045 - loss: 0.4172 - val_accuracy: 0.7000 - val_loss: 0.7495
Epoch 107/300
472/472 2s 4ms/step - accuracy: 0.8035 - loss: 0.4199 - val_accuracy: 0.6887 - val_loss: 0.7652
Epoch 108/300
472/472 2s 4ms/step - accuracy: 0.8038 - loss: 0.4195 - val_accuracy: 0.6957 - val_loss: 0.7617
Epoch 109/300
472/472 2s 4ms/step - accuracy: 0.8011 - loss: 0.4220 - val_accuracy: 0.6995 - val_loss: 0.7703
Epoch 110/300
472/472 2s 4ms/step - accuracy: 0.8051 - loss: 0.4184 - val_accuracy: 0.7020 - val_loss: 0.7621
Epoch 111/300
472/472 2s 3ms/step - accuracy: 0.8046 - loss: 0.4183 - val_accuracy: 0.6995 - val_loss: 0.7661
Epoch 112/300
472/472 2s 3ms/step - accuracy: 0.8066 - loss: 0.4157 - val_accuracy: 0.6938 - val_loss: 0.7583
Epoch 113/300
472/472 2s 3ms/step - accuracy: 0.8036 - loss: 0.4161 - val_accuracy: 0.6952 - val_loss: 0.7656
Epoch 114/300
472/472 2s 3ms/step - accuracy: 0.8052 - loss: 0.4181 - val_accuracy: 0.6994 - val_loss: 0.7605
Epoch 115/300
472/472 2s 4ms/step - accuracy: 0.8048 - loss: 0.4144 - val_accuracy: 0.6990 - val_loss: 0.7665
Epoch 116/300
472/472 2s 4ms/step - accuracy: 0.8081 - loss: 0.4136 - val_accuracy: 0.6930 - val_loss: 0.7699
Epoch 117/300
472/472 2s 4ms/step - accuracy: 0.8069 - loss: 0.4131 - val_accuracy: 0.7016 - val_loss: 0.7480
Epoch 118/300
472/472 2s 3ms/step - accuracy: 0.8066 - loss: 0.4148 - val_accuracy: 0.7052 - val_loss: 0.7518
Epoch 119/300
472/472 2s 4ms/step - accuracy: 0.8075 - loss: 0.4108 - val_accuracy: 0.7051 - val_loss: 0.7608
Epoch 120/300
472/472 2s 4ms/step - accuracy: 0.8058 - loss: 0.4115 - val_accuracy: 0.7011 - val_loss: 0.7611
Epoch 121/300
472/472 2s 4ms/step - accuracy: 0.8103 - loss: 0.4053 - val_accuracy: 0.7032 - val_loss: 0.7633
Epoch 122/300
472/472 2s 3ms/step - accuracy: 0.8079 - loss: 0.4104 - val_accuracy: 0.7023 - val_loss: 0.7771
Epoch 123/300
472/472 2s 3ms/step - accuracy: 0.8081 - loss: 0.4086 - val_accuracy: 0.6976 - val_loss: 0.7509
Epoch 124/300
472/472 2s 3ms/step - accuracy: 0.8076 - loss: 0.4101 - val_accuracy: 0.6918 - val_loss: 0.7698
Epoch 125/300
472/472 2s 3ms/step - accuracy: 0.8063 - loss: 0.4140 - val_accuracy: 0.7047 - val_loss: 0.7453
Epoch 126/300
472/472 2s 3ms/step - accuracy: 0.8093 - loss: 0.4096 - val_accuracy: 0.7034 - val_loss: 0.7481
Epoch 127/300
472/472 2s 3ms/step - accuracy: 0.8080 - loss: 0.4099 - val_accuracy: 0.7069 - val_loss:

s: 0.7329
Epoch 128/300
472/472 2s 3ms/step - accuracy: 0.8098 - loss: 0.4084 - val_accuracy: 0.7018 - val_loss: 0.7688
Epoch 129/300
472/472 2s 3ms/step - accuracy: 0.8082 - loss: 0.4099 - val_accuracy: 0.7066 - val_loss: 0.7503
Epoch 130/300
472/472 2s 3ms/step - accuracy: 0.8091 - loss: 0.4088 - val_accuracy: 0.6992 - val_loss: 0.7706
s: 0.7515
Epoch 131/300
472/472 2s 4ms/step - accuracy: 0.8086 - loss: 0.4094 - val_accuracy: 0.7047 - val_loss: 0.7611
s: 0.7581
Epoch 132/300
472/472 2s 3ms/step - accuracy: 0.8117 - loss: 0.4039 - val_accuracy: 0.7023 - val_loss: 0.7581
s: 0.7539
Epoch 133/300
472/472 2s 3ms/step - accuracy: 0.8117 - loss: 0.4052 - val_accuracy: 0.7041 - val_loss: 0.7739
s: 0.7571
Epoch 134/300
472/472 2s 3ms/step - accuracy: 0.8064 - loss: 0.4070 - val_accuracy: 0.7055 - val_loss: 0.7502
s: 0.7412
Epoch 135/300
472/472 2s 3ms/step - accuracy: 0.8087 - loss: 0.4075 - val_accuracy: 0.7038 - val_loss: 0.7412
s: 0.7619
Epoch 136/300
472/472 2s 3ms/step - accuracy: 0.8102 - loss: 0.4045 - val_accuracy: 0.7051 - val_loss: 0.7619
s: 0.7641
Epoch 137/300
472/472 2s 3ms/step - accuracy: 0.8128 - loss: 0.4021 - val_accuracy: 0.7028 - val_loss: 0.7641
s: 0.7572
Epoch 138/300
472/472 2s 4ms/step - accuracy: 0.8133 - loss: 0.3994 - val_accuracy: 0.6979 - val_loss: 0.7572
s: 0.7334
Epoch 139/300
472/472 2s 4ms/step - accuracy: 0.8106 - loss: 0.4041 - val_accuracy: 0.6978 - val_loss: 0.7334
s: 0.7720
Epoch 140/300
472/472 2s 4ms/step - accuracy: 0.8096 - loss: 0.4082 - val_accuracy: 0.7035 - val_loss: 0.7720
s: 0.7587
Epoch 141/300
472/472 2s 3ms/step - accuracy: 0.8136 - loss: 0.3992 - val_accuracy: 0.7052 - val_loss: 0.7459
s: 0.7425
Epoch 142/300
472/472 2s 4ms/step - accuracy: 0.8112 - loss: 0.4056 - val_accuracy: 0.6928 - val_loss: 0.7425
s: 0.7577
Epoch 143/300
472/472 2s 3ms/step - accuracy: 0.8102 - loss: 0.4006 - val_accuracy: 0.7055 - val_loss: 0.7577
s: 0.7524
Epoch 144/300
472/472 2s 3ms/step - accuracy: 0.8113 - loss: 0.4054 - val_accuracy: 0.7054 - val_loss: 0.7524
s: 0.7613
Epoch 145/300
472/472 2s 3ms/step - accuracy: 0.8132 - loss: 0.3985 - val_accuracy: 0.7079 - val_loss: 0.7613
s: 0.7506
Epoch 146/300
472/472 2s 3ms/step - accuracy: 0.8132 - loss: 0.3988 - val_accuracy: 0.7014 - val_loss: 0.7506
s: 0.7613
Epoch 147/300
472/472 2s 3ms/step - accuracy: 0.8113 - loss: 0.3992 - val_accuracy: 0.6996 - val_loss: 0.7613
s: 0.7524
Epoch 148/300
472/472 2s 4ms/step - accuracy: 0.8106 - loss: 0.4013 - val_accuracy: 0.7053 - val_loss: 0.7448
s: 0.7538
Epoch 149/300
472/472 2s 4ms/step - accuracy: 0.8113 - loss: 0.4019 - val_accuracy: 0.7072 - val_loss: 0.7538
s: 0.7506
Epoch 150/300
472/472 2s 3ms/step - accuracy: 0.8159 - loss: 0.3954 - val_accuracy: 0.7081 - val_loss: 0.7506
s: 0.7506

Epoch 153/300
472/472 2s 3ms/step - accuracy: 0.8135 - loss: 0.3992 - val_accuracy: 0.7000 - val_loss:
s: 0.7568
Epoch 154/300
472/472 2s 4ms/step - accuracy: 0.8142 - loss: 0.3955 - val_accuracy: 0.7065 - val_loss:
s: 0.7392
Epoch 155/300
472/472 2s 3ms/step - accuracy: 0.8132 - loss: 0.3975 - val_accuracy: 0.7044 - val_loss:
s: 0.7457
Epoch 156/300
472/472 2s 3ms/step - accuracy: 0.8114 - loss: 0.4014 - val_accuracy: 0.7092 - val_loss:
s: 0.7353
Epoch 157/300
472/472 2s 4ms/step - accuracy: 0.8122 - loss: 0.3981 - val_accuracy: 0.7044 - val_loss:
s: 0.7495
Epoch 158/300
472/472 2s 4ms/step - accuracy: 0.8153 - loss: 0.3947 - val_accuracy: 0.7076 - val_loss:
s: 0.7654
Epoch 159/300
472/472 2s 3ms/step - accuracy: 0.8121 - loss: 0.3992 - val_accuracy: 0.7068 - val_loss:
s: 0.7413
Epoch 160/300
472/472 2s 3ms/step - accuracy: 0.8130 - loss: 0.3985 - val_accuracy: 0.7075 - val_loss:
s: 0.7297
Epoch 161/300
472/472 2s 3ms/step - accuracy: 0.8149 - loss: 0.3971 - val_accuracy: 0.7090 - val_loss:
s: 0.7379
Epoch 162/300
472/472 2s 3ms/step - accuracy: 0.8142 - loss: 0.3970 - val_accuracy: 0.7127 - val_loss:
s: 0.7522
Epoch 163/300
472/472 2s 3ms/step - accuracy: 0.8134 - loss: 0.3966 - val_accuracy: 0.7119 - val_loss:
s: 0.7328
Epoch 164/300
472/472 2s 4ms/step - accuracy: 0.8149 - loss: 0.3954 - val_accuracy: 0.7023 - val_loss:
s: 0.7634
Epoch 165/300
472/472 2s 3ms/step - accuracy: 0.8185 - loss: 0.3895 - val_accuracy: 0.7157 - val_loss:
s: 0.7209
Epoch 166/300
472/472 2s 4ms/step - accuracy: 0.8146 - loss: 0.3978 - val_accuracy: 0.7038 - val_loss:
s: 0.7701
Epoch 167/300
472/472 2s 4ms/step - accuracy: 0.8162 - loss: 0.3932 - val_accuracy: 0.7068 - val_loss:
s: 0.7401
Epoch 168/300
472/472 2s 3ms/step - accuracy: 0.8115 - loss: 0.3996 - val_accuracy: 0.7153 - val_loss:
s: 0.7504
Epoch 169/300
472/472 2s 3ms/step - accuracy: 0.8178 - loss: 0.3918 - val_accuracy: 0.7143 - val_loss:
s: 0.7356
Epoch 170/300
472/472 2s 3ms/step - accuracy: 0.8186 - loss: 0.3934 - val_accuracy: 0.7104 - val_loss:
s: 0.7345
Epoch 171/300
472/472 2s 3ms/step - accuracy: 0.8159 - loss: 0.3922 - val_accuracy: 0.7018 - val_loss:
s: 0.7425
Epoch 172/300
472/472 2s 3ms/step - accuracy: 0.8135 - loss: 0.3951 - val_accuracy: 0.7108 - val_loss:
s: 0.7374
Epoch 173/300
472/472 2s 3ms/step - accuracy: 0.8148 - loss: 0.3916 - val_accuracy: 0.7113 - val_loss:
s: 0.7343
Epoch 174/300
472/472 2s 3ms/step - accuracy: 0.8183 - loss: 0.3922 - val_accuracy: 0.7066 - val_loss:
s: 0.7401
Epoch 175/300
472/472 2s 3ms/step - accuracy: 0.8180 - loss: 0.3924 - val_accuracy: 0.7076 - val_loss:
s: 0.7559
Epoch 176/300
472/472 2s 4ms/step - accuracy: 0.8162 - loss: 0.3897 - val_accuracy: 0.7081 - val_loss:
s: 0.7513
Epoch 177/300
472/472 2s 3ms/step - accuracy: 0.8180 - loss: 0.3917 - val_accuracy: 0.7125 - val_loss:
s: 0.7391
Epoch 178/300

472/472 2s 4ms/step - accuracy: 0.8142 - loss: 0.3979 - val_accuracy: 0.7068 - val_loss: 0.7436
Epoch 179/300
472/472 2s 3ms/step - accuracy: 0.8180 - loss: 0.3921 - val_accuracy: 0.7097 - val_loss: 0.7483
Epoch 180/300
472/472 2s 3ms/step - accuracy: 0.8180 - loss: 0.3894 - val_accuracy: 0.7150 - val_loss: 0.7253
Epoch 181/300
472/472 2s 3ms/step - accuracy: 0.8197 - loss: 0.3888 - val_accuracy: 0.7170 - val_loss: 0.7269
Epoch 182/300
472/472 2s 4ms/step - accuracy: 0.8206 - loss: 0.3885 - val_accuracy: 0.7103 - val_loss: 0.7472
Epoch 183/300
472/472 2s 3ms/step - accuracy: 0.8186 - loss: 0.3897 - val_accuracy: 0.7153 - val_loss: 0.7321
Epoch 184/300
472/472 2s 4ms/step - accuracy: 0.8170 - loss: 0.3910 - val_accuracy: 0.7103 - val_loss: 0.7368
Epoch 185/300
472/472 2s 3ms/step - accuracy: 0.8169 - loss: 0.3887 - val_accuracy: 0.7146 - val_loss: 0.7415
Epoch 186/300
472/472 2s 4ms/step - accuracy: 0.8193 - loss: 0.3877 - val_accuracy: 0.7155 - val_loss: 0.7347
Epoch 187/300
472/472 2s 4ms/step - accuracy: 0.8192 - loss: 0.3882 - val_accuracy: 0.7133 - val_loss: 0.7260
Epoch 188/300
472/472 2s 4ms/step - accuracy: 0.8165 - loss: 0.3922 - val_accuracy: 0.7130 - val_loss: 0.7372
Epoch 189/300
472/472 2s 4ms/step - accuracy: 0.8206 - loss: 0.3846 - val_accuracy: 0.7236 - val_loss: 0.7299
Epoch 190/300
472/472 2s 4ms/step - accuracy: 0.8182 - loss: 0.3888 - val_accuracy: 0.7182 - val_loss: 0.7321
Epoch 191/300
472/472 2s 4ms/step - accuracy: 0.8173 - loss: 0.3895 - val_accuracy: 0.7201 - val_loss: 0.7235
Epoch 192/300
472/472 2s 4ms/step - accuracy: 0.8196 - loss: 0.3868 - val_accuracy: 0.7094 - val_loss: 0.7334
Epoch 193/300
472/472 2s 4ms/step - accuracy: 0.8185 - loss: 0.3902 - val_accuracy: 0.7164 - val_loss: 0.7356
Epoch 194/300
472/472 2s 4ms/step - accuracy: 0.8213 - loss: 0.3834 - val_accuracy: 0.7140 - val_loss: 0.7438
Epoch 195/300
472/472 2s 4ms/step - accuracy: 0.8178 - loss: 0.3878 - val_accuracy: 0.7121 - val_loss: 0.7580
Epoch 196/300
472/472 2s 4ms/step - accuracy: 0.8180 - loss: 0.3877 - val_accuracy: 0.7125 - val_loss: 0.7349
Epoch 197/300
472/472 2s 4ms/step - accuracy: 0.8204 - loss: 0.3852 - val_accuracy: 0.7187 - val_loss: 0.7342
Epoch 198/300
472/472 2s 4ms/step - accuracy: 0.8194 - loss: 0.3881 - val_accuracy: 0.7181 - val_loss: 0.7152
Epoch 199/300
472/472 2s 4ms/step - accuracy: 0.8218 - loss: 0.3827 - val_accuracy: 0.7194 - val_loss: 0.7274
Epoch 200/300
472/472 2s 4ms/step - accuracy: 0.8219 - loss: 0.3838 - val_accuracy: 0.7216 - val_loss: 0.7169
Epoch 201/300
472/472 2s 4ms/step - accuracy: 0.8210 - loss: 0.3828 - val_accuracy: 0.7125 - val_loss: 0.7566
Epoch 202/300
472/472 2s 4ms/step - accuracy: 0.8203 - loss: 0.3847 - val_accuracy: 0.7100 - val_loss: 0.7623
Epoch 203/300
472/472 2s 4ms/step - accuracy: 0.8189 - loss: 0.3853 - val_accuracy: 0.7163 - val_loss:

s: 0.7374
Epoch 204/300
472/472 2s 4ms/step - accuracy: 0.8207 - loss: 0.3842 - val_accuracy: 0.7249 - val_loss: 0.7138
s: 0.7138
Epoch 205/300
472/472 2s 4ms/step - accuracy: 0.8213 - loss: 0.3837 - val_accuracy: 0.7155 - val_loss: 0.7456
s: 0.7456
Epoch 206/300
472/472 2s 4ms/step - accuracy: 0.8218 - loss: 0.3852 - val_accuracy: 0.7164 - val_loss: 0.7248
s: 0.7248
Epoch 207/300
472/472 2s 4ms/step - accuracy: 0.8224 - loss: 0.3809 - val_accuracy: 0.7178 - val_loss: 0.7477
s: 0.7477
Epoch 208/300
472/472 2s 4ms/step - accuracy: 0.8184 - loss: 0.3891 - val_accuracy: 0.7138 - val_loss: 0.7359
s: 0.7359
Epoch 209/300
472/472 2s 4ms/step - accuracy: 0.8252 - loss: 0.3790 - val_accuracy: 0.7170 - val_loss: 0.7311
s: 0.7311
Epoch 210/300
472/472 2s 4ms/step - accuracy: 0.8194 - loss: 0.3862 - val_accuracy: 0.7177 - val_loss: 0.7473
s: 0.7473
Epoch 211/300
472/472 2s 4ms/step - accuracy: 0.8223 - loss: 0.3817 - val_accuracy: 0.7158 - val_loss: 0.7325
s: 0.7325
Epoch 212/300
472/472 2s 4ms/step - accuracy: 0.8207 - loss: 0.3840 - val_accuracy: 0.7227 - val_loss: 0.7283
s: 0.7283
Epoch 213/300
472/472 2s 4ms/step - accuracy: 0.8231 - loss: 0.3794 - val_accuracy: 0.7229 - val_loss: 0.7186
s: 0.7186
Epoch 214/300
472/472 2s 4ms/step - accuracy: 0.8223 - loss: 0.3811 - val_accuracy: 0.7120 - val_loss: 0.7384
s: 0.7384
Epoch 215/300
472/472 2s 4ms/step - accuracy: 0.8203 - loss: 0.3797 - val_accuracy: 0.7155 - val_loss: 0.7250
s: 0.7250
Epoch 216/300
472/472 2s 4ms/step - accuracy: 0.8208 - loss: 0.3811 - val_accuracy: 0.7199 - val_loss: 0.7254
s: 0.7254
Epoch 217/300
472/472 2s 4ms/step - accuracy: 0.8214 - loss: 0.3823 - val_accuracy: 0.7192 - val_loss: 0.7384
s: 0.7384
Epoch 218/300
472/472 2s 4ms/step - accuracy: 0.8217 - loss: 0.3829 - val_accuracy: 0.7187 - val_loss: 0.7244
s: 0.7244
Epoch 219/300
472/472 2s 4ms/step - accuracy: 0.8211 - loss: 0.3856 - val_accuracy: 0.7226 - val_loss: 0.7087
s: 0.7087
Epoch 220/300
472/472 3s 4ms/step - accuracy: 0.8227 - loss: 0.3826 - val_accuracy: 0.7197 - val_loss: 0.7270
s: 0.7270
Epoch 221/300
472/472 2s 4ms/step - accuracy: 0.8223 - loss: 0.3815 - val_accuracy: 0.7206 - val_loss: 0.7267
s: 0.7267
Epoch 222/300
472/472 2s 4ms/step - accuracy: 0.8200 - loss: 0.3847 - val_accuracy: 0.7190 - val_loss: 0.7399
s: 0.7399
Epoch 223/300
472/472 2s 4ms/step - accuracy: 0.8203 - loss: 0.3838 - val_accuracy: 0.7153 - val_loss: 0.7431
s: 0.7431
Epoch 224/300
472/472 2s 4ms/step - accuracy: 0.8249 - loss: 0.3749 - val_accuracy: 0.7102 - val_loss: 0.7584
s: 0.7584
Epoch 225/300
472/472 2s 4ms/step - accuracy: 0.8215 - loss: 0.3805 - val_accuracy: 0.7272 - val_loss: 0.7125
s: 0.7125
Epoch 226/300
472/472 2s 4ms/step - accuracy: 0.8239 - loss: 0.3797 - val_accuracy: 0.7175 - val_loss: 0.7315
s: 0.7315
Epoch 227/300
472/472 2s 4ms/step - accuracy: 0.8213 - loss: 0.3779 - val_accuracy: 0.7154 - val_loss: 0.7332
s: 0.7332
Epoch 228/300
472/472 2s 4ms/step - accuracy: 0.8208 - loss: 0.3836 - val_accuracy: 0.7214 - val_loss: 0.7282
s: 0.7282

Epoch 229/300
472/472 2s 4ms/step - accuracy: 0.8239 - loss: 0.3806 - val_accuracy: 0.7143 - val_loss: 0.7560
Epoch 230/300
472/472 2s 4ms/step - accuracy: 0.8241 - loss: 0.3799 - val_accuracy: 0.7196 - val_loss: 0.7323
Epoch 231/300
472/472 2s 4ms/step - accuracy: 0.8249 - loss: 0.3764 - val_accuracy: 0.7223 - val_loss: 0.7487
s: 0.7333
Epoch 232/300
472/472 2s 4ms/step - accuracy: 0.8226 - loss: 0.3825 - val_accuracy: 0.7181 - val_loss: 0.7125
s: 0.7687
Epoch 233/300
472/472 2s 4ms/step - accuracy: 0.8263 - loss: 0.3767 - val_accuracy: 0.7217 - val_loss: 0.7373
s: 0.7398
Epoch 234/300
472/472 2s 4ms/step - accuracy: 0.8239 - loss: 0.3783 - val_accuracy: 0.7122 - val_loss: 0.7375
s: 0.7375
Epoch 235/300
472/472 2s 4ms/step - accuracy: 0.8231 - loss: 0.3799 - val_accuracy: 0.7194 - val_loss: 0.7375
s: 0.7398
Epoch 236/300
472/472 2s 4ms/step - accuracy: 0.8232 - loss: 0.3760 - val_accuracy: 0.7200 - val_loss: 0.7375
s: 0.7375
Epoch 237/300
472/472 2s 4ms/step - accuracy: 0.8237 - loss: 0.3783 - val_accuracy: 0.7180 - val_loss: 0.7375
s: 0.7375
Epoch 238/300
472/472 2s 4ms/step - accuracy: 0.8218 - loss: 0.3834 - val_accuracy: 0.7287 - val_loss: 0.6955
s: 0.7319
Epoch 239/300
472/472 2s 4ms/step - accuracy: 0.8230 - loss: 0.3816 - val_accuracy: 0.7203 - val_loss: 0.7260
s: 0.7260
Epoch 240/300
472/472 2s 4ms/step - accuracy: 0.8258 - loss: 0.3753 - val_accuracy: 0.7167 - val_loss: 0.7274
s: 0.7274
Epoch 241/300
472/472 2s 4ms/step - accuracy: 0.8255 - loss: 0.3732 - val_accuracy: 0.7214 - val_loss: 0.7113
s: 0.7113
Epoch 242/300
472/472 2s 4ms/step - accuracy: 0.8254 - loss: 0.3749 - val_accuracy: 0.7291 - val_loss: 0.7401
s: 0.7401
Epoch 243/300
472/472 2s 4ms/step - accuracy: 0.8238 - loss: 0.3768 - val_accuracy: 0.7140 - val_loss: 0.7457
s: 0.7457
Epoch 244/300
472/472 2s 4ms/step - accuracy: 0.8252 - loss: 0.3747 - val_accuracy: 0.7193 - val_loss: 0.7244
s: 0.7244
Epoch 245/300
472/472 2s 4ms/step - accuracy: 0.8248 - loss: 0.3784 - val_accuracy: 0.7228 - val_loss: 0.7457
s: 0.7457
Epoch 246/300
472/472 2s 4ms/step - accuracy: 0.8256 - loss: 0.3756 - val_accuracy: 0.7125 - val_loss: 0.7341
s: 0.7341
Epoch 247/300
472/472 2s 4ms/step - accuracy: 0.8254 - loss: 0.3709 - val_accuracy: 0.7150 - val_loss: 0.7352
s: 0.7352
Epoch 248/300
472/472 2s 4ms/step - accuracy: 0.8240 - loss: 0.3753 - val_accuracy: 0.7250 - val_loss: 0.7302
s: 0.7302
Epoch 249/300
472/472 2s 4ms/step - accuracy: 0.8250 - loss: 0.3775 - val_accuracy: 0.7197 - val_loss: 0.7246
s: 0.7246
Epoch 250/300
472/472 2s 4ms/step - accuracy: 0.8238 - loss: 0.3770 - val_accuracy: 0.7188 - val_loss: 0.7170
s: 0.7170
Epoch 251/300
472/472 2s 4ms/step - accuracy: 0.8235 - loss: 0.3744 - val_accuracy: 0.7254 - val_loss: 0.7252
s: 0.7252
Epoch 252/300
472/472 2s 4ms/step - accuracy: 0.8229 - loss: 0.3782 - val_accuracy: 0.7255 - val_loss: 0.7237
s: 0.7237
Epoch 253/300
472/472 2s 4ms/step - accuracy: 0.8261 - loss: 0.3755 - val_accuracy: 0.7227 - val_loss: 0.7237
s: 0.7237
Epoch 254/300

472/472 2s 4ms/step - accuracy: 0.8260 - loss: 0.3730 - val_accuracy: 0.7228 - val_loss: 0.7246
Epoch 255/300
472/472 2s 4ms/step - accuracy: 0.8253 - loss: 0.3756 - val_accuracy: 0.7180 - val_loss: 0.7241
Epoch 256/300
472/472 2s 4ms/step - accuracy: 0.8255 - loss: 0.3763 - val_accuracy: 0.7216 - val_loss: 0.7179
Epoch 257/300
472/472 2s 4ms/step - accuracy: 0.8269 - loss: 0.3772 - val_accuracy: 0.7197 - val_loss: 0.7344
Epoch 258/300
472/472 2s 4ms/step - accuracy: 0.8251 - loss: 0.3737 - val_accuracy: 0.7193 - val_loss: 0.7168
Epoch 259/300
472/472 2s 4ms/step - accuracy: 0.8282 - loss: 0.3712 - val_accuracy: 0.7167 - val_loss: 0.7262
Epoch 260/300
472/472 2s 4ms/step - accuracy: 0.8257 - loss: 0.3746 - val_accuracy: 0.7242 - val_loss: 0.7221
Epoch 261/300
472/472 2s 4ms/step - accuracy: 0.8263 - loss: 0.3751 - val_accuracy: 0.7240 - val_loss: 0.7207
Epoch 262/300
472/472 2s 4ms/step - accuracy: 0.8261 - loss: 0.3755 - val_accuracy: 0.7233 - val_loss: 0.7282
Epoch 263/300
472/472 2s 4ms/step - accuracy: 0.8266 - loss: 0.3743 - val_accuracy: 0.7217 - val_loss: 0.7180
Epoch 264/300
472/472 2s 4ms/step - accuracy: 0.8255 - loss: 0.3741 - val_accuracy: 0.7252 - val_loss: 0.7207
Epoch 265/300
472/472 2s 4ms/step - accuracy: 0.8259 - loss: 0.3718 - val_accuracy: 0.7207 - val_loss: 0.7287
Epoch 266/300
472/472 2s 4ms/step - accuracy: 0.8269 - loss: 0.3702 - val_accuracy: 0.7292 - val_loss: 0.7067
Epoch 267/300
472/472 2s 4ms/step - accuracy: 0.8266 - loss: 0.3760 - val_accuracy: 0.7278 - val_loss: 0.7078
Epoch 268/300
472/472 2s 4ms/step - accuracy: 0.8277 - loss: 0.3684 - val_accuracy: 0.7201 - val_loss: 0.7116
Epoch 269/300
472/472 2s 4ms/step - accuracy: 0.8275 - loss: 0.3687 - val_accuracy: 0.7249 - val_loss: 0.7018
Epoch 270/300
472/472 2s 5ms/step - accuracy: 0.8278 - loss: 0.3711 - val_accuracy: 0.7166 - val_loss: 0.7436
Epoch 271/300
472/472 2s 4ms/step - accuracy: 0.8307 - loss: 0.3687 - val_accuracy: 0.7214 - val_loss: 0.7234
Epoch 272/300
472/472 2s 4ms/step - accuracy: 0.8273 - loss: 0.3697 - val_accuracy: 0.7179 - val_loss: 0.7489
Epoch 273/300
472/472 2s 4ms/step - accuracy: 0.8236 - loss: 0.3766 - val_accuracy: 0.7282 - val_loss: 0.7070
Epoch 274/300
472/472 2s 4ms/step - accuracy: 0.8276 - loss: 0.3699 - val_accuracy: 0.7280 - val_loss: 0.6923
Epoch 275/300
472/472 2s 4ms/step - accuracy: 0.8251 - loss: 0.3753 - val_accuracy: 0.7331 - val_loss: 0.6987
Epoch 276/300
472/472 2s 4ms/step - accuracy: 0.8262 - loss: 0.3729 - val_accuracy: 0.7224 - val_loss: 0.7369
Epoch 277/300
472/472 2s 4ms/step - accuracy: 0.8276 - loss: 0.3725 - val_accuracy: 0.7329 - val_loss: 0.6910
Epoch 278/300
472/472 2s 4ms/step - accuracy: 0.8263 - loss: 0.3708 - val_accuracy: 0.7269 - val_loss: 0.7194
Epoch 279/300
472/472 2s 4ms/step - accuracy: 0.8277 - loss: 0.3708 - val_accuracy: 0.7312 - val_loss:

```
s: 0.6908
Epoch 280/300
472/472 2s 4ms/step - accuracy: 0.8276 - loss: 0.3712 - val_accuracy: 0.7262 - val_loss: 0.3664
s: 0.7154
Epoch 281/300
472/472 2s 4ms/step - accuracy: 0.8278 - loss: 0.3664 - val_accuracy: 0.7254 - val_loss: 0.3730
s: 0.7068
Epoch 282/300
472/472 2s 4ms/step - accuracy: 0.8283 - loss: 0.3730 - val_accuracy: 0.7193 - val_loss: 0.3706
s: 0.7242
Epoch 283/300
472/472 2s 4ms/step - accuracy: 0.8270 - loss: 0.3709 - val_accuracy: 0.7186 - val_loss: 0.3709
s: 0.7391
Epoch 284/300
472/472 2s 4ms/step - accuracy: 0.8272 - loss: 0.3709 - val_accuracy: 0.7292 - val_loss: 0.3716
s: 0.6953
Epoch 285/300
472/472 2s 4ms/step - accuracy: 0.8282 - loss: 0.3716 - val_accuracy: 0.7262 - val_loss: 0.3688
s: 0.7226
Epoch 286/300
472/472 2s 4ms/step - accuracy: 0.8290 - loss: 0.3688 - val_accuracy: 0.7261 - val_loss: 0.3687
s: 0.6947
Epoch 287/300
472/472 2s 4ms/step - accuracy: 0.8281 - loss: 0.3705 - val_accuracy: 0.7265 - val_loss: 0.3687
s: 0.7088
Epoch 288/300
472/472 2s 4ms/step - accuracy: 0.8270 - loss: 0.3687 - val_accuracy: 0.7281 - val_loss: 0.3698
s: 0.7011
Epoch 289/300
472/472 2s 4ms/step - accuracy: 0.8270 - loss: 0.3688 - val_accuracy: 0.7270 - val_loss: 0.3699
s: 0.7068
Epoch 290/300
472/472 2s 4ms/step - accuracy: 0.8281 - loss: 0.3698 - val_accuracy: 0.7270 - val_loss: 0.3722
s: 0.7131
Epoch 291/300
472/472 2s 4ms/step - accuracy: 0.8292 - loss: 0.3699 - val_accuracy: 0.7209 - val_loss: 0.3722
s: 0.7404
Epoch 292/300
472/472 2s 4ms/step - accuracy: 0.8267 - loss: 0.3722 - val_accuracy: 0.7246 - val_loss: 0.3709
s: 0.7089
Epoch 293/300
472/472 2s 4ms/step - accuracy: 0.8282 - loss: 0.3688 - val_accuracy: 0.7188 - val_loss: 0.3661
s: 0.7420
Epoch 294/300
472/472 2s 4ms/step - accuracy: 0.8280 - loss: 0.3692 - val_accuracy: 0.7325 - val_loss: 0.3714
s: 0.7140
Epoch 295/300
472/472 2s 4ms/step - accuracy: 0.8309 - loss: 0.3661 - val_accuracy: 0.7275 - val_loss: 0.3663
s: 0.7140
Epoch 296/300
472/472 2s 4ms/step - accuracy: 0.8280 - loss: 0.3663 - val_accuracy: 0.7193 - val_loss: 0.3674
s: 0.7321
Epoch 297/300
472/472 2s 4ms/step - accuracy: 0.8284 - loss: 0.3674 - val_accuracy: 0.7260 - val_loss: 0.3667
s: 0.6994
Epoch 298/300
472/472 2s 4ms/step - accuracy: 0.8281 - loss: 0.3667 - val_accuracy: 0.7323 - val_loss: 0.3646
s: 0.7013
Epoch 299/300
472/472 2s 4ms/step - accuracy: 0.8317 - loss: 0.3646 - val_accuracy: 0.7125 - val_loss: 0.3681
s: 0.7465
Epoch 300/300
472/472 2s 4ms/step - accuracy: 0.8295 - loss: 0.3681 - val_accuracy: 0.7268 - val_loss: 0.3721
```

```
In [232...]: ann2_smote.evaluate(X_train_smote, y_train_smote)
```

```
3770/3770 3s 826us/step - accuracy: 0.8261 - loss: 0.4294
```

```
Out[232...]: [0.31491026282310486, 0.8796206712722778]
```

```
In [233...]: ann2_smote.summary()
```

```
Model: "sequential_4"
```

Layer (type)	Output Shape	Param #
dense_24 (Dense)	(None, 512)	23,040
dropout_20 (Dropout)	(None, 512)	0
dense_25 (Dense)	(None, 256)	131,328
dropout_21 (Dropout)	(None, 256)	0
dense_26 (Dense)	(None, 128)	32,896
dropout_22 (Dropout)	(None, 128)	0
dense_27 (Dense)	(None, 64)	8,256
dropout_23 (Dropout)	(None, 64)	0
dense_28 (Dense)	(None, 3)	195

Total params: 587,147 (2.24 MB)

Trainable params: 195,715 (764.51 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 391,432 (1.49 MB)

```
In [234]: eval_metric(ann2_smote, X_train_smote, y_train_smote, X_val, y_val)
```

```
3770/3770 - 3s 844us/step
591/591 - 0s 788us/step
```

Test Set:

```
[[4836 462 210]
 [2070 6054 1929]
 [ 85 290 2967]]
```

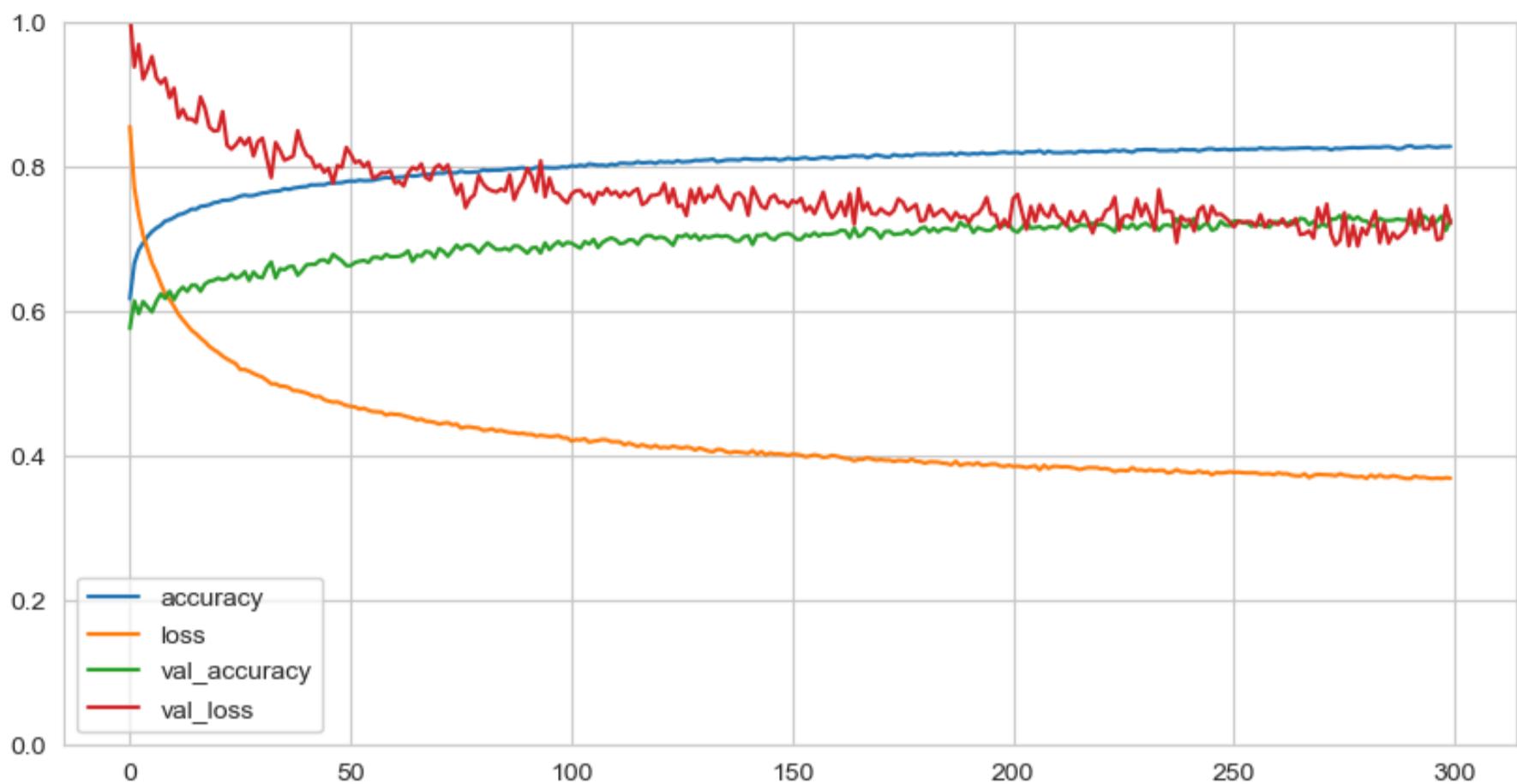
	precision	recall	f1-score	support
0	0.69	0.88	0.77	5508
1	0.89	0.60	0.72	10053
2	0.58	0.89	0.70	3342
accuracy			0.73	18903
macro avg	0.72	0.79	0.73	18903
weighted avg	0.78	0.73	0.73	18903

Train Set:

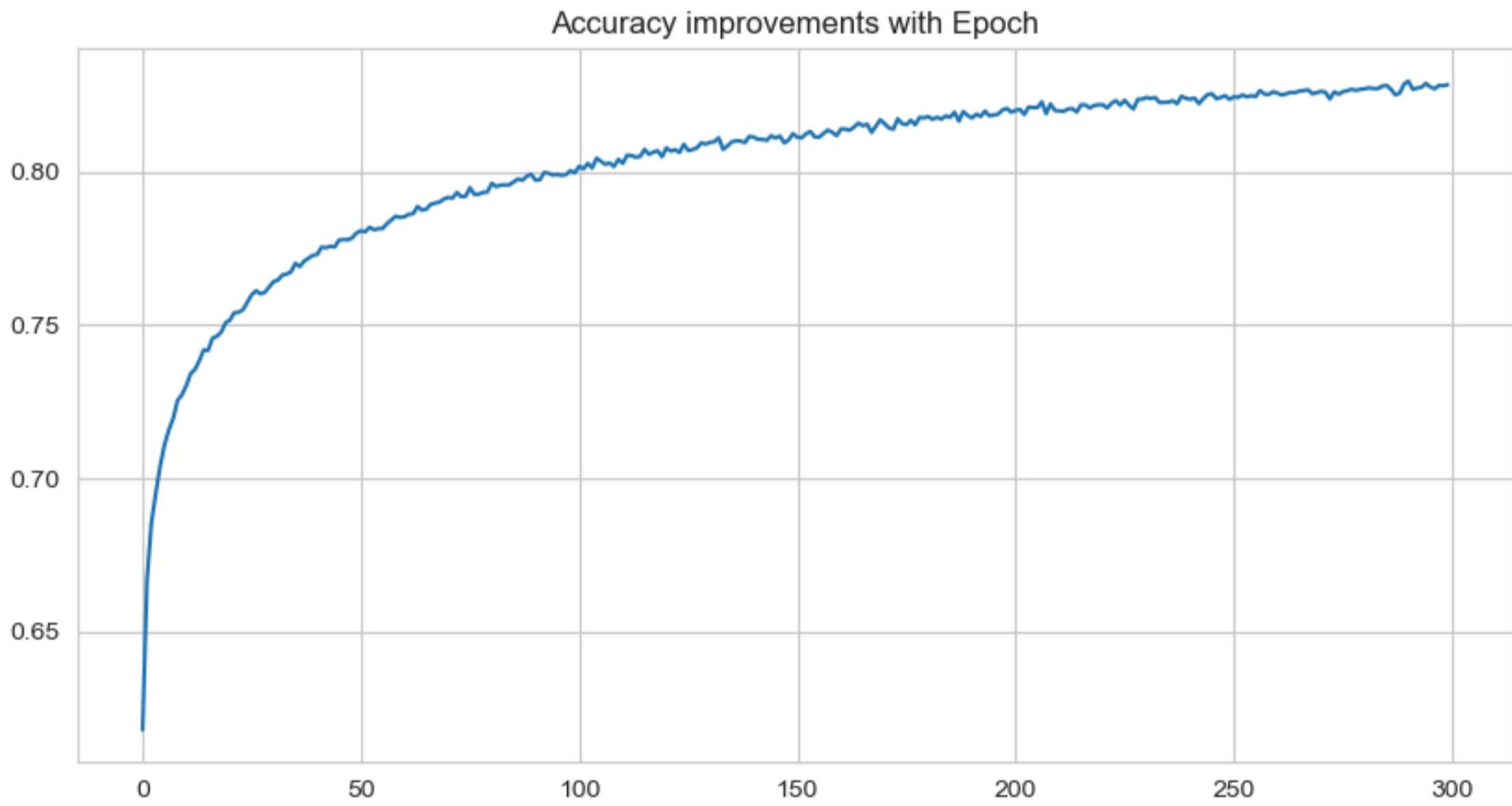
```
[[39205 648 356]
 [ 6859 26736 6614]
 [ 7 37 40165]]
```

	precision	recall	f1-score	support
0	0.85	0.98	0.91	40209
1	0.98	0.66	0.79	40209
2	0.85	1.00	0.92	40209
accuracy			0.88	120627
macro avg	0.89	0.88	0.87	120627
weighted avg	0.89	0.88	0.87	120627

```
In [235]: pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



```
In [236]: pd.DataFrame(history.history)[["accuracy"]].plot(figsize=(10, 5))
plt.title("Accuracy improvements with Epoch")
plt.show()
```



```
In [237]: # Save the Model

from tensorflow.keras.models import load_model

# Save the model to 'model.h5' file
ann2_smote.save('ann2_smote_model.h5')

# To Load the model Later for further use
loaded_ann2_smote = load_model('ann2_smote_model.h5')
```

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)` . This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')` .
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.

ANN-2 Smote Model Summary:

- **(Dense) layers:** 5 / **Neurons:** 512-256-128-64 / **Dropout:** 40-30-25-25% / **Learning Rate:** 0.001 / **Batch Size:** 256 / **Epochs:** 300 / **Early Stop (val_accuracy):** 50

- **Accuracy:** 0.88 / **Val_Accuracy:** 0.73 / **Loss:** 0.31 / **Val_Loss:** 0.72 / **Train Recall(Class 2):** 1.00 / **Test Recall (Class 2):** 0.89

The application of SMOTE in the ANN-2 model has significantly improved the recall for class 2, indicating better handling of the minority class. The increase in neurons and adjustments in the dropout rates have enhanced model capacity and regularization, helping mitigate overfitting to some extent.

However, the visible gap between training and validation performance suggests potential overfitting.

Future adjustments might include further tuning of dropout rates or exploring alternative regularization techniques to enhance generalization and reduce the validation loss without compromising the recall gains observed.

ANN-3 Model + Dense +Epoch (%80)

In [221...]

```
# 1) Model Architecture:  
  
# Increased Fully connected (Dense) from 5 to ----> 7 Layers  
# Increased Neurons number from 256 to ----> 512  
ann3 = Sequential([  
    Dense(512, input_dim=X_train.shape[1], activation='relu'),  
    Dropout(0.3),  
    Dense(256, activation='relu'),  
    Dropout(0.3),  
    Dense(128, activation='relu'),  
    Dropout(0.25),  
    Dense(128, activation='relu'),  
    Dropout(0.25),  
    Dense(128, activation='relu'),  
    Dropout(0.2),  
    Dense(64, activation='relu'),  
    Dropout(0.2),  
    Dense(3, activation='softmax')  
])  
  
# 2) Compiling the Model:  
ann3.compile(optimizer=Adam(learning_rate=0.001),  
              loss='sparse_categorical_crossentropy',  
              metrics=['accuracy'])  
  
# 3) Early stopping  
early_stopping = EarlyStopping(monitor='val_accuracy',  
                               patience=60, # Increased from 50 to --> 60  
                               mode="auto",  
                               restore_best_weights=True)  
  
# 4) Train the model  
history = ann3.fit(  
    x=X_train,  
    y=y_train,  
    validation_data=(X_val, y_val),  
    batch_size=256, # same  
    epochs=400,      # Increased Epoch from 300 to --> 400  
    verbose=1,  
    callbacks=[early_stopping])
```

Epoch 1/400
296/296 5s 6ms/step - accuracy: 0.5553 - loss: 0.8844 - val_accuracy: 0.6394 - val_loss: 0.7558
Epoch 2/400
296/296 1s 5ms/step - accuracy: 0.6280 - loss: 0.7690 - val_accuracy: 0.6583 - val_loss: 0.7313
Epoch 3/400
296/296 1s 5ms/step - accuracy: 0.6484 - loss: 0.7425 - val_accuracy: 0.6708 - val_loss: 0.7181
Epoch 4/400
296/296 2s 5ms/step - accuracy: 0.6614 - loss: 0.7275 - val_accuracy: 0.6664 - val_loss: 0.7206
Epoch 5/400
296/296 1s 5ms/step - accuracy: 0.6645 - loss: 0.7243 - val_accuracy: 0.6766 - val_loss: 0.7081
Epoch 6/400
296/296 2s 5ms/step - accuracy: 0.6718 - loss: 0.7136 - val_accuracy: 0.6817 - val_loss: 0.7061
Epoch 7/400
296/296 1s 5ms/step - accuracy: 0.6770 - loss: 0.7029 - val_accuracy: 0.6839 - val_loss: 0.7019
Epoch 8/400
296/296 2s 6ms/step - accuracy: 0.6800 - loss: 0.7006 - val_accuracy: 0.6836 - val_loss: 0.6993
Epoch 9/400
296/296 1s 5ms/step - accuracy: 0.6828 - loss: 0.7013 - val_accuracy: 0.6872 - val_loss: 0.6948
Epoch 10/400
296/296 2s 5ms/step - accuracy: 0.6889 - loss: 0.6910 - val_accuracy: 0.6901 - val_loss: 0.6901
Epoch 11/400
296/296 2s 5ms/step - accuracy: 0.6900 - loss: 0.6881 - val_accuracy: 0.6900 - val_loss: 0.6873
Epoch 12/400
296/296 1s 5ms/step - accuracy: 0.6931 - loss: 0.6811 - val_accuracy: 0.6868 - val_loss: 0.6903
Epoch 13/400
296/296 1s 5ms/step - accuracy: 0.6987 - loss: 0.6733 - val_accuracy: 0.6927 - val_loss: 0.6809
Epoch 14/400
296/296 1s 5ms/step - accuracy: 0.6982 - loss: 0.6735 - val_accuracy: 0.6956 - val_loss: 0.6788
Epoch 15/400
296/296 2s 5ms/step - accuracy: 0.6999 - loss: 0.6682 - val_accuracy: 0.6953 - val_loss: 0.6752
Epoch 16/400
296/296 2s 6ms/step - accuracy: 0.7009 - loss: 0.6659 - val_accuracy: 0.6997 - val_loss: 0.6727
Epoch 17/400
296/296 2s 5ms/step - accuracy: 0.7053 - loss: 0.6573 - val_accuracy: 0.7013 - val_loss: 0.6675
Epoch 18/400
296/296 1s 5ms/step - accuracy: 0.7054 - loss: 0.6557 - val_accuracy: 0.7018 - val_loss: 0.6691
Epoch 19/400
296/296 2s 5ms/step - accuracy: 0.7113 - loss: 0.6495 - val_accuracy: 0.7040 - val_loss: 0.6657
Epoch 20/400
296/296 1s 5ms/step - accuracy: 0.7125 - loss: 0.6475 - val_accuracy: 0.7023 - val_loss: 0.6637
Epoch 21/400
296/296 2s 5ms/step - accuracy: 0.7136 - loss: 0.6395 - val_accuracy: 0.7051 - val_loss: 0.6589
Epoch 22/400
296/296 1s 5ms/step - accuracy: 0.7150 - loss: 0.6358 - val_accuracy: 0.7069 - val_loss: 0.6601
Epoch 23/400
296/296 2s 5ms/step - accuracy: 0.7151 - loss: 0.6359 - val_accuracy: 0.7099 - val_loss: 0.6539
Epoch 24/400
296/296 1s 5ms/step - accuracy: 0.7233 - loss: 0.6242 - val_accuracy: 0.7107 - val_loss: 0.6527
Epoch 25/400
296/296 1s 5ms/step - accuracy: 0.7259 - loss: 0.6228 - val_accuracy: 0.7106 - val_loss: 0.6491
Epoch 26/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7230 - loss: 0.6234 - val_accuracy: 0.7129 - val_loss: 0.6464
Epoch 27/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7247 - loss: 0.6197 - val_accuracy: 0.7142 - val_loss: 0.6450
Epoch 28/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7260 - loss: 0.6152 - val_accuracy: 0.7151 - val_loss: 0.6421
Epoch 29/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7309 - loss: 0.6087 - val_accuracy: 0.7178 - val_loss: 0.6390
Epoch 30/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7298 - loss: 0.6051 - val_accuracy: 0.7175 - val_loss: 0.6411
Epoch 31/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7345 - loss: 0.6049 - val_accuracy: 0.7195 - val_loss: 0.6361
Epoch 32/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7335 - loss: 0.6035 - val_accuracy: 0.7223 - val_loss: 0.6345
Epoch 33/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7370 - loss: 0.5948 - val_accuracy: 0.7164 - val_loss: 0.6338
Epoch 34/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7339 - loss: 0.6003 - val_accuracy: 0.7222 - val_loss: 0.6388
Epoch 35/400

296/296 ━━━━━━━━ 2s 5ms/step - accuracy: 0.7392 - loss: 0.5890 - val_accuracy: 0.7235 - val_loss: 0.6331
Epoch 36/400

296/296 ━━━━━━━━ 2s 5ms/step - accuracy: 0.7397 - loss: 0.5872 - val_accuracy: 0.7252 - val_loss: 0.6314
Epoch 37/400

296/296 ━━━━━━━━ 2s 5ms/step - accuracy: 0.7399 - loss: 0.5842 - val_accuracy: 0.7258 - val_loss: 0.6254
Epoch 38/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7418 - loss: 0.5833 - val_accuracy: 0.7256 - val_loss: 0.6253
Epoch 39/400

296/296 ━━━━━━━━ 1s 4ms/step - accuracy: 0.7411 - loss: 0.5810 - val_accuracy: 0.7280 - val_loss: 0.6207
Epoch 40/400

296/296 ━━━━━━━━ 2s 5ms/step - accuracy: 0.7466 - loss: 0.5728 - val_accuracy: 0.7289 - val_loss: 0.6231
Epoch 41/400

296/296 ━━━━━━━━ 2s 5ms/step - accuracy: 0.7444 - loss: 0.5767 - val_accuracy: 0.7296 - val_loss: 0.6192
Epoch 42/400

296/296 ━━━━━━━━ 2s 5ms/step - accuracy: 0.7483 - loss: 0.5689 - val_accuracy: 0.7290 - val_loss: 0.6228
Epoch 43/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7471 - loss: 0.5689 - val_accuracy: 0.7286 - val_loss: 0.6222
Epoch 44/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7502 - loss: 0.5681 - val_accuracy: 0.7304 - val_loss: 0.6140
Epoch 45/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7522 - loss: 0.5620 - val_accuracy: 0.7319 - val_loss: 0.6163
Epoch 46/400

296/296 ━━━━━━━━ 2s 5ms/step - accuracy: 0.7521 - loss: 0.5626 - val_accuracy: 0.7310 - val_loss: 0.6150
Epoch 47/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7508 - loss: 0.5631 - val_accuracy: 0.7354 - val_loss: 0.6099
Epoch 48/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7554 - loss: 0.5573 - val_accuracy: 0.7354 - val_loss: 0.6143
Epoch 49/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7573 - loss: 0.5525 - val_accuracy: 0.7320 - val_loss: 0.6176
Epoch 50/400

296/296 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7546 - loss: 0.5560 - val_accuracy: 0.7392 - val_loss: 0.6120
Epoch 51/400

296/296 ━━━━━━━━ 2s 5ms/step - accuracy: 0.7574 - loss: 0.5536 - val_accuracy: 0.7371 - val_loss:

```
s: 0.6069
Epoch 52/400
296/296 1s 5ms/step - accuracy: 0.7586 - loss: 0.5501 - val_accuracy: 0.7364 - val_loss: 0.5501
s: 0.6110
Epoch 53/400
296/296 1s 5ms/step - accuracy: 0.7554 - loss: 0.5532 - val_accuracy: 0.7348 - val_loss: 0.5532
s: 0.6081
Epoch 54/400
296/296 1s 5ms/step - accuracy: 0.7626 - loss: 0.5440 - val_accuracy: 0.7382 - val_loss: 0.5440
s: 0.6030
Epoch 55/400
296/296 1s 5ms/step - accuracy: 0.7596 - loss: 0.5470 - val_accuracy: 0.7416 - val_loss: 0.5470
s: 0.6070
Epoch 56/400
296/296 1s 5ms/step - accuracy: 0.7630 - loss: 0.5385 - val_accuracy: 0.7394 - val_loss: 0.5385
s: 0.6056
Epoch 57/400
296/296 1s 5ms/step - accuracy: 0.7604 - loss: 0.5447 - val_accuracy: 0.7397 - val_loss: 0.5447
s: 0.6027
Epoch 58/400
296/296 1s 5ms/step - accuracy: 0.7630 - loss: 0.5443 - val_accuracy: 0.7391 - val_loss: 0.5443
s: 0.6110
Epoch 59/400
296/296 1s 5ms/step - accuracy: 0.7656 - loss: 0.5364 - val_accuracy: 0.7394 - val_loss: 0.5364
s: 0.6048
Epoch 60/400
296/296 1s 5ms/step - accuracy: 0.7644 - loss: 0.5391 - val_accuracy: 0.7417 - val_loss: 0.5391
s: 0.6058
Epoch 61/400
296/296 1s 5ms/step - accuracy: 0.7652 - loss: 0.5333 - val_accuracy: 0.7434 - val_loss: 0.5333
s: 0.6035
Epoch 62/400
296/296 1s 5ms/step - accuracy: 0.7655 - loss: 0.5382 - val_accuracy: 0.7415 - val_loss: 0.5382
s: 0.6039
Epoch 63/400
296/296 1s 5ms/step - accuracy: 0.7672 - loss: 0.5297 - val_accuracy: 0.7407 - val_loss: 0.5297
s: 0.5968
Epoch 64/400
296/296 1s 5ms/step - accuracy: 0.7666 - loss: 0.5329 - val_accuracy: 0.7444 - val_loss: 0.5329
s: 0.6042
Epoch 65/400
296/296 1s 5ms/step - accuracy: 0.7663 - loss: 0.5337 - val_accuracy: 0.7435 - val_loss: 0.5337
s: 0.6017
Epoch 66/400
296/296 1s 5ms/step - accuracy: 0.7704 - loss: 0.5249 - val_accuracy: 0.7464 - val_loss: 0.5249
s: 0.6034
Epoch 67/400
296/296 1s 5ms/step - accuracy: 0.7698 - loss: 0.5251 - val_accuracy: 0.7447 - val_loss: 0.5251
s: 0.5969
Epoch 68/400
296/296 1s 5ms/step - accuracy: 0.7724 - loss: 0.5232 - val_accuracy: 0.7461 - val_loss: 0.5232
s: 0.5901
Epoch 69/400
296/296 1s 5ms/step - accuracy: 0.7713 - loss: 0.5232 - val_accuracy: 0.7444 - val_loss: 0.5232
s: 0.5937
Epoch 70/400
296/296 1s 5ms/step - accuracy: 0.7720 - loss: 0.5213 - val_accuracy: 0.7440 - val_loss: 0.5213
s: 0.5953
Epoch 71/400
296/296 1s 5ms/step - accuracy: 0.7730 - loss: 0.5251 - val_accuracy: 0.7463 - val_loss: 0.5251
s: 0.5964
Epoch 72/400
296/296 1s 5ms/step - accuracy: 0.7733 - loss: 0.5190 - val_accuracy: 0.7477 - val_loss: 0.5190
s: 0.5946
Epoch 73/400
296/296 1s 5ms/step - accuracy: 0.7728 - loss: 0.5149 - val_accuracy: 0.7487 - val_loss: 0.5149
s: 0.5895
Epoch 74/400
296/296 1s 5ms/step - accuracy: 0.7742 - loss: 0.5140 - val_accuracy: 0.7463 - val_loss: 0.5140
s: 0.5929
Epoch 75/400
296/296 1s 5ms/step - accuracy: 0.7755 - loss: 0.5123 - val_accuracy: 0.7468 - val_loss: 0.5123
s: 0.5936
Epoch 76/400
296/296 1s 5ms/step - accuracy: 0.7781 - loss: 0.5135 - val_accuracy: 0.7447 - val_loss: 0.5135
s: 0.6016
```

Epoch 77/400
296/296 1s 5ms/step - accuracy: 0.7760 - loss: 0.5158 - val_accuracy: 0.7482 - val_loss: 0.5914

Epoch 78/400
296/296 1s 5ms/step - accuracy: 0.7783 - loss: 0.5121 - val_accuracy: 0.7476 - val_loss: 0.5859

Epoch 79/400
296/296 1s 5ms/step - accuracy: 0.7768 - loss: 0.5134 - val_accuracy: 0.7473 - val_loss: 0.5893

Epoch 80/400
296/296 1s 5ms/step - accuracy: 0.7769 - loss: 0.5099 - val_accuracy: 0.7475 - val_loss: 0.5827

Epoch 81/400
296/296 1s 5ms/step - accuracy: 0.7765 - loss: 0.5113 - val_accuracy: 0.7463 - val_loss: 0.5900

Epoch 82/400
296/296 2s 5ms/step - accuracy: 0.7789 - loss: 0.5093 - val_accuracy: 0.7477 - val_loss: 0.5936

Epoch 83/400
296/296 2s 5ms/step - accuracy: 0.7829 - loss: 0.5001 - val_accuracy: 0.7485 - val_loss: 0.5925

Epoch 84/400
296/296 1s 5ms/step - accuracy: 0.7806 - loss: 0.5064 - val_accuracy: 0.7509 - val_loss: 0.5915

Epoch 85/400
296/296 1s 5ms/step - accuracy: 0.7792 - loss: 0.5056 - val_accuracy: 0.7514 - val_loss: 0.5904

Epoch 86/400
296/296 1s 5ms/step - accuracy: 0.7789 - loss: 0.5055 - val_accuracy: 0.7506 - val_loss: 0.5866

Epoch 87/400
296/296 1s 5ms/step - accuracy: 0.7859 - loss: 0.4965 - val_accuracy: 0.7522 - val_loss: 0.5898

Epoch 88/400
296/296 1s 5ms/step - accuracy: 0.7832 - loss: 0.4973 - val_accuracy: 0.7553 - val_loss: 0.5861

Epoch 89/400
296/296 1s 5ms/step - accuracy: 0.7832 - loss: 0.5007 - val_accuracy: 0.7483 - val_loss: 0.5881

Epoch 90/400
296/296 1s 5ms/step - accuracy: 0.7829 - loss: 0.4998 - val_accuracy: 0.7513 - val_loss: 0.5974

Epoch 91/400
296/296 2s 5ms/step - accuracy: 0.7865 - loss: 0.4963 - val_accuracy: 0.7526 - val_loss: 0.5867

Epoch 92/400
296/296 1s 5ms/step - accuracy: 0.7847 - loss: 0.4961 - val_accuracy: 0.7511 - val_loss: 0.5921

Epoch 93/400
296/296 1s 5ms/step - accuracy: 0.7860 - loss: 0.4961 - val_accuracy: 0.7492 - val_loss: 0.5880

Epoch 94/400
296/296 1s 5ms/step - accuracy: 0.7818 - loss: 0.5013 - val_accuracy: 0.7494 - val_loss: 0.5916

Epoch 95/400
296/296 1s 5ms/step - accuracy: 0.7869 - loss: 0.4957 - val_accuracy: 0.7497 - val_loss: 0.5908

Epoch 96/400
296/296 1s 5ms/step - accuracy: 0.7861 - loss: 0.4921 - val_accuracy: 0.7510 - val_loss: 0.5876

Epoch 97/400
296/296 1s 5ms/step - accuracy: 0.7879 - loss: 0.4933 - val_accuracy: 0.7545 - val_loss: 0.5842

Epoch 98/400
296/296 1s 5ms/step - accuracy: 0.7881 - loss: 0.4906 - val_accuracy: 0.7538 - val_loss: 0.5824

Epoch 99/400
296/296 2s 5ms/step - accuracy: 0.7866 - loss: 0.4907 - val_accuracy: 0.7540 - val_loss: 0.5814

Epoch 100/400
296/296 1s 5ms/step - accuracy: 0.7899 - loss: 0.4857 - val_accuracy: 0.7561 - val_loss: 0.5863

Epoch 101/400
296/296 1s 4ms/step - accuracy: 0.7871 - loss: 0.4891 - val_accuracy: 0.7545 - val_loss: 0.5880

Epoch 102/400

296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7861 - loss: 0.4943 - val_accuracy: 0.7523 - val_loss: 0.5855
Epoch 103/400
296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7903 - loss: 0.4860 - val_accuracy: 0.7527 - val_loss: 0.5885
Epoch 104/400
296/296 ━━━━━━ 2s 5ms/step - accuracy: 0.7891 - loss: 0.4881 - val_accuracy: 0.7563 - val_loss: 0.5879
Epoch 105/400
296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7920 - loss: 0.4858 - val_accuracy: 0.7542 - val_loss: 0.5924
Epoch 106/400
296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7902 - loss: 0.4910 - val_accuracy: 0.7542 - val_loss: 0.5829
Epoch 107/400
296/296 ━━━━━━ 1s 4ms/step - accuracy: 0.7912 - loss: 0.4832 - val_accuracy: 0.7560 - val_loss: 0.5843
Epoch 108/400
296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7941 - loss: 0.4820 - val_accuracy: 0.7581 - val_loss: 0.5838
Epoch 109/400
296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7924 - loss: 0.4854 - val_accuracy: 0.7601 - val_loss: 0.5831
Epoch 110/400
296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7904 - loss: 0.4851 - val_accuracy: 0.7555 - val_loss: 0.5819
Epoch 111/400
296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7904 - loss: 0.4862 - val_accuracy: 0.7536 - val_loss: 0.5795
Epoch 112/400
296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7919 - loss: 0.4805 - val_accuracy: 0.7579 - val_loss: 0.5847
Epoch 113/400
296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7942 - loss: 0.4769 - val_accuracy: 0.7542 - val_loss: 0.5902
Epoch 114/400
296/296 ━━━━━━ 2s 5ms/step - accuracy: 0.7898 - loss: 0.4836 - val_accuracy: 0.7568 - val_loss: 0.5834
Epoch 115/400
296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7934 - loss: 0.4798 - val_accuracy: 0.7536 - val_loss: 0.5809
Epoch 116/400
296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7948 - loss: 0.4777 - val_accuracy: 0.7538 - val_loss: 0.5940
Epoch 117/400
296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7987 - loss: 0.4732 - val_accuracy: 0.7546 - val_loss: 0.5772
Epoch 118/400
296/296 ━━━━━━ 2s 5ms/step - accuracy: 0.7967 - loss: 0.4768 - val_accuracy: 0.7563 - val_loss: 0.5820
Epoch 119/400
296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7945 - loss: 0.4803 - val_accuracy: 0.7565 - val_loss: 0.5804
Epoch 120/400
296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7929 - loss: 0.4764 - val_accuracy: 0.7570 - val_loss: 0.5807
Epoch 121/400
296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7959 - loss: 0.4736 - val_accuracy: 0.7550 - val_loss: 0.5830
Epoch 122/400
296/296 ━━━━━━ 2s 5ms/step - accuracy: 0.7968 - loss: 0.4742 - val_accuracy: 0.7589 - val_loss: 0.5809
Epoch 123/400
296/296 ━━━━━━ 2s 5ms/step - accuracy: 0.7966 - loss: 0.4747 - val_accuracy: 0.7531 - val_loss: 0.5884
Epoch 124/400
296/296 ━━━━━━ 2s 6ms/step - accuracy: 0.7954 - loss: 0.4777 - val_accuracy: 0.7599 - val_loss: 0.5751
Epoch 125/400
296/296 ━━━━━━ 2s 6ms/step - accuracy: 0.7947 - loss: 0.4781 - val_accuracy: 0.7558 - val_loss: 0.5791
Epoch 126/400
296/296 ━━━━━━ 2s 5ms/step - accuracy: 0.7986 - loss: 0.4681 - val_accuracy: 0.7563 - val_loss: 0.5846
Epoch 127/400
296/296 ━━━━━━ 1s 5ms/step - accuracy: 0.7945 - loss: 0.4742 - val_accuracy: 0.7568 - val_loss:

s: 0.5836
Epoch 128/400
296/296 1s 5ms/step - accuracy: 0.8003 - loss: 0.4650 - val_accuracy: 0.7565 - val_loss: 0.5851
Epoch 129/400
296/296 2s 6ms/step - accuracy: 0.7955 - loss: 0.4732 - val_accuracy: 0.7561 - val_loss: 0.5807
Epoch 130/400
296/296 2s 5ms/step - accuracy: 0.7994 - loss: 0.4680 - val_accuracy: 0.7577 - val_loss: 0.5857
s: 0.5831
Epoch 131/400
296/296 2s 5ms/step - accuracy: 0.8018 - loss: 0.4649 - val_accuracy: 0.7604 - val_loss: 0.5865
s: 0.5814
Epoch 132/400
296/296 2s 7ms/step - accuracy: 0.7991 - loss: 0.4684 - val_accuracy: 0.7592 - val_loss: 0.5814
s: 0.5866
Epoch 133/400
296/296 2s 5ms/step - accuracy: 0.7968 - loss: 0.4699 - val_accuracy: 0.7588 - val_loss: 0.5798
s: 0.5811
Epoch 134/400
296/296 1s 5ms/step - accuracy: 0.7981 - loss: 0.4699 - val_accuracy: 0.7589 - val_loss: 0.5866
s: 0.5811
Epoch 135/400
296/296 2s 6ms/step - accuracy: 0.7962 - loss: 0.4737 - val_accuracy: 0.7570 - val_loss: 0.5914
s: 0.5811
Epoch 136/400
296/296 2s 5ms/step - accuracy: 0.8006 - loss: 0.4632 - val_accuracy: 0.7578 - val_loss: 0.5833
s: 0.5801
Epoch 137/400
296/296 2s 5ms/step - accuracy: 0.8011 - loss: 0.4658 - val_accuracy: 0.7576 - val_loss: 0.5774
s: 0.5833
Epoch 138/400
296/296 1s 5ms/step - accuracy: 0.8007 - loss: 0.4633 - val_accuracy: 0.7621 - val_loss: 0.5774
s: 0.5875
Epoch 139/400
296/296 1s 4ms/step - accuracy: 0.7994 - loss: 0.4674 - val_accuracy: 0.7574 - val_loss: 0.5875
s: 0.5875
Epoch 140/400
296/296 1s 5ms/step - accuracy: 0.7994 - loss: 0.4644 - val_accuracy: 0.7599 - val_loss: 0.5853
s: 0.5853
Epoch 141/400
296/296 1s 5ms/step - accuracy: 0.8031 - loss: 0.4620 - val_accuracy: 0.7568 - val_loss: 0.5857
s: 0.5857
Epoch 142/400
296/296 1s 5ms/step - accuracy: 0.8003 - loss: 0.4638 - val_accuracy: 0.7573 - val_loss: 0.5797
s: 0.5857
Epoch 143/400
296/296 1s 4ms/step - accuracy: 0.8017 - loss: 0.4647 - val_accuracy: 0.7580 - val_loss: 0.5797
s: 0.5857
Epoch 144/400
296/296 1s 5ms/step - accuracy: 0.8028 - loss: 0.4631 - val_accuracy: 0.7584 - val_loss: 0.5824
s: 0.5824
Epoch 145/400
296/296 1s 5ms/step - accuracy: 0.8025 - loss: 0.4648 - val_accuracy: 0.7591 - val_loss: 0.5789
s: 0.5789
Epoch 146/400
296/296 1s 5ms/step - accuracy: 0.8024 - loss: 0.4605 - val_accuracy: 0.7586 - val_loss: 0.5812
s: 0.5812
Epoch 147/400
296/296 1s 5ms/step - accuracy: 0.8037 - loss: 0.4568 - val_accuracy: 0.7606 - val_loss: 0.5789
s: 0.5789
Epoch 148/400
296/296 1s 4ms/step - accuracy: 0.8016 - loss: 0.4623 - val_accuracy: 0.7593 - val_loss: 0.5867
s: 0.5867
Epoch 149/400
296/296 1s 4ms/step - accuracy: 0.7986 - loss: 0.4649 - val_accuracy: 0.7599 - val_loss: 0.5772
s: 0.5772
Epoch 150/400
296/296 1s 4ms/step - accuracy: 0.8078 - loss: 0.4531 - val_accuracy: 0.7602 - val_loss: 0.5775
s: 0.5775
Epoch 151/400
296/296 1s 4ms/step - accuracy: 0.8054 - loss: 0.4593 - val_accuracy: 0.7600 - val_loss: 0.5839
s: 0.5839

Epoch 153/400
296/296 1s 4ms/step - accuracy: 0.8006 - loss: 0.4643 - val_accuracy: 0.7613 - val_loss: 0.5911
Epoch 154/400
296/296 1s 5ms/step - accuracy: 0.8040 - loss: 0.4563 - val_accuracy: 0.7605 - val_loss: 0.5822
Epoch 155/400
296/296 1s 4ms/step - accuracy: 0.8044 - loss: 0.4619 - val_accuracy: 0.7574 - val_loss: 0.5908
Epoch 156/400
296/296 1s 4ms/step - accuracy: 0.8037 - loss: 0.4575 - val_accuracy: 0.7626 - val_loss: 0.5851
Epoch 157/400
296/296 1s 5ms/step - accuracy: 0.8052 - loss: 0.4557 - val_accuracy: 0.7592 - val_loss: 0.5897
Epoch 158/400
296/296 2s 5ms/step - accuracy: 0.8032 - loss: 0.4571 - val_accuracy: 0.7625 - val_loss: 0.5855
Epoch 159/400
296/296 1s 5ms/step - accuracy: 0.8044 - loss: 0.4556 - val_accuracy: 0.7615 - val_loss: 0.5770
Epoch 160/400
296/296 1s 4ms/step - accuracy: 0.8052 - loss: 0.4532 - val_accuracy: 0.7602 - val_loss: 0.5847
Epoch 161/400
296/296 1s 5ms/step - accuracy: 0.8063 - loss: 0.4542 - val_accuracy: 0.7645 - val_loss: 0.5859
Epoch 162/400
296/296 1s 4ms/step - accuracy: 0.8056 - loss: 0.4529 - val_accuracy: 0.7596 - val_loss: 0.5821
Epoch 163/400
296/296 1s 4ms/step - accuracy: 0.8034 - loss: 0.4565 - val_accuracy: 0.7600 - val_loss: 0.5851
Epoch 164/400
296/296 1s 4ms/step - accuracy: 0.8088 - loss: 0.4499 - val_accuracy: 0.7605 - val_loss: 0.5840
Epoch 165/400
296/296 1s 4ms/step - accuracy: 0.8063 - loss: 0.4532 - val_accuracy: 0.7654 - val_loss: 0.5864
Epoch 166/400
296/296 1s 4ms/step - accuracy: 0.8068 - loss: 0.4508 - val_accuracy: 0.7617 - val_loss: 0.5803
Epoch 167/400
296/296 1s 5ms/step - accuracy: 0.8056 - loss: 0.4527 - val_accuracy: 0.7648 - val_loss: 0.5754
Epoch 168/400
296/296 1s 5ms/step - accuracy: 0.8067 - loss: 0.4504 - val_accuracy: 0.7625 - val_loss: 0.5804
Epoch 169/400
296/296 1s 5ms/step - accuracy: 0.8088 - loss: 0.4484 - val_accuracy: 0.7645 - val_loss: 0.5795
Epoch 170/400
296/296 1s 5ms/step - accuracy: 0.8091 - loss: 0.4475 - val_accuracy: 0.7620 - val_loss: 0.5762
Epoch 171/400
296/296 1s 4ms/step - accuracy: 0.8082 - loss: 0.4539 - val_accuracy: 0.7607 - val_loss: 0.5807
Epoch 172/400
296/296 1s 5ms/step - accuracy: 0.8083 - loss: 0.4495 - val_accuracy: 0.7590 - val_loss: 0.5762
Epoch 173/400
296/296 1s 5ms/step - accuracy: 0.8067 - loss: 0.4492 - val_accuracy: 0.7650 - val_loss: 0.5757
Epoch 174/400
296/296 1s 4ms/step - accuracy: 0.8054 - loss: 0.4496 - val_accuracy: 0.7615 - val_loss: 0.5833
Epoch 175/400
296/296 1s 5ms/step - accuracy: 0.8085 - loss: 0.4449 - val_accuracy: 0.7622 - val_loss: 0.5788
Epoch 176/400
296/296 1s 5ms/step - accuracy: 0.8084 - loss: 0.4501 - val_accuracy: 0.7614 - val_loss: 0.5888
Epoch 177/400
296/296 1s 5ms/step - accuracy: 0.8108 - loss: 0.4448 - val_accuracy: 0.7606 - val_loss: 0.5871
Epoch 178/400

296/296 1s 4ms/step - accuracy: 0.8062 - loss: 0.4469 - val_accuracy: 0.7635 - val_loss: 0.5695
Epoch 179/400
296/296 1s 5ms/step - accuracy: 0.8081 - loss: 0.4488 - val_accuracy: 0.7608 - val_loss: 0.5777
Epoch 180/400
296/296 2s 5ms/step - accuracy: 0.8114 - loss: 0.4454 - val_accuracy: 0.7651 - val_loss: 0.5742
Epoch 181/400
296/296 2s 5ms/step - accuracy: 0.8071 - loss: 0.4450 - val_accuracy: 0.7609 - val_loss: 0.5856
Epoch 182/400
296/296 1s 5ms/step - accuracy: 0.8132 - loss: 0.4438 - val_accuracy: 0.7602 - val_loss: 0.5782
Epoch 183/400
296/296 2s 5ms/step - accuracy: 0.8119 - loss: 0.4419 - val_accuracy: 0.7658 - val_loss: 0.5713
Epoch 184/400
296/296 1s 5ms/step - accuracy: 0.8101 - loss: 0.4496 - val_accuracy: 0.7596 - val_loss: 0.5862
Epoch 185/400
296/296 1s 5ms/step - accuracy: 0.8074 - loss: 0.4503 - val_accuracy: 0.7651 - val_loss: 0.5840
Epoch 186/400
296/296 1s 5ms/step - accuracy: 0.8110 - loss: 0.4447 - val_accuracy: 0.7637 - val_loss: 0.5857
Epoch 187/400
296/296 3s 5ms/step - accuracy: 0.8099 - loss: 0.4455 - val_accuracy: 0.7656 - val_loss: 0.5727
Epoch 188/400
296/296 1s 5ms/step - accuracy: 0.8119 - loss: 0.4451 - val_accuracy: 0.7627 - val_loss: 0.5755
Epoch 189/400
296/296 1s 5ms/step - accuracy: 0.8101 - loss: 0.4434 - val_accuracy: 0.7660 - val_loss: 0.5725
Epoch 190/400
296/296 1s 5ms/step - accuracy: 0.8100 - loss: 0.4474 - val_accuracy: 0.7651 - val_loss: 0.5776
Epoch 191/400
296/296 1s 5ms/step - accuracy: 0.8104 - loss: 0.4448 - val_accuracy: 0.7655 - val_loss: 0.5719
Epoch 192/400
296/296 1s 5ms/step - accuracy: 0.8110 - loss: 0.4400 - val_accuracy: 0.7641 - val_loss: 0.5736
Epoch 193/400
296/296 1s 4ms/step - accuracy: 0.8095 - loss: 0.4432 - val_accuracy: 0.7628 - val_loss: 0.5751
Epoch 194/400
296/296 1s 4ms/step - accuracy: 0.8167 - loss: 0.4380 - val_accuracy: 0.7634 - val_loss: 0.5709
Epoch 195/400
296/296 1s 4ms/step - accuracy: 0.8115 - loss: 0.4427 - val_accuracy: 0.7636 - val_loss: 0.5906
Epoch 196/400
296/296 1s 5ms/step - accuracy: 0.8102 - loss: 0.4452 - val_accuracy: 0.7647 - val_loss: 0.5805
Epoch 197/400
296/296 1s 5ms/step - accuracy: 0.8098 - loss: 0.4431 - val_accuracy: 0.7636 - val_loss: 0.5753
Epoch 198/400
296/296 1s 5ms/step - accuracy: 0.8124 - loss: 0.4394 - val_accuracy: 0.7626 - val_loss: 0.5806
Epoch 199/400
296/296 1s 5ms/step - accuracy: 0.8131 - loss: 0.4377 - val_accuracy: 0.7616 - val_loss: 0.5775
Epoch 200/400
296/296 1s 4ms/step - accuracy: 0.8128 - loss: 0.4392 - val_accuracy: 0.7619 - val_loss: 0.5842
Epoch 201/400
296/296 1s 5ms/step - accuracy: 0.8136 - loss: 0.4379 - val_accuracy: 0.7610 - val_loss: 0.5802
Epoch 202/400
296/296 1s 5ms/step - accuracy: 0.8119 - loss: 0.4406 - val_accuracy: 0.7664 - val_loss: 0.5839
Epoch 203/400
296/296 1s 5ms/step - accuracy: 0.8137 - loss: 0.4343 - val_accuracy: 0.7656 - val_loss:

s: 0.5824
Epoch 204/400
296/296 1s 5ms/step - accuracy: 0.8128 - loss: 0.4367 - val_accuracy: 0.7641 - val_loss: 0.5782
Epoch 205/400
296/296 1s 5ms/step - accuracy: 0.8115 - loss: 0.4401 - val_accuracy: 0.7610 - val_loss: 0.5785
Epoch 206/400
296/296 1s 4ms/step - accuracy: 0.8108 - loss: 0.4420 - val_accuracy: 0.7642 - val_loss: 0.5762
Epoch 207/400
296/296 1s 4ms/step - accuracy: 0.8119 - loss: 0.4391 - val_accuracy: 0.7631 - val_loss: 0.5798
Epoch 208/400
296/296 1s 4ms/step - accuracy: 0.8138 - loss: 0.4434 - val_accuracy: 0.7624 - val_loss: 0.5879
Epoch 209/400
296/296 1s 4ms/step - accuracy: 0.8135 - loss: 0.4389 - val_accuracy: 0.7664 - val_loss: 0.5837
Epoch 210/400
296/296 1s 5ms/step - accuracy: 0.8143 - loss: 0.4391 - val_accuracy: 0.7663 - val_loss: 0.5681
Epoch 211/400
296/296 1s 4ms/step - accuracy: 0.8138 - loss: 0.4361 - val_accuracy: 0.7646 - val_loss: 0.5815
Epoch 212/400
296/296 1s 5ms/step - accuracy: 0.8149 - loss: 0.4362 - val_accuracy: 0.7674 - val_loss: 0.5832
Epoch 213/400
296/296 1s 5ms/step - accuracy: 0.8119 - loss: 0.4433 - val_accuracy: 0.7635 - val_loss: 0.5752
Epoch 214/400
296/296 1s 5ms/step - accuracy: 0.8159 - loss: 0.4323 - val_accuracy: 0.7654 - val_loss: 0.5809
Epoch 215/400
296/296 1s 5ms/step - accuracy: 0.8134 - loss: 0.4306 - val_accuracy: 0.7628 - val_loss: 0.5844
Epoch 216/400
296/296 1s 5ms/step - accuracy: 0.8136 - loss: 0.4386 - val_accuracy: 0.7636 - val_loss: 0.5845
Epoch 217/400
296/296 1s 5ms/step - accuracy: 0.8158 - loss: 0.4341 - val_accuracy: 0.7635 - val_loss: 0.5859
Epoch 218/400
296/296 1s 5ms/step - accuracy: 0.8201 - loss: 0.4265 - val_accuracy: 0.7587 - val_loss: 0.5826
Epoch 219/400
296/296 1s 5ms/step - accuracy: 0.8158 - loss: 0.4347 - val_accuracy: 0.7653 - val_loss: 0.5790
Epoch 220/400
296/296 1s 5ms/step - accuracy: 0.8162 - loss: 0.4342 - val_accuracy: 0.7638 - val_loss: 0.5784
Epoch 221/400
296/296 1s 5ms/step - accuracy: 0.8170 - loss: 0.4304 - val_accuracy: 0.7611 - val_loss: 0.5773
Epoch 222/400
296/296 1s 4ms/step - accuracy: 0.8171 - loss: 0.4328 - val_accuracy: 0.7610 - val_loss: 0.5845
Epoch 223/400
296/296 1s 4ms/step - accuracy: 0.8139 - loss: 0.4363 - val_accuracy: 0.7645 - val_loss: 0.5901
Epoch 224/400
296/296 1s 5ms/step - accuracy: 0.8172 - loss: 0.4311 - val_accuracy: 0.7662 - val_loss: 0.5798
Epoch 225/400
296/296 1s 5ms/step - accuracy: 0.8149 - loss: 0.4339 - val_accuracy: 0.7626 - val_loss: 0.5883
Epoch 226/400
296/296 1s 5ms/step - accuracy: 0.8162 - loss: 0.4339 - val_accuracy: 0.7587 - val_loss: 0.5799
Epoch 227/400
296/296 1s 5ms/step - accuracy: 0.8191 - loss: 0.4291 - val_accuracy: 0.7626 - val_loss: 0.5806
Epoch 228/400
296/296 1s 5ms/step - accuracy: 0.8178 - loss: 0.4277 - val_accuracy: 0.7608 - val_loss: 0.5834

Epoch 229/400
296/296 1s 4ms/step - accuracy: 0.8185 - loss: 0.4301 - val_accuracy: 0.7672 - val_loss: 0.5874
Epoch 230/400
296/296 1s 4ms/step - accuracy: 0.8174 - loss: 0.4290 - val_accuracy: 0.7650 - val_loss: 0.5848
Epoch 231/400
296/296 1s 4ms/step - accuracy: 0.8192 - loss: 0.4333 - val_accuracy: 0.7650 - val_loss: 0.5871
Epoch 232/400
296/296 1s 5ms/step - accuracy: 0.8169 - loss: 0.4328 - val_accuracy: 0.7635 - val_loss: 0.5828
Epoch 233/400
296/296 1s 5ms/step - accuracy: 0.8160 - loss: 0.4301 - val_accuracy: 0.7599 - val_loss: 0.5804
Epoch 234/400
296/296 1s 5ms/step - accuracy: 0.8155 - loss: 0.4361 - val_accuracy: 0.7649 - val_loss: 0.5763
Epoch 235/400
296/296 1s 5ms/step - accuracy: 0.8167 - loss: 0.4338 - val_accuracy: 0.7649 - val_loss: 0.5754
Epoch 236/400
296/296 1s 5ms/step - accuracy: 0.8164 - loss: 0.4329 - val_accuracy: 0.7606 - val_loss: 0.5780
Epoch 237/400
296/296 1s 5ms/step - accuracy: 0.8153 - loss: 0.4311 - val_accuracy: 0.7649 - val_loss: 0.5783
Epoch 238/400
296/296 1s 5ms/step - accuracy: 0.8192 - loss: 0.4302 - val_accuracy: 0.7636 - val_loss: 0.5713
Epoch 239/400
296/296 1s 4ms/step - accuracy: 0.8153 - loss: 0.4296 - val_accuracy: 0.7652 - val_loss: 0.5820
Epoch 240/400
296/296 1s 5ms/step - accuracy: 0.8191 - loss: 0.4274 - val_accuracy: 0.7615 - val_loss: 0.5856
Epoch 241/400
296/296 1s 5ms/step - accuracy: 0.8157 - loss: 0.4321 - val_accuracy: 0.7611 - val_loss: 0.5867
Epoch 242/400
296/296 1s 5ms/step - accuracy: 0.8192 - loss: 0.4284 - val_accuracy: 0.7626 - val_loss: 0.5804
Epoch 243/400
296/296 1s 5ms/step - accuracy: 0.8201 - loss: 0.4267 - val_accuracy: 0.7663 - val_loss: 0.5745
Epoch 244/400
296/296 1s 5ms/step - accuracy: 0.8199 - loss: 0.4275 - val_accuracy: 0.7643 - val_loss: 0.5863
Epoch 245/400
296/296 1s 4ms/step - accuracy: 0.8153 - loss: 0.4344 - val_accuracy: 0.7657 - val_loss: 0.5757
Epoch 246/400
296/296 1s 4ms/step - accuracy: 0.8153 - loss: 0.4318 - val_accuracy: 0.7638 - val_loss: 0.5815
Epoch 247/400
296/296 1s 5ms/step - accuracy: 0.8206 - loss: 0.4230 - val_accuracy: 0.7663 - val_loss: 0.5872
Epoch 248/400
296/296 1s 5ms/step - accuracy: 0.8182 - loss: 0.4273 - val_accuracy: 0.7641 - val_loss: 0.5764
Epoch 249/400
296/296 1s 5ms/step - accuracy: 0.8210 - loss: 0.4242 - val_accuracy: 0.7651 - val_loss: 0.5701
Epoch 250/400
296/296 1s 5ms/step - accuracy: 0.8175 - loss: 0.4300 - val_accuracy: 0.7664 - val_loss: 0.5856
Epoch 251/400
296/296 1s 4ms/step - accuracy: 0.8192 - loss: 0.4281 - val_accuracy: 0.7628 - val_loss: 0.5900
Epoch 252/400
296/296 1s 4ms/step - accuracy: 0.8207 - loss: 0.4248 - val_accuracy: 0.7624 - val_loss: 0.5830
Epoch 253/400
296/296 1s 4ms/step - accuracy: 0.8179 - loss: 0.4276 - val_accuracy: 0.7617 - val_loss: 0.5790
Epoch 254/400

```
296/296 ━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8237 - loss: 0.4215 - val_accuracy: 0.7635 - val_loss: 0.5847
Epoch 255/400
296/296 ━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8206 - loss: 0.4207 - val_accuracy: 0.7602 - val_loss: 0.5790
Epoch 256/400
296/296 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8174 - loss: 0.4282 - val_accuracy: 0.7648 - val_loss: 0.5850
Epoch 257/400
296/296 ━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8184 - loss: 0.4249 - val_accuracy: 0.7651 - val_loss: 0.5819
Epoch 258/400
296/296 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8196 - loss: 0.4248 - val_accuracy: 0.7660 - val_loss: 0.5806
Epoch 259/400
296/296 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8185 - loss: 0.4264 - val_accuracy: 0.7647 - val_loss: 0.5864
Epoch 260/400
296/296 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8198 - loss: 0.4212 - val_accuracy: 0.7637 - val_loss: 0.5853
Epoch 261/400
296/296 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8185 - loss: 0.4259 - val_accuracy: 0.7648 - val_loss: 0.5761
Epoch 262/400
296/296 ━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8225 - loss: 0.4228 - val_accuracy: 0.7666 - val_loss: 0.5845
Epoch 263/400
296/296 ━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8197 - loss: 0.4252 - val_accuracy: 0.7641 - val_loss: 0.5835
Epoch 264/400
296/296 ━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8203 - loss: 0.4271 - val_accuracy: 0.7651 - val_loss: 0.5813
Epoch 265/400
296/296 ━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8221 - loss: 0.4236 - val_accuracy: 0.7631 - val_loss: 0.5805
Epoch 266/400
296/296 ━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8201 - loss: 0.4248 - val_accuracy: 0.7643 - val_loss: 0.5783
Epoch 267/400
296/296 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8199 - loss: 0.4209 - val_accuracy: 0.7649 - val_loss: 0.5805
Epoch 268/400
296/296 ━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8218 - loss: 0.4222 - val_accuracy: 0.7625 - val_loss: 0.5872
Epoch 269/400
296/296 ━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8224 - loss: 0.4240 - val_accuracy: 0.7638 - val_loss: 0.5798
Epoch 270/400
296/296 ━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8197 - loss: 0.4243 - val_accuracy: 0.7667 - val_loss: 0.5836
Epoch 271/400
296/296 ━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8189 - loss: 0.4247 - val_accuracy: 0.7670 - val_loss: 0.5727
Epoch 272/400
296/296 ━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8197 - loss: 0.4262 - val_accuracy: 0.7651 - val_loss: 0.5885
```

```
In [225...]: ann3.evaluate(X_train, y_train)
```

```
2363/2363 ━━━━━━━━━━ 2s 859us/step - accuracy: 0.8819 - loss: 0.3003
```

```
Out[225...]: [0.30011290311813354, 0.8829387426376343]
```

```
In [226...]: ann3.summary()
```

```
Model: "sequential_3"
```

Layer (type)	Output Shape	Param #
dense_17 (Dense)	(None, 512)	23,040
dropout_14 (Dropout)	(None, 512)	0
dense_18 (Dense)	(None, 256)	131,328
dropout_15 (Dropout)	(None, 256)	0
dense_19 (Dense)	(None, 128)	32,896
dropout_16 (Dropout)	(None, 128)	0
dense_20 (Dense)	(None, 128)	16,512
dropout_17 (Dropout)	(None, 128)	0
dense_21 (Dense)	(None, 128)	16,512
dropout_18 (Dropout)	(None, 128)	0
dense_22 (Dense)	(None, 64)	8,256
dropout_19 (Dropout)	(None, 64)	0
dense_23 (Dense)	(None, 3)	195

Total params: 686,219 (2.62 MB)

Trainable params: 228,739 (893.51 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 457,480 (1.75 MB)

```
In [227]: eval_metric(ann3, X_train, y_train, X_val, y_val)
```

```
2363/2363 ━━━━━━━━ 2s 885us/step
591/591 ━━━━━━━━ 1s 976us/step
```

Test Set:

```
[[4318 1049 141]
 [1303 7771 979]
 [ 58 867 2417]]
```

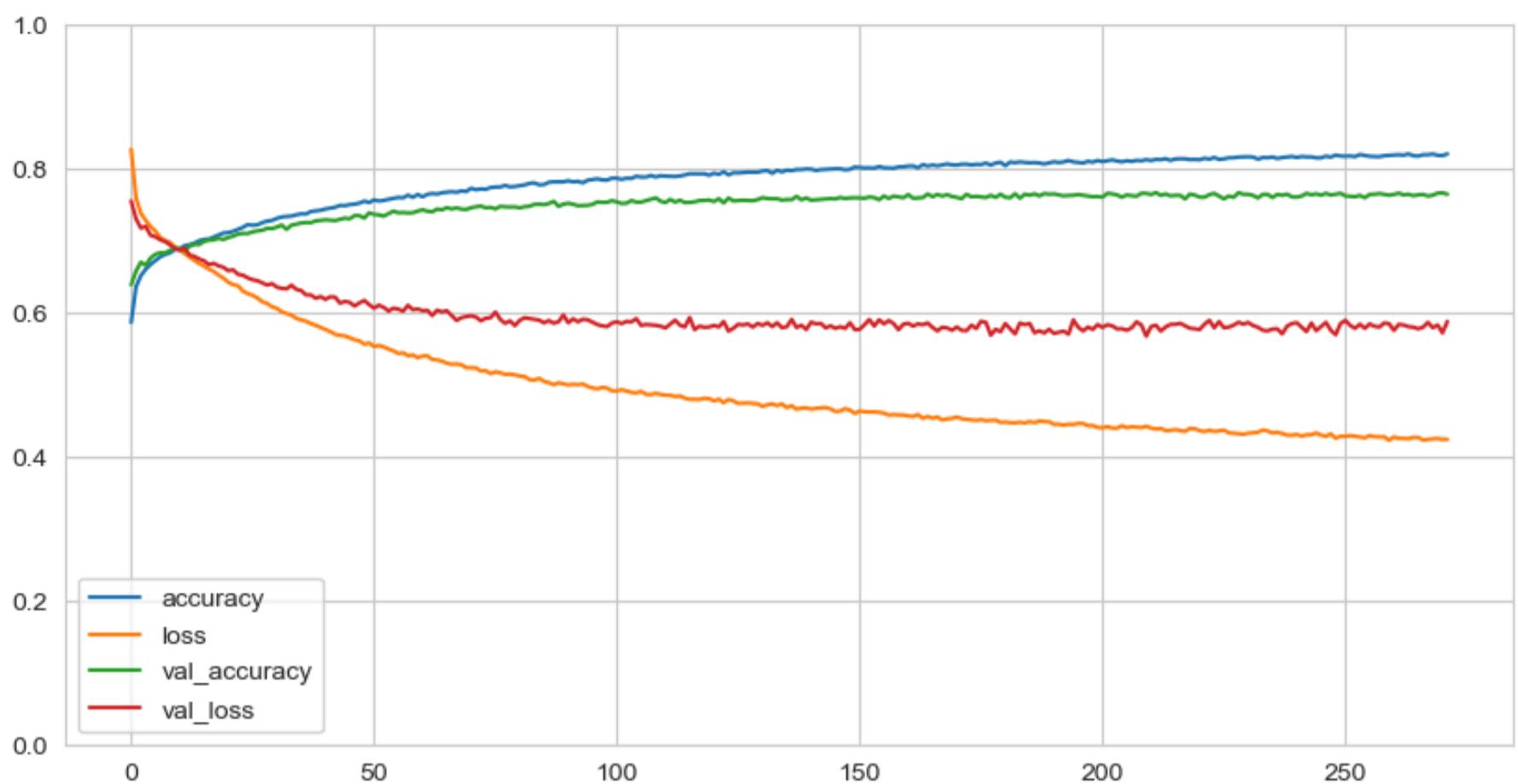
	precision	recall	f1-score	support
0	0.76	0.78	0.77	5508
1	0.80	0.77	0.79	10053
2	0.68	0.72	0.70	3342
accuracy			0.77	18903
macro avg	0.75	0.76	0.75	18903
weighted avg	0.77	0.77	0.77	18903

Train Set:

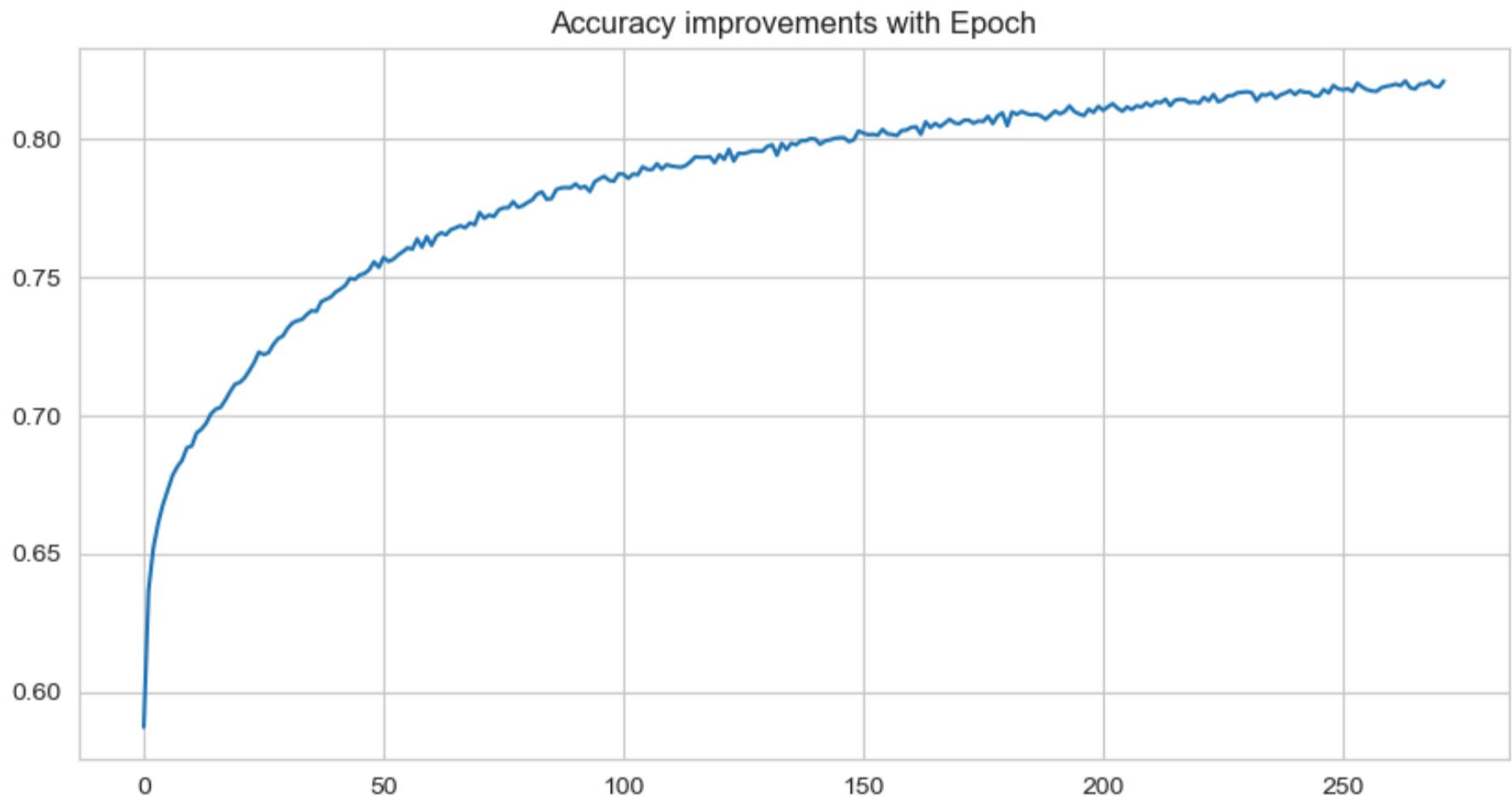
```
[[20230 1618 183]
 [ 2965 34854 2390]
 [ 25 1670 11675]]
```

	precision	recall	f1-score	support
0	0.87	0.92	0.89	22031
1	0.91	0.87	0.89	40209
2	0.82	0.87	0.85	13370
accuracy			0.88	75610
macro avg	0.87	0.89	0.88	75610
weighted avg	0.88	0.88	0.88	75610

```
In [228]: pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



```
In [229...]: pd.DataFrame(history.history)[["accuracy"]].plot(figsize=(10, 5))
plt.title("Accuracy improvements with Epoch")
plt.show()
```



```
In [230...]: # Save the Model

from tensorflow.keras.models import load_model

# Save the model to 'model.h5' file
ann3.save('ann3_model.h5')

# To Load the model Later for further use
loaded_ann3 = load_model('ann3_model.h5')
```

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)` . This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')` .
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.

ANN-3 Model Summary:

- **(Dense) layers:** 7 / **Neurons:** 512-256-128-128-128-64 / **Dropout:** 30-30-25-25-20-20% / **Learning Rate:** 0.001 / **Batch Size:** 256 / **Epochs:** 400 / **Early Stop (val_accuracy):** 50

- **Accuracy:** 0.8819 / **Val_Accuracy:** 0.7658 / **Loss:** 0.3003 / **Val_Loss:** 0.5716
- **Train Recall (Class 2):** 0.87 / **Test Recall(Class 2):** 0.71

The ANN-3 model shows significant improvements in training accuracy and loss, indicating effective learning on the training dataset. However, the validation metrics have not shown similar improvement, suggesting potential overfitting despite higher dropout settings. The consistent gap between training and validation performance, especially in recall for Class 2, points to challenges in generalizing to new data, possibly due to class imbalance. Adjustments in future models might focus on enhancing data representation or using techniques to improve the model's ability to generalize across different data distributions.

ANN-3 Model with SMOTE (%85)

```
In [29]: print('Credi_Score Classes before Smote: ', y_train.value_counts())
print('\nCredi_Score Classes after Smote: ', y_train_smote.value_counts())
```

```
Credi_Score Classes before Smote: Credit_Score
1    40209
0    22031
2    13370
Name: count, dtype: int64
```

```
Credi_Score Classes after Smote: Credit_Score
1    40209
2    40209
0    40209
Name: count, dtype: int64
```

```
In [244... # ANN-3 Model with SMOTE
```

```
# 1) Model Architecture:
# Increased Fully connected (Dense) from 5 to ----> 7 Layers
# Increased Neurons number from 256 to ----> 512
ann3_smote = Sequential([
    Dense(512, input_dim=X_train_smote.shape[1], activation='relu'),
    Dropout(0.3),
    Dense(256, activation='relu'),
    Dropout(0.3),
    Dense(128, activation='relu'),
    Dropout(0.25),
    Dense(128, activation='relu'),
    Dropout(0.25),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(3, activation='softmax')
])

# 2) Compiling the Model:
ann3_smote.compile(optimizer=Adam(learning_rate=0.001),
                    loss='sparse_categorical_crossentropy',
                    metrics=['accuracy'])

# 3) Early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=60, # Increased from 50 to --> 60
                               mode="auto",
                               restore_best_weights=True)

# 4) Train the model
history = ann3_smote.fit(
    x=X_train_smote,
    y=y_train_smote,
    validation_data=(X_val, y_val),
    batch_size=256, # same
    epochs=400,      # Increased Epoch from 300 to --> 400
    verbose=1,
```

```
        callbacks=[early_stopping],  
        class_weight = class_weights_dict)    # + class_weight is been used
```

Epoch 1/400
472/472 5s 5ms/step - accuracy: 0.5582 - loss: 0.9259 - val_accuracy: 0.5553 - val_loss: 1.0139
Epoch 2/400
472/472 2s 5ms/step - accuracy: 0.6633 - loss: 0.7825 - val_accuracy: 0.6139 - val_loss: 1.0013
Epoch 3/400
472/472 2s 4ms/step - accuracy: 0.6882 - loss: 0.7392 - val_accuracy: 0.5699 - val_loss: 0.9739
Epoch 4/400
472/472 2s 5ms/step - accuracy: 0.6944 - loss: 0.7122 - val_accuracy: 0.6011 - val_loss: 0.9234
Epoch 5/400
472/472 2s 5ms/step - accuracy: 0.7065 - loss: 0.6832 - val_accuracy: 0.5988 - val_loss: 0.9570
Epoch 6/400
472/472 2s 5ms/step - accuracy: 0.7146 - loss: 0.6606 - val_accuracy: 0.6076 - val_loss: 0.9542
Epoch 7/400
472/472 2s 5ms/step - accuracy: 0.7224 - loss: 0.6445 - val_accuracy: 0.6232 - val_loss: 0.9029
Epoch 8/400
472/472 2s 5ms/step - accuracy: 0.7268 - loss: 0.6281 - val_accuracy: 0.6298 - val_loss: 0.8723
Epoch 9/400
472/472 2s 4ms/step - accuracy: 0.7293 - loss: 0.6143 - val_accuracy: 0.6374 - val_loss: 0.8882
Epoch 10/400
472/472 2s 5ms/step - accuracy: 0.7331 - loss: 0.6016 - val_accuracy: 0.6184 - val_loss: 0.8990
Epoch 11/400
472/472 2s 5ms/step - accuracy: 0.7407 - loss: 0.5864 - val_accuracy: 0.6181 - val_loss: 0.9187
Epoch 12/400
472/472 2s 5ms/step - accuracy: 0.7426 - loss: 0.5754 - val_accuracy: 0.6307 - val_loss: 0.8649
Epoch 13/400
472/472 2s 5ms/step - accuracy: 0.7428 - loss: 0.5699 - val_accuracy: 0.6345 - val_loss: 0.8945
Epoch 14/400
472/472 2s 5ms/step - accuracy: 0.7457 - loss: 0.5616 - val_accuracy: 0.6455 - val_loss: 0.8420
Epoch 15/400
472/472 2s 5ms/step - accuracy: 0.7527 - loss: 0.5484 - val_accuracy: 0.6104 - val_loss: 0.9394
Epoch 16/400
472/472 2s 5ms/step - accuracy: 0.7521 - loss: 0.5433 - val_accuracy: 0.6246 - val_loss: 0.8855
Epoch 17/400
472/472 2s 5ms/step - accuracy: 0.7552 - loss: 0.5352 - val_accuracy: 0.6315 - val_loss: 0.8833
Epoch 18/400
472/472 2s 5ms/step - accuracy: 0.7566 - loss: 0.5309 - val_accuracy: 0.6369 - val_loss: 0.8850
Epoch 19/400
472/472 2s 4ms/step - accuracy: 0.7565 - loss: 0.5295 - val_accuracy: 0.6569 - val_loss: 0.8455
Epoch 20/400
472/472 2s 5ms/step - accuracy: 0.7619 - loss: 0.5180 - val_accuracy: 0.6287 - val_loss: 0.8530
Epoch 21/400
472/472 3s 5ms/step - accuracy: 0.7646 - loss: 0.5106 - val_accuracy: 0.6487 - val_loss: 0.8456
Epoch 22/400
472/472 2s 5ms/step - accuracy: 0.7663 - loss: 0.5098 - val_accuracy: 0.6497 - val_loss: 0.8394
Epoch 23/400
472/472 2s 5ms/step - accuracy: 0.7673 - loss: 0.5047 - val_accuracy: 0.6368 - val_loss: 0.8555
Epoch 24/400
472/472 3s 5ms/step - accuracy: 0.7699 - loss: 0.4943 - val_accuracy: 0.6438 - val_loss: 0.8549
Epoch 25/400
472/472 2s 5ms/step - accuracy: 0.7680 - loss: 0.4987 - val_accuracy: 0.6550 - val_loss: 0.8416
Epoch 26/400

472/472 2s 5ms/step - accuracy: 0.7743 - loss: 0.4941 - val_accuracy: 0.6485 - val_loss: 0.8479
Epoch 27/400
472/472 3s 5ms/step - accuracy: 0.7736 - loss: 0.4916 - val_accuracy: 0.6595 - val_loss: 0.8527
Epoch 28/400
472/472 2s 5ms/step - accuracy: 0.7757 - loss: 0.4826 - val_accuracy: 0.6579 - val_loss: 0.8182
Epoch 29/400
472/472 2s 5ms/step - accuracy: 0.7788 - loss: 0.4783 - val_accuracy: 0.6642 - val_loss: 0.8059
Epoch 30/400
472/472 2s 5ms/step - accuracy: 0.7792 - loss: 0.4799 - val_accuracy: 0.6536 - val_loss: 0.8380
Epoch 31/400
472/472 2s 5ms/step - accuracy: 0.7775 - loss: 0.4734 - val_accuracy: 0.6536 - val_loss: 0.8390
Epoch 32/400
472/472 2s 5ms/step - accuracy: 0.7800 - loss: 0.4707 - val_accuracy: 0.6698 - val_loss: 0.8076
Epoch 33/400
472/472 2s 5ms/step - accuracy: 0.7831 - loss: 0.4704 - val_accuracy: 0.6503 - val_loss: 0.8575
Epoch 34/400
472/472 2s 5ms/step - accuracy: 0.7852 - loss: 0.4622 - val_accuracy: 0.6714 - val_loss: 0.8044
Epoch 35/400
472/472 2s 5ms/step - accuracy: 0.7835 - loss: 0.4661 - val_accuracy: 0.6787 - val_loss: 0.7892
Epoch 36/400
472/472 2s 5ms/step - accuracy: 0.7864 - loss: 0.4646 - val_accuracy: 0.6677 - val_loss: 0.8068
Epoch 37/400
472/472 2s 4ms/step - accuracy: 0.7857 - loss: 0.4617 - val_accuracy: 0.6668 - val_loss: 0.8151
Epoch 38/400
472/472 2s 4ms/step - accuracy: 0.7879 - loss: 0.4575 - val_accuracy: 0.6568 - val_loss: 0.8311
Epoch 39/400
472/472 2s 5ms/step - accuracy: 0.7871 - loss: 0.4576 - val_accuracy: 0.6652 - val_loss: 0.8198
Epoch 40/400
472/472 2s 4ms/step - accuracy: 0.7917 - loss: 0.4471 - val_accuracy: 0.6796 - val_loss: 0.8117
Epoch 41/400
472/472 2s 4ms/step - accuracy: 0.7902 - loss: 0.4511 - val_accuracy: 0.6857 - val_loss: 0.8011
Epoch 42/400
472/472 2s 5ms/step - accuracy: 0.7937 - loss: 0.4471 - val_accuracy: 0.6897 - val_loss: 0.7686
Epoch 43/400
472/472 2s 5ms/step - accuracy: 0.7951 - loss: 0.4433 - val_accuracy: 0.6921 - val_loss: 0.7669
Epoch 44/400
472/472 2s 5ms/step - accuracy: 0.7939 - loss: 0.4491 - val_accuracy: 0.6656 - val_loss: 0.8468
Epoch 45/400
472/472 2s 4ms/step - accuracy: 0.7927 - loss: 0.4451 - val_accuracy: 0.6791 - val_loss: 0.7837
Epoch 46/400
472/472 2s 4ms/step - accuracy: 0.7933 - loss: 0.4416 - val_accuracy: 0.6734 - val_loss: 0.7893
Epoch 47/400
472/472 2s 4ms/step - accuracy: 0.7947 - loss: 0.4387 - val_accuracy: 0.6743 - val_loss: 0.8145
Epoch 48/400
472/472 2s 5ms/step - accuracy: 0.7951 - loss: 0.4386 - val_accuracy: 0.6850 - val_loss: 0.7728
Epoch 49/400
472/472 2s 5ms/step - accuracy: 0.8005 - loss: 0.4293 - val_accuracy: 0.6837 - val_loss: 0.7867
Epoch 50/400
472/472 2s 5ms/step - accuracy: 0.8008 - loss: 0.4301 - val_accuracy: 0.6834 - val_loss: 0.7687
Epoch 51/400
472/472 2s 5ms/step - accuracy: 0.7995 - loss: 0.4338 - val_accuracy: 0.6795 - val_loss:

s: 0.7987
Epoch 52/400
472/472 2s 5ms/step - accuracy: 0.8003 - loss: 0.4316 - val_accuracy: 0.6787 - val_loss: 0.4316
s: 0.8113
Epoch 53/400
472/472 2s 5ms/step - accuracy: 0.8016 - loss: 0.4311 - val_accuracy: 0.6775 - val_loss: 0.4311
s: 0.7984
Epoch 54/400
472/472 2s 5ms/step - accuracy: 0.8007 - loss: 0.4281 - val_accuracy: 0.6796 - val_loss: 0.4281
s: 0.7888
Epoch 55/400
472/472 2s 5ms/step - accuracy: 0.8015 - loss: 0.4286 - val_accuracy: 0.6798 - val_loss: 0.4286
s: 0.8005
Epoch 56/400
472/472 2s 5ms/step - accuracy: 0.7988 - loss: 0.4287 - val_accuracy: 0.6920 - val_loss: 0.4287
s: 0.7938
Epoch 57/400
472/472 2s 5ms/step - accuracy: 0.8054 - loss: 0.4220 - val_accuracy: 0.6830 - val_loss: 0.4220
s: 0.8024
Epoch 58/400
472/472 2s 5ms/step - accuracy: 0.8042 - loss: 0.4179 - val_accuracy: 0.6854 - val_loss: 0.4179
s: 0.7904
Epoch 59/400
472/472 2s 5ms/step - accuracy: 0.8059 - loss: 0.4211 - val_accuracy: 0.6998 - val_loss: 0.4211
s: 0.7618
Epoch 60/400
472/472 2s 5ms/step - accuracy: 0.8071 - loss: 0.4167 - val_accuracy: 0.6894 - val_loss: 0.4167
s: 0.7832
Epoch 61/400
472/472 2s 5ms/step - accuracy: 0.8076 - loss: 0.4149 - val_accuracy: 0.6815 - val_loss: 0.4149
s: 0.7977
Epoch 62/400
472/472 2s 5ms/step - accuracy: 0.8068 - loss: 0.4190 - val_accuracy: 0.6929 - val_loss: 0.4190
s: 0.7969
Epoch 63/400
472/472 2s 5ms/step - accuracy: 0.8092 - loss: 0.4120 - val_accuracy: 0.6895 - val_loss: 0.4120
s: 0.7987
Epoch 64/400
472/472 2s 5ms/step - accuracy: 0.8063 - loss: 0.4144 - val_accuracy: 0.6951 - val_loss: 0.4144
s: 0.7823
Epoch 65/400
472/472 2s 4ms/step - accuracy: 0.8096 - loss: 0.4107 - val_accuracy: 0.6970 - val_loss: 0.4107
s: 0.7738
Epoch 66/400
472/472 2s 5ms/step - accuracy: 0.8111 - loss: 0.4105 - val_accuracy: 0.6893 - val_loss: 0.4105
s: 0.8027
Epoch 67/400
472/472 2s 5ms/step - accuracy: 0.8103 - loss: 0.4107 - val_accuracy: 0.6932 - val_loss: 0.4107
s: 0.7584
Epoch 68/400
472/472 2s 5ms/step - accuracy: 0.8102 - loss: 0.4091 - val_accuracy: 0.7020 - val_loss: 0.4091
s: 0.7655
Epoch 69/400
472/472 2s 5ms/step - accuracy: 0.8120 - loss: 0.4050 - val_accuracy: 0.7027 - val_loss: 0.4050
s: 0.7443
Epoch 70/400
472/472 2s 5ms/step - accuracy: 0.8114 - loss: 0.4050 - val_accuracy: 0.7045 - val_loss: 0.4050
s: 0.7699
Epoch 71/400
472/472 2s 5ms/step - accuracy: 0.8133 - loss: 0.4067 - val_accuracy: 0.7007 - val_loss: 0.4067
s: 0.7753
Epoch 72/400
472/472 2s 5ms/step - accuracy: 0.8135 - loss: 0.4041 - val_accuracy: 0.7040 - val_loss: 0.4041
s: 0.7356
Epoch 73/400
472/472 2s 5ms/step - accuracy: 0.8101 - loss: 0.4095 - val_accuracy: 0.6866 - val_loss: 0.4095
s: 0.8218
Epoch 74/400
472/472 2s 5ms/step - accuracy: 0.8112 - loss: 0.4047 - val_accuracy: 0.7038 - val_loss: 0.4047
s: 0.7629
Epoch 75/400
472/472 2s 5ms/step - accuracy: 0.8157 - loss: 0.3991 - val_accuracy: 0.6893 - val_loss: 0.3991
s: 0.7827
Epoch 76/400
472/472 2s 5ms/step - accuracy: 0.8136 - loss: 0.4029 - val_accuracy: 0.7004 - val_loss: 0.4029
s: 0.7629

Epoch 77/400
472/472 2s 5ms/step - accuracy: 0.8123 - loss: 0.4044 - val_accuracy: 0.7065 - val_loss:
s: 0.7726

Epoch 78/400
472/472 3s 5ms/step - accuracy: 0.8151 - loss: 0.3982 - val_accuracy: 0.6943 - val_loss:
s: 0.7729

Epoch 79/400
472/472 2s 5ms/step - accuracy: 0.8160 - loss: 0.3980 - val_accuracy: 0.6988 - val_loss:
s: 0.7864

Epoch 80/400
472/472 2s 5ms/step - accuracy: 0.8170 - loss: 0.3976 - val_accuracy: 0.7017 - val_loss:
s: 0.7683

Epoch 81/400
472/472 2s 5ms/step - accuracy: 0.8163 - loss: 0.3940 - val_accuracy: 0.7042 - val_loss:
s: 0.7585

Epoch 82/400
472/472 2s 5ms/step - accuracy: 0.8159 - loss: 0.4000 - val_accuracy: 0.6971 - val_loss:
s: 0.7771

Epoch 83/400
472/472 2s 5ms/step - accuracy: 0.8161 - loss: 0.3993 - val_accuracy: 0.6974 - val_loss:
s: 0.7814

Epoch 84/400
472/472 2s 5ms/step - accuracy: 0.8172 - loss: 0.3934 - val_accuracy: 0.7044 - val_loss:
s: 0.7360

Epoch 85/400
472/472 2s 5ms/step - accuracy: 0.8177 - loss: 0.3973 - val_accuracy: 0.7031 - val_loss:
s: 0.7560

Epoch 86/400
472/472 2s 5ms/step - accuracy: 0.8181 - loss: 0.3937 - val_accuracy: 0.7025 - val_loss:
s: 0.7611

Epoch 87/400
472/472 2s 5ms/step - accuracy: 0.8194 - loss: 0.3912 - val_accuracy: 0.7013 - val_loss:
s: 0.7749

Epoch 88/400
472/472 2s 5ms/step - accuracy: 0.8168 - loss: 0.3941 - val_accuracy: 0.7095 - val_loss:
s: 0.7641

Epoch 89/400
472/472 2s 5ms/step - accuracy: 0.8200 - loss: 0.3918 - val_accuracy: 0.6962 - val_loss:
s: 0.7747

Epoch 90/400
472/472 2s 5ms/step - accuracy: 0.8185 - loss: 0.3898 - val_accuracy: 0.7048 - val_loss:
s: 0.7468

Epoch 91/400
472/472 2s 5ms/step - accuracy: 0.8204 - loss: 0.3882 - val_accuracy: 0.7018 - val_loss:
s: 0.7707

Epoch 92/400
472/472 2s 5ms/step - accuracy: 0.8197 - loss: 0.3918 - val_accuracy: 0.7045 - val_loss:
s: 0.7708

Epoch 93/400
472/472 3s 5ms/step - accuracy: 0.8197 - loss: 0.3901 - val_accuracy: 0.6924 - val_loss:
s: 0.7826

Epoch 94/400
472/472 2s 5ms/step - accuracy: 0.8176 - loss: 0.3895 - val_accuracy: 0.7126 - val_loss:
s: 0.7544

Epoch 95/400
472/472 2s 5ms/step - accuracy: 0.8220 - loss: 0.3844 - val_accuracy: 0.7069 - val_loss:
s: 0.7592

Epoch 96/400
472/472 2s 5ms/step - accuracy: 0.8182 - loss: 0.3897 - val_accuracy: 0.7081 - val_loss:
s: 0.7409

Epoch 97/400
472/472 2s 5ms/step - accuracy: 0.8230 - loss: 0.3843 - val_accuracy: 0.6992 - val_loss:
s: 0.7577

Epoch 98/400
472/472 2s 5ms/step - accuracy: 0.8222 - loss: 0.3863 - val_accuracy: 0.7111 - val_loss:
s: 0.7463

Epoch 99/400
472/472 2s 5ms/step - accuracy: 0.8208 - loss: 0.3869 - val_accuracy: 0.7028 - val_loss:
s: 0.7483

Epoch 100/400
472/472 2s 5ms/step - accuracy: 0.8245 - loss: 0.3817 - val_accuracy: 0.7040 - val_loss:
s: 0.7646

Epoch 101/400
472/472 2s 5ms/step - accuracy: 0.8213 - loss: 0.3828 - val_accuracy: 0.7073 - val_loss:
s: 0.7368

Epoch 102/400

472/472 2s 5ms/step - accuracy: 0.8238 - loss: 0.3812 - val_accuracy: 0.7091 - val_loss: 0.7644
Epoch 103/400
472/472 2s 5ms/step - accuracy: 0.8220 - loss: 0.3873 - val_accuracy: 0.7090 - val_loss: 0.7553
Epoch 104/400
472/472 2s 5ms/step - accuracy: 0.8256 - loss: 0.3797 - val_accuracy: 0.7057 - val_loss: 0.7535
Epoch 105/400
472/472 2s 5ms/step - accuracy: 0.8223 - loss: 0.3812 - val_accuracy: 0.7108 - val_loss: 0.7569
Epoch 106/400
472/472 2s 5ms/step - accuracy: 0.8211 - loss: 0.3833 - val_accuracy: 0.7106 - val_loss: 0.7582
Epoch 107/400
472/472 2s 5ms/step - accuracy: 0.8242 - loss: 0.3818 - val_accuracy: 0.7066 - val_loss: 0.7714
Epoch 108/400
472/472 2s 5ms/step - accuracy: 0.8216 - loss: 0.3816 - val_accuracy: 0.7068 - val_loss: 0.7643
Epoch 109/400
472/472 2s 5ms/step - accuracy: 0.8243 - loss: 0.3777 - val_accuracy: 0.7090 - val_loss: 0.7714
Epoch 110/400
472/472 2s 5ms/step - accuracy: 0.8255 - loss: 0.3765 - val_accuracy: 0.7104 - val_loss: 0.7632
Epoch 111/400
472/472 2s 5ms/step - accuracy: 0.8246 - loss: 0.3797 - val_accuracy: 0.7136 - val_loss: 0.7403
Epoch 112/400
472/472 2s 5ms/step - accuracy: 0.8228 - loss: 0.3797 - val_accuracy: 0.7080 - val_loss: 0.7313
Epoch 113/400
472/472 2s 5ms/step - accuracy: 0.8258 - loss: 0.3764 - val_accuracy: 0.7109 - val_loss: 0.7463
Epoch 114/400
472/472 2s 4ms/step - accuracy: 0.8262 - loss: 0.3777 - val_accuracy: 0.7108 - val_loss: 0.7408
Epoch 115/400
472/472 2s 4ms/step - accuracy: 0.8269 - loss: 0.3764 - val_accuracy: 0.7103 - val_loss: 0.7448
Epoch 116/400
472/472 2s 4ms/step - accuracy: 0.8244 - loss: 0.3758 - val_accuracy: 0.7133 - val_loss: 0.7634
Epoch 117/400
472/472 3s 4ms/step - accuracy: 0.8241 - loss: 0.3772 - val_accuracy: 0.7099 - val_loss: 0.7495
Epoch 118/400
472/472 2s 4ms/step - accuracy: 0.8269 - loss: 0.3719 - val_accuracy: 0.7061 - val_loss: 0.7639
Epoch 119/400
472/472 2s 5ms/step - accuracy: 0.8280 - loss: 0.3720 - val_accuracy: 0.7084 - val_loss: 0.7853
Epoch 120/400
472/472 2s 5ms/step - accuracy: 0.8284 - loss: 0.3720 - val_accuracy: 0.7154 - val_loss: 0.7473
Epoch 121/400
472/472 2s 4ms/step - accuracy: 0.8274 - loss: 0.3737 - val_accuracy: 0.7068 - val_loss: 0.7729
Epoch 122/400
472/472 2s 4ms/step - accuracy: 0.8295 - loss: 0.3713 - val_accuracy: 0.7049 - val_loss: 0.7528
Epoch 123/400
472/472 2s 4ms/step - accuracy: 0.8291 - loss: 0.3723 - val_accuracy: 0.7179 - val_loss: 0.7421
Epoch 124/400
472/472 2s 5ms/step - accuracy: 0.8277 - loss: 0.3716 - val_accuracy: 0.7143 - val_loss: 0.7364
Epoch 125/400
472/472 2s 4ms/step - accuracy: 0.8289 - loss: 0.3686 - val_accuracy: 0.7115 - val_loss: 0.7573
Epoch 126/400
472/472 2s 4ms/step - accuracy: 0.8295 - loss: 0.3676 - val_accuracy: 0.7179 - val_loss: 0.7558
Epoch 127/400
472/472 2s 5ms/step - accuracy: 0.8299 - loss: 0.3662 - val_accuracy: 0.7137 - val_loss:

s: 0.7578
Epoch 128/400
472/472 2s 4ms/step - accuracy: 0.8296 - loss: 0.3692 - val_accuracy: 0.7199 - val_loss: 0.7263
s: 0.7423
Epoch 129/400
472/472 2s 4ms/step - accuracy: 0.8314 - loss: 0.3692 - val_accuracy: 0.7105 - val_loss: 0.7263
s: 0.7364
Epoch 130/400
472/472 2s 4ms/step - accuracy: 0.8296 - loss: 0.3704 - val_accuracy: 0.7131 - val_loss: 0.7263
s: 0.7494
Epoch 131/400
472/472 2s 4ms/step - accuracy: 0.8309 - loss: 0.3653 - val_accuracy: 0.7161 - val_loss: 0.7263
s: 0.7453
Epoch 132/400
472/472 2s 4ms/step - accuracy: 0.8288 - loss: 0.3720 - val_accuracy: 0.7049 - val_loss: 0.7263
s: 0.7833
Epoch 133/400
472/472 2s 4ms/step - accuracy: 0.8270 - loss: 0.3727 - val_accuracy: 0.7185 - val_loss: 0.7263
s: 0.7423
Epoch 134/400
472/472 2s 5ms/step - accuracy: 0.8344 - loss: 0.3626 - val_accuracy: 0.7192 - val_loss: 0.7284
s: 0.7284
Epoch 135/400
472/472 2s 5ms/step - accuracy: 0.8329 - loss: 0.3653 - val_accuracy: 0.7133 - val_loss: 0.7481
s: 0.7481
Epoch 136/400
472/472 2s 4ms/step - accuracy: 0.8313 - loss: 0.3650 - val_accuracy: 0.7168 - val_loss: 0.7519
s: 0.7519
Epoch 137/400
472/472 2s 4ms/step - accuracy: 0.8338 - loss: 0.3615 - val_accuracy: 0.7132 - val_loss: 0.7447
s: 0.7447
Epoch 138/400
472/472 2s 4ms/step - accuracy: 0.8308 - loss: 0.3640 - val_accuracy: 0.7220 - val_loss: 0.7248
s: 0.7248
Epoch 139/400
472/472 2s 4ms/step - accuracy: 0.8326 - loss: 0.3659 - val_accuracy: 0.7107 - val_loss: 0.7464
s: 0.7464
Epoch 140/400
472/472 2s 4ms/step - accuracy: 0.8319 - loss: 0.3656 - val_accuracy: 0.7134 - val_loss: 0.7624
s: 0.7624
Epoch 141/400
472/472 2s 5ms/step - accuracy: 0.8308 - loss: 0.3664 - val_accuracy: 0.7162 - val_loss: 0.7458
s: 0.7458
Epoch 142/400
472/472 2s 5ms/step - accuracy: 0.8330 - loss: 0.3641 - val_accuracy: 0.7013 - val_loss: 0.7724
s: 0.7724
Epoch 143/400
472/472 2s 5ms/step - accuracy: 0.8304 - loss: 0.3643 - val_accuracy: 0.7154 - val_loss: 0.7401
s: 0.7401
Epoch 144/400
472/472 2s 4ms/step - accuracy: 0.8334 - loss: 0.3648 - val_accuracy: 0.7158 - val_loss: 0.7534
s: 0.7534
Epoch 145/400
472/472 2s 4ms/step - accuracy: 0.8326 - loss: 0.3609 - val_accuracy: 0.7142 - val_loss: 0.7562
s: 0.7562
Epoch 146/400
472/472 2s 4ms/step - accuracy: 0.8331 - loss: 0.3616 - val_accuracy: 0.7091 - val_loss: 0.7669
s: 0.7669
Epoch 147/400
472/472 2s 4ms/step - accuracy: 0.8312 - loss: 0.3620 - val_accuracy: 0.7120 - val_loss: 0.7495
s: 0.7495
Epoch 148/400
472/472 2s 4ms/step - accuracy: 0.8334 - loss: 0.3643 - val_accuracy: 0.7142 - val_loss: 0.7558
s: 0.7558
Epoch 149/400
472/472 2s 5ms/step - accuracy: 0.8310 - loss: 0.3666 - val_accuracy: 0.7198 - val_loss: 0.7487
s: 0.7487
Epoch 150/400
472/472 2s 5ms/step - accuracy: 0.8363 - loss: 0.3563 - val_accuracy: 0.7202 - val_loss: 0.7263
s: 0.7263
Epoch 151/400
472/472 2s 4ms/step - accuracy: 0.8325 - loss: 0.3592 - val_accuracy: 0.7179 - val_loss: 0.7423
s: 0.7423
Epoch 152/400
472/472 2s 5ms/step - accuracy: 0.8349 - loss: 0.3580 - val_accuracy: 0.7153 - val_loss: 0.7403
s: 0.7403

Epoch 153/400
472/472 2s 4ms/step - accuracy: 0.8339 - loss: 0.3611 - val_accuracy: 0.7133 - val_loss:
s: 0.7533
Epoch 154/400
472/472 2s 4ms/step - accuracy: 0.8347 - loss: 0.3600 - val_accuracy: 0.7136 - val_loss:
s: 0.7345
Epoch 155/400
472/472 2s 4ms/step - accuracy: 0.8343 - loss: 0.3565 - val_accuracy: 0.7225 - val_loss:
s: 0.7198
Epoch 156/400
472/472 2s 4ms/step - accuracy: 0.8351 - loss: 0.3562 - val_accuracy: 0.7212 - val_loss:
s: 0.7436
Epoch 157/400
472/472 2s 5ms/step - accuracy: 0.8330 - loss: 0.3600 - val_accuracy: 0.7139 - val_loss:
s: 0.7439
Epoch 158/400
472/472 2s 4ms/step - accuracy: 0.8324 - loss: 0.3599 - val_accuracy: 0.7230 - val_loss:
s: 0.7220
Epoch 159/400
472/472 2s 4ms/step - accuracy: 0.8336 - loss: 0.3594 - val_accuracy: 0.7166 - val_loss:
s: 0.7711
Epoch 160/400
472/472 2s 4ms/step - accuracy: 0.8341 - loss: 0.3585 - val_accuracy: 0.7140 - val_loss:
s: 0.7451
Epoch 161/400
472/472 2s 4ms/step - accuracy: 0.8361 - loss: 0.3562 - val_accuracy: 0.7111 - val_loss:
s: 0.7736
Epoch 162/400
472/472 2s 5ms/step - accuracy: 0.8365 - loss: 0.3596 - val_accuracy: 0.7241 - val_loss:
s: 0.7161
Epoch 163/400
472/472 2s 4ms/step - accuracy: 0.8408 - loss: 0.3523 - val_accuracy: 0.7079 - val_loss:
s: 0.7497
Epoch 164/400
472/472 2s 4ms/step - accuracy: 0.8348 - loss: 0.3559 - val_accuracy: 0.7194 - val_loss:
s: 0.7424
Epoch 165/400
472/472 2s 5ms/step - accuracy: 0.8353 - loss: 0.3582 - val_accuracy: 0.7109 - val_loss:
s: 0.7405
Epoch 166/400
472/472 2s 4ms/step - accuracy: 0.8345 - loss: 0.3549 - val_accuracy: 0.7125 - val_loss:
s: 0.7754
Epoch 167/400
472/472 2s 4ms/step - accuracy: 0.8309 - loss: 0.3628 - val_accuracy: 0.7234 - val_loss:
s: 0.7497
Epoch 168/400
472/472 2s 4ms/step - accuracy: 0.8368 - loss: 0.3575 - val_accuracy: 0.7157 - val_loss:
s: 0.7502
Epoch 169/400
472/472 2s 4ms/step - accuracy: 0.8369 - loss: 0.3527 - val_accuracy: 0.7159 - val_loss:
s: 0.7445
Epoch 170/400
472/472 2s 5ms/step - accuracy: 0.8375 - loss: 0.3559 - val_accuracy: 0.7128 - val_loss:
s: 0.7575
Epoch 171/400
472/472 2s 4ms/step - accuracy: 0.8381 - loss: 0.3528 - val_accuracy: 0.7235 - val_loss:
s: 0.7333
Epoch 172/400
472/472 2s 5ms/step - accuracy: 0.8399 - loss: 0.3491 - val_accuracy: 0.7185 - val_loss:
s: 0.7339
Epoch 173/400
472/472 2s 5ms/step - accuracy: 0.8373 - loss: 0.3551 - val_accuracy: 0.7152 - val_loss:
s: 0.7414
Epoch 174/400
472/472 2s 4ms/step - accuracy: 0.8374 - loss: 0.3530 - val_accuracy: 0.7253 - val_loss:
s: 0.7214
Epoch 175/400
472/472 2s 4ms/step - accuracy: 0.8387 - loss: 0.3514 - val_accuracy: 0.7191 - val_loss:
s: 0.7695
Epoch 176/400
472/472 2s 4ms/step - accuracy: 0.8350 - loss: 0.3582 - val_accuracy: 0.7234 - val_loss:
s: 0.7435
Epoch 177/400
472/472 2s 4ms/step - accuracy: 0.8365 - loss: 0.3553 - val_accuracy: 0.7205 - val_loss:
s: 0.7630
Epoch 178/400

472/472 2s 5ms/step - accuracy: 0.8383 - loss: 0.3483 - val_accuracy: 0.7306 - val_loss: 0.7523
Epoch 179/400
472/472 2s 5ms/step - accuracy: 0.8406 - loss: 0.3501 - val_accuracy: 0.7191 - val_loss: 0.7186
Epoch 180/400
472/472 2s 5ms/step - accuracy: 0.8381 - loss: 0.3507 - val_accuracy: 0.7219 - val_loss: 0.7518
Epoch 181/400
472/472 2s 5ms/step - accuracy: 0.8374 - loss: 0.3533 - val_accuracy: 0.7202 - val_loss: 0.7411
Epoch 182/400
472/472 2s 4ms/step - accuracy: 0.8388 - loss: 0.3517 - val_accuracy: 0.7166 - val_loss: 0.7634
Epoch 183/400
472/472 2s 4ms/step - accuracy: 0.8394 - loss: 0.3493 - val_accuracy: 0.7253 - val_loss: 0.7269
Epoch 184/400
472/472 2s 4ms/step - accuracy: 0.8405 - loss: 0.3461 - val_accuracy: 0.7135 - val_loss: 0.7566
Epoch 185/400
472/472 2s 4ms/step - accuracy: 0.8386 - loss: 0.3517 - val_accuracy: 0.7197 - val_loss: 0.7379
Epoch 186/400
472/472 2s 5ms/step - accuracy: 0.8359 - loss: 0.3543 - val_accuracy: 0.7177 - val_loss: 0.7310
Epoch 187/400
472/472 2s 5ms/step - accuracy: 0.8364 - loss: 0.3522 - val_accuracy: 0.7281 - val_loss: 0.7508
Epoch 188/400
472/472 2s 5ms/step - accuracy: 0.8402 - loss: 0.3490 - val_accuracy: 0.7283 - val_loss: 0.7254
Epoch 189/400
472/472 2s 5ms/step - accuracy: 0.8403 - loss: 0.3470 - val_accuracy: 0.7161 - val_loss: 0.7399
Epoch 190/400
472/472 2s 5ms/step - accuracy: 0.8392 - loss: 0.3514 - val_accuracy: 0.7267 - val_loss: 0.7230
Epoch 191/400
472/472 2s 4ms/step - accuracy: 0.8398 - loss: 0.3485 - val_accuracy: 0.7140 - val_loss: 0.7500
Epoch 192/400
472/472 2s 4ms/step - accuracy: 0.8373 - loss: 0.3501 - val_accuracy: 0.7276 - val_loss: 0.7394
Epoch 193/400
472/472 2s 4ms/step - accuracy: 0.8396 - loss: 0.3469 - val_accuracy: 0.7285 - val_loss: 0.7275
Epoch 194/400
472/472 2s 5ms/step - accuracy: 0.8412 - loss: 0.3476 - val_accuracy: 0.7222 - val_loss: 0.7301
Epoch 195/400
472/472 2s 5ms/step - accuracy: 0.8402 - loss: 0.3500 - val_accuracy: 0.7263 - val_loss: 0.7307
Epoch 196/400
472/472 2s 4ms/step - accuracy: 0.8391 - loss: 0.3504 - val_accuracy: 0.7229 - val_loss: 0.7525
Epoch 197/400
472/472 2s 4ms/step - accuracy: 0.8385 - loss: 0.3492 - val_accuracy: 0.7231 - val_loss: 0.7283
Epoch 198/400
472/472 2s 4ms/step - accuracy: 0.8417 - loss: 0.3443 - val_accuracy: 0.7257 - val_loss: 0.7219
Epoch 199/400
472/472 2s 4ms/step - accuracy: 0.8414 - loss: 0.3483 - val_accuracy: 0.7299 - val_loss: 0.7251
Epoch 200/400
472/472 2s 4ms/step - accuracy: 0.8411 - loss: 0.3444 - val_accuracy: 0.7263 - val_loss: 0.7267
Epoch 201/400
472/472 2s 4ms/step - accuracy: 0.8412 - loss: 0.3459 - val_accuracy: 0.7246 - val_loss: 0.7180
Epoch 202/400
472/472 2s 5ms/step - accuracy: 0.8411 - loss: 0.3422 - val_accuracy: 0.7128 - val_loss: 0.7512
Epoch 203/400
472/472 2s 5ms/step - accuracy: 0.8392 - loss: 0.3460 - val_accuracy: 0.7246 - val_loss:

s: 0.7362
Epoch 204/400
472/472 2s 4ms/step - accuracy: 0.8389 - loss: 0.3478 - val_accuracy: 0.7278 - val_loss: 0.7278
s: 0.7213
Epoch 205/400
472/472 2s 4ms/step - accuracy: 0.8408 - loss: 0.3479 - val_accuracy: 0.7157 - val_loss: 0.7157
s: 0.7488
Epoch 206/400
472/472 2s 4ms/step - accuracy: 0.8424 - loss: 0.3444 - val_accuracy: 0.7183 - val_loss: 0.7183
s: 0.7274
Epoch 207/400
472/472 2s 4ms/step - accuracy: 0.8397 - loss: 0.3487 - val_accuracy: 0.7163 - val_loss: 0.7163
s: 0.7423
Epoch 208/400
472/472 2s 4ms/step - accuracy: 0.8410 - loss: 0.3452 - val_accuracy: 0.7206 - val_loss: 0.7206
s: 0.7488
Epoch 209/400
472/472 2s 4ms/step - accuracy: 0.8430 - loss: 0.3438 - val_accuracy: 0.7305 - val_loss: 0.7305
s: 0.7257
Epoch 210/400
472/472 2s 5ms/step - accuracy: 0.8430 - loss: 0.3402 - val_accuracy: 0.7279 - val_loss: 0.7279
s: 0.7304
Epoch 211/400
472/472 2s 4ms/step - accuracy: 0.8417 - loss: 0.3427 - val_accuracy: 0.7185 - val_loss: 0.7185
s: 0.7275
Epoch 212/400
472/472 2s 4ms/step - accuracy: 0.8421 - loss: 0.3435 - val_accuracy: 0.7241 - val_loss: 0.7241
s: 0.7266
Epoch 213/400
472/472 2s 5ms/step - accuracy: 0.8394 - loss: 0.3468 - val_accuracy: 0.7252 - val_loss: 0.7252
s: 0.7179
Epoch 214/400
472/472 2s 4ms/step - accuracy: 0.8429 - loss: 0.3421 - val_accuracy: 0.7251 - val_loss: 0.7251
s: 0.7139
Epoch 215/400
472/472 2s 4ms/step - accuracy: 0.8439 - loss: 0.3413 - val_accuracy: 0.7264 - val_loss: 0.7264
s: 0.7221
Epoch 216/400
472/472 2s 4ms/step - accuracy: 0.8412 - loss: 0.3458 - val_accuracy: 0.7283 - val_loss: 0.7283
s: 0.7174
Epoch 217/400
472/472 2s 5ms/step - accuracy: 0.8414 - loss: 0.3440 - val_accuracy: 0.7225 - val_loss: 0.7225
s: 0.7583
Epoch 218/400
472/472 2s 5ms/step - accuracy: 0.8417 - loss: 0.3433 - val_accuracy: 0.7242 - val_loss: 0.7242
s: 0.7395
Epoch 219/400
472/472 2s 4ms/step - accuracy: 0.8414 - loss: 0.3443 - val_accuracy: 0.7308 - val_loss: 0.7308
s: 0.7204
Epoch 220/400
472/472 2s 4ms/step - accuracy: 0.8473 - loss: 0.3368 - val_accuracy: 0.7255 - val_loss: 0.7255
s: 0.7470
Epoch 221/400
472/472 2s 4ms/step - accuracy: 0.8421 - loss: 0.3451 - val_accuracy: 0.7259 - val_loss: 0.7259
s: 0.7475
Epoch 222/400
472/472 2s 4ms/step - accuracy: 0.8425 - loss: 0.3416 - val_accuracy: 0.7282 - val_loss: 0.7282
s: 0.7391
Epoch 223/400
472/472 2s 4ms/step - accuracy: 0.8432 - loss: 0.3419 - val_accuracy: 0.7199 - val_loss: 0.7199
s: 0.7385
Epoch 224/400
472/472 2s 4ms/step - accuracy: 0.8426 - loss: 0.3384 - val_accuracy: 0.7249 - val_loss: 0.7249
s: 0.7326
Epoch 225/400
472/472 2s 5ms/step - accuracy: 0.8421 - loss: 0.3442 - val_accuracy: 0.7322 - val_loss: 0.7322
s: 0.7181
Epoch 226/400
472/472 2s 5ms/step - accuracy: 0.8456 - loss: 0.3384 - val_accuracy: 0.7271 - val_loss: 0.7271
s: 0.7308
Epoch 227/400
472/472 2s 5ms/step - accuracy: 0.8435 - loss: 0.3407 - val_accuracy: 0.7199 - val_loss: 0.7199
s: 0.7354
Epoch 228/400
472/472 2s 5ms/step - accuracy: 0.8449 - loss: 0.3385 - val_accuracy: 0.7214 - val_loss: 0.7214
s: 0.7524

Epoch 229/400
472/472 2s 4ms/step - accuracy: 0.8452 - loss: 0.3399 - val_accuracy: 0.7188 - val_loss: 0.7392
Epoch 230/400
472/472 2s 4ms/step - accuracy: 0.8431 - loss: 0.3432 - val_accuracy: 0.7231 - val_loss: 0.7342
Epoch 231/400
472/472 2s 4ms/step - accuracy: 0.8445 - loss: 0.3417 - val_accuracy: 0.7303 - val_loss: 0.7325
Epoch 232/400
472/472 2s 5ms/step - accuracy: 0.8439 - loss: 0.3400 - val_accuracy: 0.7304 - val_loss: 0.7091
Epoch 233/400
472/472 2s 5ms/step - accuracy: 0.8459 - loss: 0.3415 - val_accuracy: 0.7283 - val_loss: 0.7221
Epoch 234/400
472/472 2s 4ms/step - accuracy: 0.8438 - loss: 0.3398 - val_accuracy: 0.7261 - val_loss: 0.7264
Epoch 235/400
472/472 2s 4ms/step - accuracy: 0.8435 - loss: 0.3411 - val_accuracy: 0.7274 - val_loss: 0.7156
Epoch 236/400
472/472 2s 4ms/step - accuracy: 0.8470 - loss: 0.3332 - val_accuracy: 0.7251 - val_loss: 0.7394
Epoch 237/400
472/472 2s 4ms/step - accuracy: 0.8437 - loss: 0.3436 - val_accuracy: 0.7354 - val_loss: 0.7128
Epoch 238/400
472/472 2s 4ms/step - accuracy: 0.8434 - loss: 0.3412 - val_accuracy: 0.7224 - val_loss: 0.7205
Epoch 239/400
472/472 2s 5ms/step - accuracy: 0.8431 - loss: 0.3406 - val_accuracy: 0.7244 - val_loss: 0.7318
Epoch 240/400
472/472 3s 5ms/step - accuracy: 0.8429 - loss: 0.3407 - val_accuracy: 0.7266 - val_loss: 0.7269
Epoch 241/400
472/472 2s 4ms/step - accuracy: 0.8453 - loss: 0.3364 - val_accuracy: 0.7279 - val_loss: 0.7135
Epoch 242/400
472/472 2s 4ms/step - accuracy: 0.8450 - loss: 0.3356 - val_accuracy: 0.7258 - val_loss: 0.7270
Epoch 243/400
472/472 2s 5ms/step - accuracy: 0.8467 - loss: 0.3350 - val_accuracy: 0.7256 - val_loss: 0.7438
Epoch 244/400
472/472 2s 5ms/step - accuracy: 0.8439 - loss: 0.3396 - val_accuracy: 0.7244 - val_loss: 0.7410
Epoch 245/400
472/472 2s 5ms/step - accuracy: 0.8455 - loss: 0.3388 - val_accuracy: 0.7241 - val_loss: 0.7296
Epoch 246/400
472/472 2s 5ms/step - accuracy: 0.8429 - loss: 0.3404 - val_accuracy: 0.7231 - val_loss: 0.7293
Epoch 247/400
472/472 2s 5ms/step - accuracy: 0.8454 - loss: 0.3371 - val_accuracy: 0.7258 - val_loss: 0.7509
Epoch 248/400
472/472 2s 4ms/step - accuracy: 0.8443 - loss: 0.3369 - val_accuracy: 0.7232 - val_loss: 0.7338
Epoch 249/400
472/472 2s 4ms/step - accuracy: 0.8415 - loss: 0.3401 - val_accuracy: 0.7268 - val_loss: 0.7261
Epoch 250/400
472/472 2s 4ms/step - accuracy: 0.8443 - loss: 0.3387 - val_accuracy: 0.7351 - val_loss: 0.7260
Epoch 251/400
472/472 2s 4ms/step - accuracy: 0.8483 - loss: 0.3342 - val_accuracy: 0.7319 - val_loss: 0.7120
Epoch 252/400
472/472 2s 4ms/step - accuracy: 0.8470 - loss: 0.3363 - val_accuracy: 0.7336 - val_loss: 0.7294
Epoch 253/400
472/472 2s 4ms/step - accuracy: 0.8452 - loss: 0.3399 - val_accuracy: 0.7327 - val_loss: 0.7047
Epoch 254/400

472/472 2s 5ms/step - accuracy: 0.8477 - loss: 0.3382 - val_accuracy: 0.7299 - val_loss: 0.7127
Epoch 255/400
472/472 2s 5ms/step - accuracy: 0.8479 - loss: 0.3311 - val_accuracy: 0.7241 - val_loss: 0.7250
Epoch 256/400
472/472 2s 4ms/step - accuracy: 0.8442 - loss: 0.3383 - val_accuracy: 0.7231 - val_loss: 0.7396
Epoch 257/400
472/472 2s 4ms/step - accuracy: 0.8454 - loss: 0.3360 - val_accuracy: 0.7263 - val_loss: 0.7042
Epoch 258/400
472/472 2s 4ms/step - accuracy: 0.8455 - loss: 0.3321 - val_accuracy: 0.7271 - val_loss: 0.7183
Epoch 259/400
472/472 2s 4ms/step - accuracy: 0.8460 - loss: 0.3343 - val_accuracy: 0.7261 - val_loss: 0.7333
Epoch 260/400
472/472 2s 4ms/step - accuracy: 0.8464 - loss: 0.3336 - val_accuracy: 0.7227 - val_loss: 0.7384
Epoch 261/400
472/472 2s 4ms/step - accuracy: 0.8455 - loss: 0.3339 - val_accuracy: 0.7303 - val_loss: 0.7258
Epoch 262/400
472/472 2s 5ms/step - accuracy: 0.8452 - loss: 0.3364 - val_accuracy: 0.7310 - val_loss: 0.7064
Epoch 263/400
472/472 2s 4ms/step - accuracy: 0.8467 - loss: 0.3319 - val_accuracy: 0.7315 - val_loss: 0.7316
Epoch 264/400
472/472 2s 4ms/step - accuracy: 0.8485 - loss: 0.3324 - val_accuracy: 0.7336 - val_loss: 0.7330
Epoch 265/400
472/472 2s 4ms/step - accuracy: 0.8455 - loss: 0.3360 - val_accuracy: 0.7271 - val_loss: 0.7411
Epoch 266/400
472/472 2s 4ms/step - accuracy: 0.8471 - loss: 0.3316 - val_accuracy: 0.7324 - val_loss: 0.7135
Epoch 267/400
472/472 2s 4ms/step - accuracy: 0.8463 - loss: 0.3359 - val_accuracy: 0.7299 - val_loss: 0.7390
Epoch 268/400
472/472 2s 4ms/step - accuracy: 0.8476 - loss: 0.3365 - val_accuracy: 0.7264 - val_loss: 0.7173
Epoch 269/400
472/472 2s 5ms/step - accuracy: 0.8485 - loss: 0.3303 - val_accuracy: 0.7256 - val_loss: 0.7334
Epoch 270/400
472/472 2s 5ms/step - accuracy: 0.8455 - loss: 0.3310 - val_accuracy: 0.7306 - val_loss: 0.7111
Epoch 271/400
472/472 2s 4ms/step - accuracy: 0.8459 - loss: 0.3375 - val_accuracy: 0.7288 - val_loss: 0.7231
Epoch 272/400
472/472 3s 4ms/step - accuracy: 0.8464 - loss: 0.3336 - val_accuracy: 0.7278 - val_loss: 0.7266
Epoch 273/400
472/472 2s 4ms/step - accuracy: 0.8438 - loss: 0.3370 - val_accuracy: 0.7357 - val_loss: 0.7397
Epoch 274/400
472/472 2s 4ms/step - accuracy: 0.8476 - loss: 0.3389 - val_accuracy: 0.7297 - val_loss: 0.7274
Epoch 275/400
472/472 2s 4ms/step - accuracy: 0.8485 - loss: 0.3326 - val_accuracy: 0.7217 - val_loss: 0.7230
Epoch 276/400
472/472 2s 5ms/step - accuracy: 0.8450 - loss: 0.3341 - val_accuracy: 0.7260 - val_loss: 0.7397
Epoch 277/400
472/472 2s 5ms/step - accuracy: 0.8500 - loss: 0.3277 - val_accuracy: 0.7291 - val_loss: 0.7388
Epoch 278/400
472/472 2s 4ms/step - accuracy: 0.8474 - loss: 0.3315 - val_accuracy: 0.7302 - val_loss: 0.7086
Epoch 279/400
472/472 2s 4ms/step - accuracy: 0.8476 - loss: 0.3316 - val_accuracy: 0.7252 - val_loss:

s: 0.7428
Epoch 280/400
472/472 2s 4ms/step - accuracy: 0.8473 - loss: 0.3294 - val_accuracy: 0.7192 - val_loss: 0.7319
Epoch 281/400
472/472 2s 4ms/step - accuracy: 0.8454 - loss: 0.3353 - val_accuracy: 0.7304 - val_loss: 0.7292
Epoch 282/400
472/472 2s 4ms/step - accuracy: 0.8463 - loss: 0.3307 - val_accuracy: 0.7313 - val_loss: 0.7123
Epoch 283/400
472/472 2s 4ms/step - accuracy: 0.8488 - loss: 0.3336 - val_accuracy: 0.7272 - val_loss: 0.7310
s: 0.7152
Epoch 284/400
472/472 2s 5ms/step - accuracy: 0.8460 - loss: 0.3348 - val_accuracy: 0.7260 - val_loss: 0.7473
s: 0.7473
Epoch 285/400
472/472 2s 5ms/step - accuracy: 0.8468 - loss: 0.3290 - val_accuracy: 0.7200 - val_loss: 0.7327
s: 0.7327
Epoch 286/400
472/472 2s 4ms/step - accuracy: 0.8476 - loss: 0.3312 - val_accuracy: 0.7334 - val_loss: 0.7233
s: 0.7233
Epoch 288/400
472/472 2s 4ms/step - accuracy: 0.8467 - loss: 0.3334 - val_accuracy: 0.7285 - val_loss: 0.7128
s: 0.7128
Epoch 289/400
472/472 2s 4ms/step - accuracy: 0.8474 - loss: 0.3327 - val_accuracy: 0.7317 - val_loss: 0.6889
s: 0.6889
Epoch 290/400
472/472 2s 5ms/step - accuracy: 0.8483 - loss: 0.3327 - val_accuracy: 0.7305 - val_loss: 0.7126
s: 0.7126
Epoch 291/400
472/472 2s 5ms/step - accuracy: 0.8469 - loss: 0.3333 - val_accuracy: 0.7244 - val_loss: 0.7419
s: 0.7419
Epoch 292/400
472/472 2s 5ms/step - accuracy: 0.8493 - loss: 0.3300 - val_accuracy: 0.7275 - val_loss: 0.7258
s: 0.7258
Epoch 293/400
472/472 2s 4ms/step - accuracy: 0.8477 - loss: 0.3299 - val_accuracy: 0.7228 - val_loss: 0.7344
s: 0.7344
Epoch 294/400
472/472 2s 4ms/step - accuracy: 0.8481 - loss: 0.3285 - val_accuracy: 0.7340 - val_loss: 0.7198
s: 0.7198
Epoch 295/400
472/472 2s 4ms/step - accuracy: 0.8487 - loss: 0.3290 - val_accuracy: 0.7278 - val_loss: 0.6988
s: 0.6988
Epoch 296/400
472/472 2s 4ms/step - accuracy: 0.8474 - loss: 0.3332 - val_accuracy: 0.7265 - val_loss: 0.7245
s: 0.7245
Epoch 297/400
472/472 2s 4ms/step - accuracy: 0.8480 - loss: 0.3299 - val_accuracy: 0.7320 - val_loss: 0.7340
s: 0.7340
Epoch 298/400
472/472 2s 4ms/step - accuracy: 0.8483 - loss: 0.3314 - val_accuracy: 0.7363 - val_loss: 0.7312
s: 0.7312
Epoch 299/400
472/472 2s 5ms/step - accuracy: 0.8504 - loss: 0.3288 - val_accuracy: 0.7269 - val_loss: 0.7152
s: 0.7152
Epoch 300/400
472/472 2s 5ms/step - accuracy: 0.8478 - loss: 0.3303 - val_accuracy: 0.7283 - val_loss: 0.7447
s: 0.7447
Epoch 301/400
472/472 2s 4ms/step - accuracy: 0.8493 - loss: 0.3283 - val_accuracy: 0.7296 - val_loss: 0.7316
s: 0.7316
Epoch 302/400
472/472 2s 5ms/step - accuracy: 0.8492 - loss: 0.3303 - val_accuracy: 0.7280 - val_loss: 0.7139
s: 0.7139
Epoch 303/400
472/472 2s 4ms/step - accuracy: 0.8487 - loss: 0.3297 - val_accuracy: 0.7323 - val_loss: 0.7268
s: 0.7268
Epoch 304/400
472/472 2s 4ms/step - accuracy: 0.8500 - loss: 0.3279 - val_accuracy: 0.7334 - val_loss: 0.7295
s: 0.7295

Epoch 305/400
472/472 2s 4ms/step - accuracy: 0.8485 - loss: 0.3287 - val_accuracy: 0.7286 - val_loss:
s: 0.7315
Epoch 306/400
472/472 2s 5ms/step - accuracy: 0.8488 - loss: 0.3282 - val_accuracy: 0.7368 - val_loss:
s: 0.7207
Epoch 307/400
472/472 2s 5ms/step - accuracy: 0.8474 - loss: 0.3295 - val_accuracy: 0.7277 - val_loss:
s: 0.7377
Epoch 308/400
472/472 2s 4ms/step - accuracy: 0.8479 - loss: 0.3293 - val_accuracy: 0.7264 - val_loss:
s: 0.7295
Epoch 309/400
472/472 2s 4ms/step - accuracy: 0.8484 - loss: 0.3285 - val_accuracy: 0.7312 - val_loss:
s: 0.7112
Epoch 310/400
472/472 2s 4ms/step - accuracy: 0.8494 - loss: 0.3293 - val_accuracy: 0.7285 - val_loss:
s: 0.7079
Epoch 311/400
472/472 2s 4ms/step - accuracy: 0.8493 - loss: 0.3285 - val_accuracy: 0.7342 - val_loss:
s: 0.7190
Epoch 312/400
472/472 2s 4ms/step - accuracy: 0.8492 - loss: 0.3288 - val_accuracy: 0.7303 - val_loss:
s: 0.7303
Epoch 313/400
472/472 2s 4ms/step - accuracy: 0.8479 - loss: 0.3298 - val_accuracy: 0.7317 - val_loss:
s: 0.7140
Epoch 314/400
472/472 2s 5ms/step - accuracy: 0.8504 - loss: 0.3281 - val_accuracy: 0.7246 - val_loss:
s: 0.7208
Epoch 315/400
472/472 2s 5ms/step - accuracy: 0.8476 - loss: 0.3291 - val_accuracy: 0.7334 - val_loss:
s: 0.7176
Epoch 316/400
472/472 2s 5ms/step - accuracy: 0.8488 - loss: 0.3277 - val_accuracy: 0.7328 - val_loss:
s: 0.7169
Epoch 317/400
472/472 2s 4ms/step - accuracy: 0.8514 - loss: 0.3238 - val_accuracy: 0.7339 - val_loss:
s: 0.7020
Epoch 318/400
472/472 2s 4ms/step - accuracy: 0.8470 - loss: 0.3304 - val_accuracy: 0.7321 - val_loss:
s: 0.7367
Epoch 319/400
472/472 2s 4ms/step - accuracy: 0.8506 - loss: 0.3241 - val_accuracy: 0.7371 - val_loss:
s: 0.7240
Epoch 320/400
472/472 2s 4ms/step - accuracy: 0.8517 - loss: 0.3252 - val_accuracy: 0.7312 - val_loss:
s: 0.7266
Epoch 321/400
472/472 2s 5ms/step - accuracy: 0.8485 - loss: 0.3272 - val_accuracy: 0.7258 - val_loss:
s: 0.7145
Epoch 322/400
472/472 2s 5ms/step - accuracy: 0.8486 - loss: 0.3260 - val_accuracy: 0.7357 - val_loss:
s: 0.7081
Epoch 323/400
472/472 2s 4ms/step - accuracy: 0.8514 - loss: 0.3239 - val_accuracy: 0.7314 - val_loss:
s: 0.7280
Epoch 324/400
472/472 2s 4ms/step - accuracy: 0.8496 - loss: 0.3281 - val_accuracy: 0.7311 - val_loss:
s: 0.7125
Epoch 325/400
472/472 2s 4ms/step - accuracy: 0.8473 - loss: 0.3292 - val_accuracy: 0.7322 - val_loss:
s: 0.7220
Epoch 326/400
472/472 2s 4ms/step - accuracy: 0.8505 - loss: 0.3293 - val_accuracy: 0.7404 - val_loss:
s: 0.7153
Epoch 327/400
472/472 2s 5ms/step - accuracy: 0.8511 - loss: 0.3258 - val_accuracy: 0.7236 - val_loss:
s: 0.7414
Epoch 328/400
472/472 2s 5ms/step - accuracy: 0.8492 - loss: 0.3274 - val_accuracy: 0.7265 - val_loss:
s: 0.7179
Epoch 329/400
472/472 2s 5ms/step - accuracy: 0.8511 - loss: 0.3295 - val_accuracy: 0.7263 - val_loss:
s: 0.7448
Epoch 330/400

472/472 2s 5ms/step - accuracy: 0.8516 - loss: 0.3264 - val_accuracy: 0.7398 - val_loss: 0.6889
Epoch 331/400
472/472 2s 5ms/step - accuracy: 0.8539 - loss: 0.3228 - val_accuracy: 0.7321 - val_loss: 0.7354
Epoch 332/400
472/472 2s 5ms/step - accuracy: 0.8495 - loss: 0.3275 - val_accuracy: 0.7391 - val_loss: 0.6960
Epoch 333/400
472/472 2s 5ms/step - accuracy: 0.8506 - loss: 0.3267 - val_accuracy: 0.7355 - val_loss: 0.6918
Epoch 334/400
472/472 2s 5ms/step - accuracy: 0.8516 - loss: 0.3259 - val_accuracy: 0.7318 - val_loss: 0.7100
Epoch 335/400
472/472 2s 5ms/step - accuracy: 0.8502 - loss: 0.3291 - val_accuracy: 0.7316 - val_loss: 0.7068
Epoch 336/400
472/472 2s 5ms/step - accuracy: 0.8499 - loss: 0.3262 - val_accuracy: 0.7319 - val_loss: 0.7299
Epoch 337/400
472/472 2s 5ms/step - accuracy: 0.8515 - loss: 0.3261 - val_accuracy: 0.7325 - val_loss: 0.7055
Epoch 338/400
472/472 2s 5ms/step - accuracy: 0.8509 - loss: 0.3256 - val_accuracy: 0.7367 - val_loss: 0.7015
Epoch 339/400
472/472 2s 5ms/step - accuracy: 0.8517 - loss: 0.3238 - val_accuracy: 0.7318 - val_loss: 0.7051
Epoch 340/400
472/472 2s 5ms/step - accuracy: 0.8478 - loss: 0.3294 - val_accuracy: 0.7283 - val_loss: 0.7262
Epoch 341/400
472/472 2s 5ms/step - accuracy: 0.8509 - loss: 0.3259 - val_accuracy: 0.7301 - val_loss: 0.7274
Epoch 342/400
472/472 2s 5ms/step - accuracy: 0.8488 - loss: 0.3256 - val_accuracy: 0.7251 - val_loss: 0.7336
Epoch 343/400
472/472 2s 5ms/step - accuracy: 0.8498 - loss: 0.3235 - val_accuracy: 0.7341 - val_loss: 0.7321
Epoch 344/400
472/472 2s 5ms/step - accuracy: 0.8530 - loss: 0.3233 - val_accuracy: 0.7341 - val_loss: 0.7255
Epoch 345/400
472/472 2s 5ms/step - accuracy: 0.8507 - loss: 0.3263 - val_accuracy: 0.7303 - val_loss: 0.7264
Epoch 346/400
472/472 2s 5ms/step - accuracy: 0.8500 - loss: 0.3263 - val_accuracy: 0.7341 - val_loss: 0.7206
Epoch 347/400
472/472 3s 5ms/step - accuracy: 0.8519 - loss: 0.3248 - val_accuracy: 0.7321 - val_loss: 0.7322
Epoch 348/400
472/472 2s 4ms/step - accuracy: 0.8511 - loss: 0.3245 - val_accuracy: 0.7408 - val_loss: 0.7113
Epoch 349/400
472/472 2s 5ms/step - accuracy: 0.8539 - loss: 0.3206 - val_accuracy: 0.7350 - val_loss: 0.7109
Epoch 350/400
472/472 3s 5ms/step - accuracy: 0.8502 - loss: 0.3273 - val_accuracy: 0.7344 - val_loss: 0.6916
Epoch 351/400
472/472 3s 7ms/step - accuracy: 0.8534 - loss: 0.3214 - val_accuracy: 0.7318 - val_loss: 0.7153
Epoch 352/400
472/472 3s 6ms/step - accuracy: 0.8508 - loss: 0.3271 - val_accuracy: 0.7287 - val_loss: 0.7078
Epoch 353/400
472/472 2s 5ms/step - accuracy: 0.8531 - loss: 0.3234 - val_accuracy: 0.7356 - val_loss: 0.6965
Epoch 354/400
472/472 2s 5ms/step - accuracy: 0.8518 - loss: 0.3262 - val_accuracy: 0.7332 - val_loss: 0.7303
Epoch 355/400
472/472 2s 5ms/step - accuracy: 0.8524 - loss: 0.3232 - val_accuracy: 0.7363 - val_loss:

s: 0.6991
Epoch 356/400
472/472 2s 5ms/step - accuracy: 0.8525 - loss: 0.3221 - val_accuracy: 0.7310 - val_loss: 0.3221
s: 0.7220
Epoch 357/400
472/472 2s 5ms/step - accuracy: 0.8518 - loss: 0.3226 - val_accuracy: 0.7336 - val_loss: 0.3226
s: 0.7245
Epoch 358/400
472/472 2s 4ms/step - accuracy: 0.8489 - loss: 0.3282 - val_accuracy: 0.7384 - val_loss: 0.3282
s: 0.7102
Epoch 359/400
472/472 2s 4ms/step - accuracy: 0.8546 - loss: 0.3190 - val_accuracy: 0.7343 - val_loss: 0.3190
s: 0.7298
Epoch 360/400
472/472 2s 5ms/step - accuracy: 0.8547 - loss: 0.3185 - val_accuracy: 0.7342 - val_loss: 0.3185
s: 0.7108
Epoch 361/400
472/472 2s 5ms/step - accuracy: 0.8527 - loss: 0.3213 - val_accuracy: 0.7357 - val_loss: 0.3213
s: 0.7172
Epoch 362/400
472/472 2s 5ms/step - accuracy: 0.8523 - loss: 0.3206 - val_accuracy: 0.7305 - val_loss: 0.3206
s: 0.7210
Epoch 363/400
472/472 2s 5ms/step - accuracy: 0.8527 - loss: 0.3200 - val_accuracy: 0.7309 - val_loss: 0.3200
s: 0.7291
Epoch 364/400
472/472 2s 5ms/step - accuracy: 0.8510 - loss: 0.3249 - val_accuracy: 0.7363 - val_loss: 0.3249
s: 0.6984
Epoch 365/400
472/472 2s 5ms/step - accuracy: 0.8519 - loss: 0.3218 - val_accuracy: 0.7329 - val_loss: 0.3218
s: 0.7004
Epoch 366/400
472/472 2s 5ms/step - accuracy: 0.8538 - loss: 0.3227 - val_accuracy: 0.7313 - val_loss: 0.3227
s: 0.7196
Epoch 367/400
472/472 2s 5ms/step - accuracy: 0.8501 - loss: 0.3244 - val_accuracy: 0.7304 - val_loss: 0.3244
s: 0.7269
Epoch 368/400
472/472 2s 5ms/step - accuracy: 0.8513 - loss: 0.3236 - val_accuracy: 0.7303 - val_loss: 0.3236
s: 0.7423
Epoch 369/400
472/472 2s 5ms/step - accuracy: 0.8531 - loss: 0.3235 - val_accuracy: 0.7367 - val_loss: 0.3235
s: 0.7193
Epoch 370/400
472/472 2s 5ms/step - accuracy: 0.8547 - loss: 0.3188 - val_accuracy: 0.7301 - val_loss: 0.3188
s: 0.7228
Epoch 371/400
472/472 3s 5ms/step - accuracy: 0.8510 - loss: 0.3245 - val_accuracy: 0.7358 - val_loss: 0.3245
s: 0.7154
Epoch 372/400
472/472 3s 5ms/step - accuracy: 0.8550 - loss: 0.3216 - val_accuracy: 0.7330 - val_loss: 0.3216
s: 0.7283
Epoch 373/400
472/472 2s 5ms/step - accuracy: 0.8550 - loss: 0.3200 - val_accuracy: 0.7326 - val_loss: 0.3200
s: 0.7187
Epoch 374/400
472/472 2s 5ms/step - accuracy: 0.8524 - loss: 0.3226 - val_accuracy: 0.7321 - val_loss: 0.3226
s: 0.6891
Epoch 375/400
472/472 3s 6ms/step - accuracy: 0.8525 - loss: 0.3217 - val_accuracy: 0.7253 - val_loss: 0.3217
s: 0.7412
Epoch 376/400
472/472 2s 5ms/step - accuracy: 0.8538 - loss: 0.3208 - val_accuracy: 0.7257 - val_loss: 0.3208
s: 0.7306
Epoch 377/400
472/472 2s 5ms/step - accuracy: 0.8544 - loss: 0.3208 - val_accuracy: 0.7329 - val_loss: 0.3208
s: 0.7220
Epoch 378/400
472/472 2s 5ms/step - accuracy: 0.8506 - loss: 0.3247 - val_accuracy: 0.7339 - val_loss: 0.3247
s: 0.7229
Epoch 379/400
472/472 2s 5ms/step - accuracy: 0.8522 - loss: 0.3192 - val_accuracy: 0.7303 - val_loss: 0.3192
s: 0.7217
Epoch 380/400
472/472 2s 5ms/step - accuracy: 0.8529 - loss: 0.3177 - val_accuracy: 0.7354 - val_loss: 0.3177
s: 0.7044

```
Epoch 381/400
472/472 2s 5ms/step - accuracy: 0.8507 - loss: 0.3225 - val_accuracy: 0.7214 - val_loss: 0.7361
Epoch 382/400
472/472 2s 5ms/step - accuracy: 0.8520 - loss: 0.3224 - val_accuracy: 0.7352 - val_loss: 0.7007
Epoch 383/400
472/472 3s 6ms/step - accuracy: 0.8533 - loss: 0.3201 - val_accuracy: 0.7400 - val_loss: 0.7275
s: 0.7275
Epoch 384/400
472/472 2s 5ms/step - accuracy: 0.8538 - loss: 0.3180 - val_accuracy: 0.7303 - val_loss: 0.7152
s: 0.7152
Epoch 385/400
472/472 2s 5ms/step - accuracy: 0.8511 - loss: 0.3218 - val_accuracy: 0.7339 - val_loss: 0.7279
s: 0.7279
Epoch 386/400
472/472 2s 5ms/step - accuracy: 0.8528 - loss: 0.3182 - val_accuracy: 0.7365 - val_loss: 0.7125
s: 0.7125
Epoch 387/400
472/472 2s 5ms/step - accuracy: 0.8504 - loss: 0.3201 - val_accuracy: 0.7363 - val_loss: 0.7177
s: 0.7177
Epoch 388/400
472/472 2s 5ms/step - accuracy: 0.8557 - loss: 0.3163 - val_accuracy: 0.7292 - val_loss: 0.7097
s: 0.7097
Epoch 389/400
472/472 2s 5ms/step - accuracy: 0.8524 - loss: 0.3211 - val_accuracy: 0.7367 - val_loss: 0.7113
s: 0.7113
Epoch 390/400
472/472 2s 5ms/step - accuracy: 0.8572 - loss: 0.3139 - val_accuracy: 0.7360 - val_loss: 0.7128
s: 0.7128
Epoch 391/400
472/472 2s 5ms/step - accuracy: 0.8553 - loss: 0.3187 - val_accuracy: 0.7360 - val_loss: 0.7060
s: 0.7060
Epoch 392/400
472/472 2s 5ms/step - accuracy: 0.8557 - loss: 0.3162 - val_accuracy: 0.7423 - val_loss: 0.7140
s: 0.7140
Epoch 393/400
472/472 2s 5ms/step - accuracy: 0.8515 - loss: 0.3219 - val_accuracy: 0.7413 - val_loss: 0.7079
s: 0.7079
Epoch 394/400
472/472 2s 5ms/step - accuracy: 0.8524 - loss: 0.3197 - val_accuracy: 0.7335 - val_loss: 0.7116
s: 0.7116
Epoch 395/400
472/472 3s 5ms/step - accuracy: 0.8547 - loss: 0.3180 - val_accuracy: 0.7337 - val_loss: 0.7084
s: 0.7084
Epoch 396/400
472/472 2s 5ms/step - accuracy: 0.8530 - loss: 0.3225 - val_accuracy: 0.7348 - val_loss: 0.7159
s: 0.7159
Epoch 397/400
472/472 2s 5ms/step - accuracy: 0.8543 - loss: 0.3171 - val_accuracy: 0.7401 - val_loss: 0.6975
s: 0.6975
Epoch 398/400
472/472 2s 5ms/step - accuracy: 0.8544 - loss: 0.3171 - val_accuracy: 0.7368 - val_loss: 0.7076
s: 0.7076
Epoch 399/400
472/472 2s 5ms/step - accuracy: 0.8562 - loss: 0.3151 - val_accuracy: 0.7401 - val_loss: 0.7138
s: 0.7138
Epoch 400/400
472/472 2s 5ms/step - accuracy: 0.8542 - loss: 0.3170 - val_accuracy: 0.7325 - val_loss: 0.7223
s: 0.7223
```

```
In [245...]: ann3_smote.evaluate(X_train_smote, y_train_smote)
```

```
3770/3770 4s 944us/step - accuracy: 0.8592 - loss: 0.3692
```

```
Out[245...]: [0.26518791913986206, 0.9030399322509766]
```

```
In [246...]: ann3_smote.summary()
```

```
Model: "sequential_5"
```

Layer (type)	Output Shape	Param #
dense_29 (Dense)	(None, 512)	23,040
dropout_24 (Dropout)	(None, 512)	0
dense_30 (Dense)	(None, 256)	131,328
dropout_25 (Dropout)	(None, 256)	0
dense_31 (Dense)	(None, 128)	32,896
dropout_26 (Dropout)	(None, 128)	0
dense_32 (Dense)	(None, 128)	16,512
dropout_27 (Dropout)	(None, 128)	0
dense_33 (Dense)	(None, 128)	16,512
dropout_28 (Dropout)	(None, 128)	0
dense_34 (Dense)	(None, 64)	8,256
dropout_29 (Dropout)	(None, 64)	0
dense_35 (Dense)	(None, 3)	195

Total params: 686,219 (2.62 MB)

Trainable params: 228,739 (893.51 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 457,480 (1.75 MB)

```
In [247...]: eval_metric(ann3_smote, X_train_smote, y_train_smote, X_val, y_val)
```

```
3770/3770 ━━━━━━━━ 4s 1ms/step
591/591 ━━━━━━ 1s 958us/step
```

Test Set:

```
[[4784 512 212]
 [1954 6358 1741]
 [ 91 362 2889]]
```

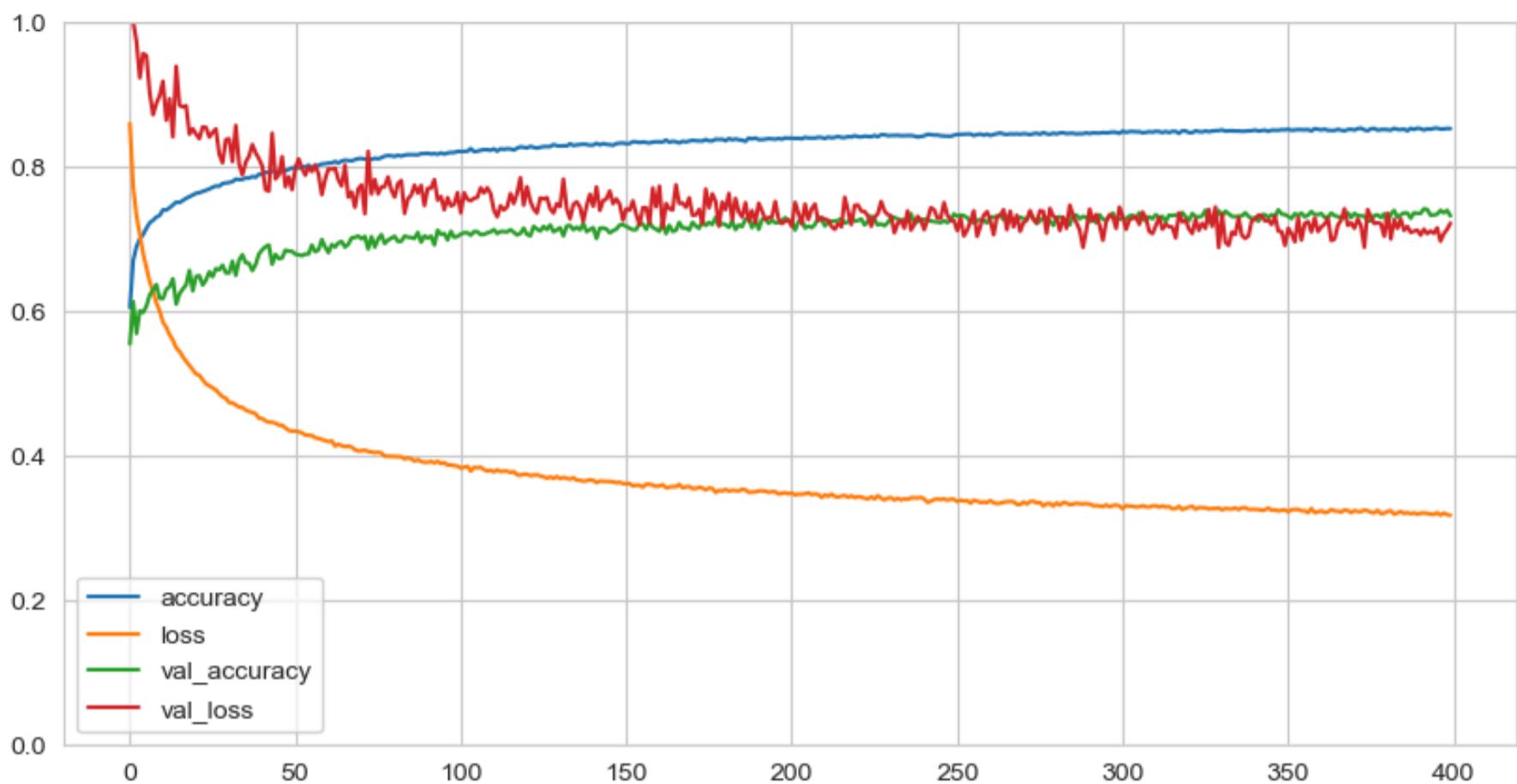
	precision	recall	f1-score	support
0	0.70	0.87	0.78	5508
1	0.88	0.63	0.74	10053
2	0.60	0.86	0.71	3342
accuracy			0.74	18903
macro avg	0.73	0.79	0.74	18903
weighted avg	0.78	0.74	0.74	18903

Train Set:

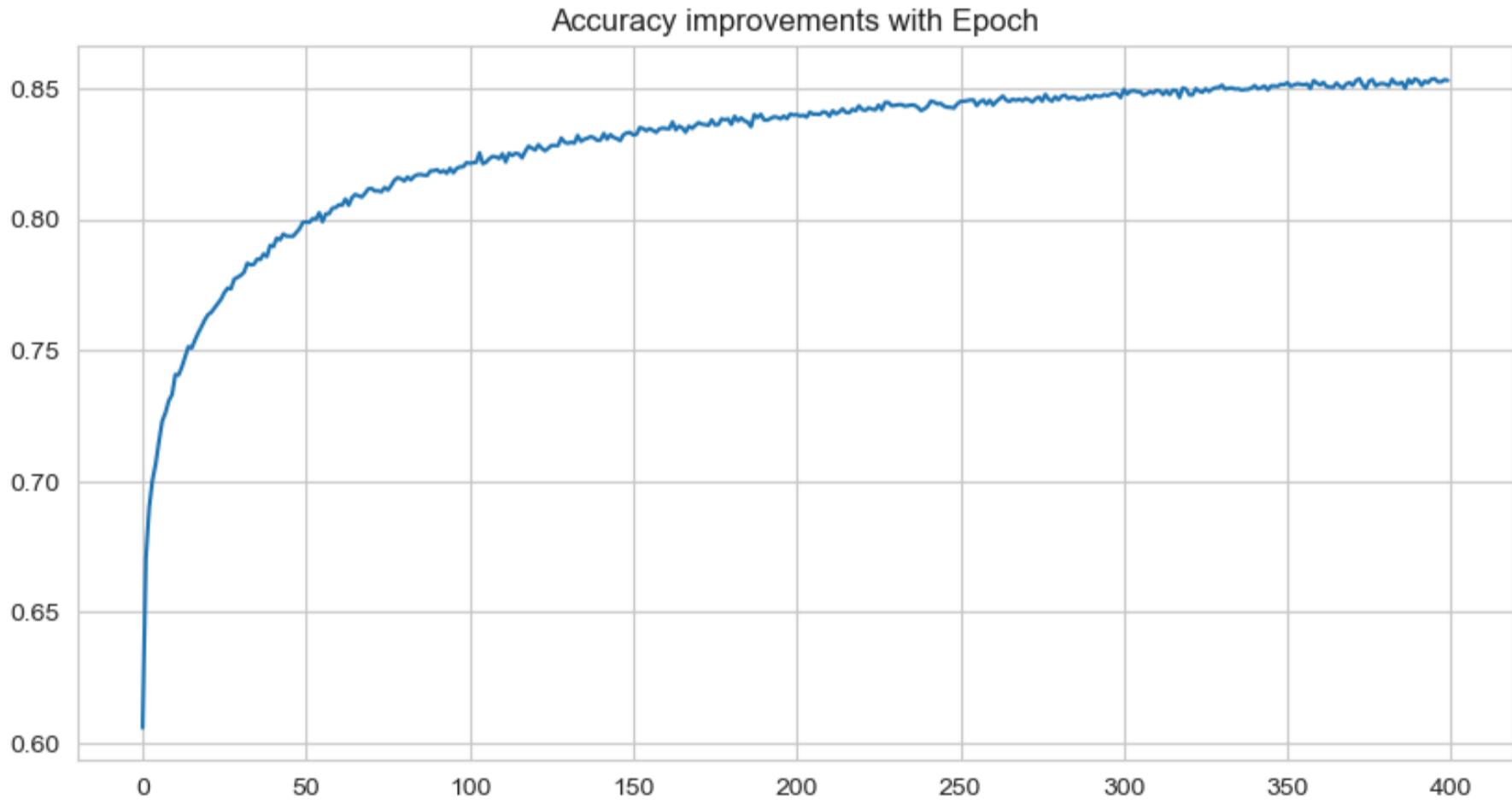
```
[[39629 384 196]
 [ 5751 29113 5345]
 [ 1 19 40189]]
```

	precision	recall	f1-score	support
0	0.87	0.99	0.93	40209
1	0.99	0.72	0.84	40209
2	0.88	1.00	0.94	40209
accuracy			0.90	120627
macro avg	0.91	0.90	0.90	120627
weighted avg	0.91	0.90	0.90	120627

```
In [248...]: pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



```
In [249]: pd.DataFrame(history.history)[["accuracy"]].plot(figsize=(10, 5))
plt.title("Accuracy improvements with Epoch")
plt.show()
```



```
In [ ]: # Save the Model

from tensorflow.keras.models import load_model

# Save the model to 'model.h5' file
ann3_smote.save('ann3_smote_model.h5')

# To Load the model Later for further use
loaded_ann3_smote = load_model('ann3_smote_model.h5')
```

ANN-3 Model with SMOTE Summary:

- **(Dense) layers: 7 / Neurons: 512-256-128-128-128-64 / Dropout: 30-30-25-25-20-20% / Learning Rate: 0.001 / Batch Size: 256 / Epochs: 400 / Early Stop (val_accuracy): 50**
- **Accuracy: 0.9030 / Val_Accuracy: 0.7410 / Loss: 0.2652 / Val_Loss: 0.7223 / Train Recall (Class 2): 1.00 / Test Recall (Class 2): 0.86**

- **Improvements:**

- High training recall for Class 2 shows effective learning of the minority class.
- Consistent training performance with 90.30% accuracy.

- **No Improvement:**

- Validation accuracy remains low at 74.10%, indicating issues with generalization.

- **Worsening Aspects:**

- The significant gap between training and validation performance and high validation loss suggest overfitting.

Future Adjustments:

- Consider increasing batch size and extending epochs to enhance model stability and allow more comprehensive learning.
- Adjust dropout rates and explore alternative architectures to improve generalization and reduce overfitting.

ANN-4 Model +Batchsize +Epoch (%90)

In [250...]

```
# 1) Model Architecture:
# Fully connected (Dense)---> 7 Layers
ann4 = Sequential([
    Dense(512, input_dim=X_train.shape[1], activation='relu'),
    Dropout(0.3),
    Dense(256, activation='relu'),
    Dropout(0.3),
    Dense(256, activation='relu'),
    Dropout(0.25),
    Dense(128, activation='relu'),
    Dropout(0.25),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(3, activation='softmax')
])

# 2) Compiling the Model:
ann4.compile(optimizer=Adam(learning_rate=0.001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 3) Early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=60,
                               mode="auto",
                               restore_best_weights=True)

# 4) Train the model
history = ann4.fit(
    x=X_train,
    y=y_train,
    validation_data=(X_val, y_val),
    batch_size=512, # Increased batch_size from 256 to --> 512
    epochs=600,      # Increased Epoch from 400 to --> 600
    verbose=1,
    callbacks=[early_stopping])
```

Epoch 1/600
148/148 7s 10ms/step - accuracy: 0.5369 - loss: 0.9152 - val_accuracy: 0.6243 - val_loss: 0.7746
Epoch 2/600
148/148 1s 8ms/step - accuracy: 0.6187 - loss: 0.7777 - val_accuracy: 0.6522 - val_loss: 0.7342
Epoch 3/600
148/148 1s 8ms/step - accuracy: 0.6484 - loss: 0.7450 - val_accuracy: 0.6643 - val_loss: 0.7239
Epoch 4/600
148/148 1s 9ms/step - accuracy: 0.6578 - loss: 0.7306 - val_accuracy: 0.6674 - val_loss: 0.7189
Epoch 5/600
148/148 1s 8ms/step - accuracy: 0.6646 - loss: 0.7259 - val_accuracy: 0.6753 - val_loss: 0.7120
Epoch 6/600
148/148 1s 9ms/step - accuracy: 0.6705 - loss: 0.7171 - val_accuracy: 0.6810 - val_loss: 0.7066
Epoch 7/600
148/148 1s 9ms/step - accuracy: 0.6761 - loss: 0.7062 - val_accuracy: 0.6787 - val_loss: 0.7051
Epoch 8/600
148/148 1s 8ms/step - accuracy: 0.6791 - loss: 0.7043 - val_accuracy: 0.6840 - val_loss: 0.7025
Epoch 9/600
148/148 1s 9ms/step - accuracy: 0.6809 - loss: 0.6969 - val_accuracy: 0.6870 - val_loss: 0.6973
Epoch 10/600
148/148 1s 8ms/step - accuracy: 0.6837 - loss: 0.6940 - val_accuracy: 0.6903 - val_loss: 0.6909
Epoch 11/600
148/148 1s 7ms/step - accuracy: 0.6896 - loss: 0.6860 - val_accuracy: 0.6861 - val_loss: 0.6917
Epoch 12/600
148/148 1s 8ms/step - accuracy: 0.6892 - loss: 0.6823 - val_accuracy: 0.6893 - val_loss: 0.6908
Epoch 13/600
148/148 1s 7ms/step - accuracy: 0.6939 - loss: 0.6768 - val_accuracy: 0.6943 - val_loss: 0.6851
Epoch 14/600
148/148 1s 8ms/step - accuracy: 0.6964 - loss: 0.6719 - val_accuracy: 0.6938 - val_loss: 0.6792
Epoch 15/600
148/148 1s 9ms/step - accuracy: 0.6960 - loss: 0.6700 - val_accuracy: 0.6949 - val_loss: 0.6795
Epoch 16/600
148/148 1s 8ms/step - accuracy: 0.7025 - loss: 0.6608 - val_accuracy: 0.6945 - val_loss: 0.6769
Epoch 17/600
148/148 1s 9ms/step - accuracy: 0.7062 - loss: 0.6526 - val_accuracy: 0.6989 - val_loss: 0.6719
Epoch 18/600
148/148 1s 8ms/step - accuracy: 0.7071 - loss: 0.6502 - val_accuracy: 0.6998 - val_loss: 0.6674
Epoch 19/600
148/148 1s 8ms/step - accuracy: 0.7067 - loss: 0.6501 - val_accuracy: 0.7019 - val_loss: 0.6669
Epoch 20/600
148/148 1s 8ms/step - accuracy: 0.7073 - loss: 0.6455 - val_accuracy: 0.7055 - val_loss: 0.6651
Epoch 21/600
148/148 1s 9ms/step - accuracy: 0.7104 - loss: 0.6409 - val_accuracy: 0.7055 - val_loss: 0.6616
Epoch 22/600
148/148 1s 8ms/step - accuracy: 0.7199 - loss: 0.6323 - val_accuracy: 0.7071 - val_loss: 0.6593
Epoch 23/600
148/148 1s 7ms/step - accuracy: 0.7178 - loss: 0.6325 - val_accuracy: 0.7067 - val_loss: 0.6540
Epoch 24/600
148/148 1s 9ms/step - accuracy: 0.7164 - loss: 0.6246 - val_accuracy: 0.7102 - val_loss: 0.6511
Epoch 25/600
148/148 1s 9ms/step - accuracy: 0.7227 - loss: 0.6173 - val_accuracy: 0.7116 - val_loss: 0.6518
Epoch 26/600

148/148 1s 10ms/step - accuracy: 0.7249 - loss: 0.6138 - val_accuracy: 0.7102 - val_loss: 0.6524
Epoch 27/600
148/148 1s 8ms/step - accuracy: 0.7243 - loss: 0.6118 - val_accuracy: 0.7112 - val_loss: 0.6489
Epoch 28/600
148/148 1s 8ms/step - accuracy: 0.7260 - loss: 0.6094 - val_accuracy: 0.7117 - val_loss: 0.6437
Epoch 29/600
148/148 1s 8ms/step - accuracy: 0.7246 - loss: 0.6092 - val_accuracy: 0.7152 - val_loss: 0.6455
Epoch 30/600
148/148 1s 8ms/step - accuracy: 0.7268 - loss: 0.6039 - val_accuracy: 0.7130 - val_loss: 0.6417
Epoch 31/600
148/148 1s 9ms/step - accuracy: 0.7323 - loss: 0.6004 - val_accuracy: 0.7169 - val_loss: 0.6379
Epoch 32/600
148/148 1s 8ms/step - accuracy: 0.7321 - loss: 0.5994 - val_accuracy: 0.7162 - val_loss: 0.6386
Epoch 33/600
148/148 1s 9ms/step - accuracy: 0.7357 - loss: 0.5905 - val_accuracy: 0.7188 - val_loss: 0.6346
Epoch 34/600
148/148 1s 8ms/step - accuracy: 0.7331 - loss: 0.5965 - val_accuracy: 0.7202 - val_loss: 0.6322
Epoch 35/600
148/148 1s 8ms/step - accuracy: 0.7387 - loss: 0.5848 - val_accuracy: 0.7184 - val_loss: 0.6324
Epoch 36/600
148/148 1s 8ms/step - accuracy: 0.7379 - loss: 0.5865 - val_accuracy: 0.7230 - val_loss: 0.6296
Epoch 37/600
148/148 1s 8ms/step - accuracy: 0.7400 - loss: 0.5813 - val_accuracy: 0.7230 - val_loss: 0.6271
Epoch 38/600
148/148 1s 8ms/step - accuracy: 0.7429 - loss: 0.5753 - val_accuracy: 0.7242 - val_loss: 0.6282
Epoch 39/600
148/148 1s 8ms/step - accuracy: 0.7421 - loss: 0.5787 - val_accuracy: 0.7254 - val_loss: 0.6243
Epoch 40/600
148/148 1s 8ms/step - accuracy: 0.7430 - loss: 0.5740 - val_accuracy: 0.7269 - val_loss: 0.6279
Epoch 41/600
148/148 1s 8ms/step - accuracy: 0.7444 - loss: 0.5730 - val_accuracy: 0.7237 - val_loss: 0.6233
Epoch 42/600
148/148 1s 10ms/step - accuracy: 0.7468 - loss: 0.5715 - val_accuracy: 0.7249 - val_loss: 0.6236
Epoch 43/600
148/148 1s 9ms/step - accuracy: 0.7462 - loss: 0.5682 - val_accuracy: 0.7267 - val_loss: 0.6234
Epoch 44/600
148/148 1s 8ms/step - accuracy: 0.7476 - loss: 0.5645 - val_accuracy: 0.7267 - val_loss: 0.6286
Epoch 45/600
148/148 1s 9ms/step - accuracy: 0.7519 - loss: 0.5593 - val_accuracy: 0.7291 - val_loss: 0.6195
Epoch 46/600
148/148 1s 8ms/step - accuracy: 0.7481 - loss: 0.5671 - val_accuracy: 0.7315 - val_loss: 0.6153
Epoch 47/600
148/148 1s 7ms/step - accuracy: 0.7504 - loss: 0.5595 - val_accuracy: 0.7306 - val_loss: 0.6188
Epoch 48/600
148/148 1s 8ms/step - accuracy: 0.7525 - loss: 0.5580 - val_accuracy: 0.7285 - val_loss: 0.6178
Epoch 49/600
148/148 1s 8ms/step - accuracy: 0.7525 - loss: 0.5553 - val_accuracy: 0.7346 - val_loss: 0.6123
Epoch 50/600
148/148 1s 7ms/step - accuracy: 0.7548 - loss: 0.5524 - val_accuracy: 0.7310 - val_loss: 0.6123
Epoch 51/600
148/148 1s 8ms/step - accuracy: 0.7581 - loss: 0.5480 - val_accuracy: 0.7321 - val_loss:

s: 0.6108
Epoch 52/600
148/148 1s 9ms/step - accuracy: 0.7573 - loss: 0.5510 - val_accuracy: 0.7326 - val_loss: 0.5930
s: 0.6139
Epoch 53/600
148/148 1s 8ms/step - accuracy: 0.7599 - loss: 0.5445 - val_accuracy: 0.7381 - val_loss: 0.5930
s: 0.6101
Epoch 54/600
148/148 1s 8ms/step - accuracy: 0.7580 - loss: 0.5454 - val_accuracy: 0.7377 - val_loss: 0.5930
s: 0.6101
Epoch 55/600
148/148 1s 8ms/step - accuracy: 0.7565 - loss: 0.5442 - val_accuracy: 0.7394 - val_loss: 0.5930
s: 0.6052
Epoch 56/600
148/148 1s 8ms/step - accuracy: 0.7595 - loss: 0.5412 - val_accuracy: 0.7395 - val_loss: 0.5930
s: 0.6042
Epoch 57/600
148/148 1s 8ms/step - accuracy: 0.7639 - loss: 0.5377 - val_accuracy: 0.7350 - val_loss: 0.5930
s: 0.6078
Epoch 58/600
148/148 1s 8ms/step - accuracy: 0.7635 - loss: 0.5355 - val_accuracy: 0.7390 - val_loss: 0.5930
s: 0.6022
Epoch 59/600
148/148 1s 9ms/step - accuracy: 0.7644 - loss: 0.5353 - val_accuracy: 0.7385 - val_loss: 0.5930
s: 0.6100
Epoch 60/600
148/148 1s 8ms/step - accuracy: 0.7642 - loss: 0.5349 - val_accuracy: 0.7362 - val_loss: 0.5930
s: 0.6107
Epoch 61/600
148/148 1s 7ms/step - accuracy: 0.7646 - loss: 0.5362 - val_accuracy: 0.7412 - val_loss: 0.5930
s: 0.6090
Epoch 62/600
148/148 1s 8ms/step - accuracy: 0.7645 - loss: 0.5300 - val_accuracy: 0.7381 - val_loss: 0.5930
s: 0.6091
Epoch 63/600
148/148 1s 7ms/step - accuracy: 0.7647 - loss: 0.5306 - val_accuracy: 0.7450 - val_loss: 0.5930
s: 0.5983
Epoch 64/600
148/148 1s 8ms/step - accuracy: 0.7648 - loss: 0.5328 - val_accuracy: 0.7408 - val_loss: 0.5930
s: 0.6031
Epoch 65/600
148/148 1s 8ms/step - accuracy: 0.7674 - loss: 0.5273 - val_accuracy: 0.7418 - val_loss: 0.5930
s: 0.6033
Epoch 66/600
148/148 1s 7ms/step - accuracy: 0.7652 - loss: 0.5306 - val_accuracy: 0.7416 - val_loss: 0.5930
s: 0.6052
Epoch 67/600
148/148 1s 8ms/step - accuracy: 0.7703 - loss: 0.5215 - val_accuracy: 0.7397 - val_loss: 0.5930
s: 0.6094
Epoch 68/600
148/148 1s 8ms/step - accuracy: 0.7684 - loss: 0.5243 - val_accuracy: 0.7465 - val_loss: 0.5930
s: 0.5933
Epoch 69/600
148/148 1s 8ms/step - accuracy: 0.7706 - loss: 0.5201 - val_accuracy: 0.7432 - val_loss: 0.5930
s: 0.6021
Epoch 70/600
148/148 1s 9ms/step - accuracy: 0.7693 - loss: 0.5227 - val_accuracy: 0.7436 - val_loss: 0.5930
s: 0.5980
Epoch 71/600
148/148 1s 8ms/step - accuracy: 0.7706 - loss: 0.5180 - val_accuracy: 0.7458 - val_loss: 0.5930
s: 0.5961
Epoch 72/600
148/148 1s 7ms/step - accuracy: 0.7721 - loss: 0.5152 - val_accuracy: 0.7486 - val_loss: 0.5930
s: 0.6002
Epoch 73/600
148/148 1s 9ms/step - accuracy: 0.7701 - loss: 0.5217 - val_accuracy: 0.7439 - val_loss: 0.5930
s: 0.5965
Epoch 74/600
148/148 1s 7ms/step - accuracy: 0.7706 - loss: 0.5190 - val_accuracy: 0.7479 - val_loss: 0.5930
s: 0.5930
Epoch 75/600
148/148 1s 7ms/step - accuracy: 0.7785 - loss: 0.5073 - val_accuracy: 0.7473 - val_loss: 0.5930
s: 0.5948
Epoch 76/600
148/148 1s 7ms/step - accuracy: 0.7741 - loss: 0.5160 - val_accuracy: 0.7473 - val_loss: 0.5930
s: 0.5976

Epoch 77/600
148/148 1s 7ms/step - accuracy: 0.7756 - loss: 0.5086 - val_accuracy: 0.7461 - val_loss: 0.5963
Epoch 78/600
148/148 1s 7ms/step - accuracy: 0.7787 - loss: 0.5093 - val_accuracy: 0.7461 - val_loss: 0.5942
Epoch 79/600
148/148 1s 8ms/step - accuracy: 0.7728 - loss: 0.5132 - val_accuracy: 0.7461 - val_loss: 0.5895
Epoch 80/600
148/148 1s 8ms/step - accuracy: 0.7763 - loss: 0.5106 - val_accuracy: 0.7469 - val_loss: 0.5960
Epoch 81/600
148/148 1s 8ms/step - accuracy: 0.7793 - loss: 0.5061 - val_accuracy: 0.7485 - val_loss: 0.5896
Epoch 82/600
148/148 1s 8ms/step - accuracy: 0.7792 - loss: 0.5032 - val_accuracy: 0.7465 - val_loss: 0.5950
Epoch 83/600
148/148 1s 8ms/step - accuracy: 0.7791 - loss: 0.5022 - val_accuracy: 0.7475 - val_loss: 0.5936
Epoch 84/600
148/148 1s 8ms/step - accuracy: 0.7809 - loss: 0.5045 - val_accuracy: 0.7478 - val_loss: 0.5949
Epoch 85/600
148/148 1s 7ms/step - accuracy: 0.7797 - loss: 0.5015 - val_accuracy: 0.7484 - val_loss: 0.5885
Epoch 86/600
148/148 1s 8ms/step - accuracy: 0.7805 - loss: 0.5031 - val_accuracy: 0.7498 - val_loss: 0.5928
Epoch 87/600
148/148 1s 8ms/step - accuracy: 0.7826 - loss: 0.4937 - val_accuracy: 0.7491 - val_loss: 0.5916
Epoch 88/600
148/148 1s 8ms/step - accuracy: 0.7806 - loss: 0.4993 - val_accuracy: 0.7473 - val_loss: 0.5934
Epoch 89/600
148/148 1s 7ms/step - accuracy: 0.7822 - loss: 0.4978 - val_accuracy: 0.7503 - val_loss: 0.5944
Epoch 90/600
148/148 1s 7ms/step - accuracy: 0.7802 - loss: 0.4989 - val_accuracy: 0.7535 - val_loss: 0.5960
Epoch 91/600
148/148 1s 8ms/step - accuracy: 0.7820 - loss: 0.4998 - val_accuracy: 0.7514 - val_loss: 0.5895
Epoch 92/600
148/148 1s 7ms/step - accuracy: 0.7853 - loss: 0.4930 - val_accuracy: 0.7510 - val_loss: 0.5918
Epoch 93/600
148/148 1s 8ms/step - accuracy: 0.7852 - loss: 0.4937 - val_accuracy: 0.7519 - val_loss: 0.5901
Epoch 94/600
148/148 1s 7ms/step - accuracy: 0.7831 - loss: 0.4936 - val_accuracy: 0.7531 - val_loss: 0.5938
Epoch 95/600
148/148 1s 7ms/step - accuracy: 0.7863 - loss: 0.4912 - val_accuracy: 0.7523 - val_loss: 0.5881
Epoch 96/600
148/148 1s 8ms/step - accuracy: 0.7887 - loss: 0.4867 - val_accuracy: 0.7558 - val_loss: 0.5840
Epoch 97/600
148/148 1s 8ms/step - accuracy: 0.7852 - loss: 0.4907 - val_accuracy: 0.7504 - val_loss: 0.5875
Epoch 98/600
148/148 1s 7ms/step - accuracy: 0.7866 - loss: 0.4881 - val_accuracy: 0.7546 - val_loss: 0.5908
Epoch 99/600
148/148 1s 7ms/step - accuracy: 0.7848 - loss: 0.4914 - val_accuracy: 0.7508 - val_loss: 0.5859
Epoch 100/600
148/148 1s 8ms/step - accuracy: 0.7844 - loss: 0.4869 - val_accuracy: 0.7553 - val_loss: 0.5870
Epoch 101/600
148/148 1s 8ms/step - accuracy: 0.7869 - loss: 0.4878 - val_accuracy: 0.7555 - val_loss: 0.5886
Epoch 102/600

148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7878 - loss: 0.4832 - val_accuracy: 0.7524 - val_loss: 0.5949
Epoch 103/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7887 - loss: 0.4853 - val_accuracy: 0.7525 - val_loss: 0.5883
Epoch 104/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7891 - loss: 0.4835 - val_accuracy: 0.7540 - val_loss: 0.5874
Epoch 105/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.7875 - loss: 0.4866 - val_accuracy: 0.7572 - val_loss: 0.5848
Epoch 106/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.7897 - loss: 0.4824 - val_accuracy: 0.7572 - val_loss: 0.5859
Epoch 107/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.7909 - loss: 0.4793 - val_accuracy: 0.7526 - val_loss: 0.5916
Epoch 108/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.7908 - loss: 0.4810 - val_accuracy: 0.7548 - val_loss: 0.5831
Epoch 109/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.7903 - loss: 0.4811 - val_accuracy: 0.7550 - val_loss: 0.5949
Epoch 110/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.7909 - loss: 0.4819 - val_accuracy: 0.7559 - val_loss: 0.5830
Epoch 111/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.7914 - loss: 0.4794 - val_accuracy: 0.7499 - val_loss: 0.5882
Epoch 112/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.7928 - loss: 0.4765 - val_accuracy: 0.7553 - val_loss: 0.5865
Epoch 113/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.7933 - loss: 0.4758 - val_accuracy: 0.7552 - val_loss: 0.5840
Epoch 114/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7931 - loss: 0.4762 - val_accuracy: 0.7562 - val_loss: 0.5818
Epoch 115/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.7898 - loss: 0.4791 - val_accuracy: 0.7555 - val_loss: 0.5842
Epoch 116/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.7918 - loss: 0.4747 - val_accuracy: 0.7577 - val_loss: 0.5792
Epoch 117/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.7909 - loss: 0.4758 - val_accuracy: 0.7560 - val_loss: 0.5788
Epoch 118/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7945 - loss: 0.4739 - val_accuracy: 0.7553 - val_loss: 0.5809
Epoch 119/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7920 - loss: 0.4751 - val_accuracy: 0.7572 - val_loss: 0.5808
Epoch 120/600
148/148 ━━━━━━ 1s 10ms/step - accuracy: 0.7955 - loss: 0.4742 - val_accuracy: 0.7550 - val_loss: 0.5822
Epoch 121/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.7922 - loss: 0.4738 - val_accuracy: 0.7606 - val_loss: 0.5776
Epoch 122/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.7962 - loss: 0.4730 - val_accuracy: 0.7551 - val_loss: 0.5895
Epoch 123/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.7953 - loss: 0.4688 - val_accuracy: 0.7604 - val_loss: 0.5812
Epoch 124/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7930 - loss: 0.4759 - val_accuracy: 0.7599 - val_loss: 0.5820
Epoch 125/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7950 - loss: 0.4693 - val_accuracy: 0.7561 - val_loss: 0.5848
Epoch 126/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7960 - loss: 0.4711 - val_accuracy: 0.7544 - val_loss: 0.5885
Epoch 127/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.7961 - loss: 0.4698 - val_accuracy: 0.7573 - val_loss:

s: 0.5814
Epoch 128/600
148/148 1s 7ms/step - accuracy: 0.7962 - loss: 0.4704 - val_accuracy: 0.7526 - val_loss: 0.5849
Epoch 129/600
148/148 1s 8ms/step - accuracy: 0.7935 - loss: 0.4732 - val_accuracy: 0.7558 - val_loss: 0.5850
Epoch 130/600
148/148 1s 7ms/step - accuracy: 0.8009 - loss: 0.4630 - val_accuracy: 0.7573 - val_loss: 0.5841
Epoch 131/600
148/148 1s 7ms/step - accuracy: 0.7991 - loss: 0.4698 - val_accuracy: 0.7607 - val_loss: 0.5849
Epoch 132/600
148/148 1s 7ms/step - accuracy: 0.7981 - loss: 0.4674 - val_accuracy: 0.7599 - val_loss: 0.5814
Epoch 133/600
148/148 1s 7ms/step - accuracy: 0.7998 - loss: 0.4652 - val_accuracy: 0.7598 - val_loss: 0.5921
Epoch 134/600
148/148 1s 7ms/step - accuracy: 0.8014 - loss: 0.4616 - val_accuracy: 0.7586 - val_loss: 0.5842
Epoch 135/600
148/148 1s 8ms/step - accuracy: 0.8020 - loss: 0.4643 - val_accuracy: 0.7575 - val_loss: 0.5847
s: 0.5777
Epoch 136/600
148/148 1s 8ms/step - accuracy: 0.7976 - loss: 0.4639 - val_accuracy: 0.7583 - val_loss: 0.5829
s: 0.5829
Epoch 138/600
148/148 1s 8ms/step - accuracy: 0.8012 - loss: 0.4597 - val_accuracy: 0.7582 - val_loss: 0.5792
s: 0.5792
Epoch 139/600
148/148 1s 9ms/step - accuracy: 0.7988 - loss: 0.4596 - val_accuracy: 0.7600 - val_loss: 0.5839
s: 0.5839
Epoch 140/600
148/148 1s 8ms/step - accuracy: 0.8021 - loss: 0.4595 - val_accuracy: 0.7618 - val_loss: 0.5834
s: 0.5834
Epoch 141/600
148/148 1s 8ms/step - accuracy: 0.7990 - loss: 0.4619 - val_accuracy: 0.7602 - val_loss: 0.5753
s: 0.5753
Epoch 142/600
148/148 1s 7ms/step - accuracy: 0.8011 - loss: 0.4606 - val_accuracy: 0.7580 - val_loss: 0.5840
s: 0.5840
Epoch 143/600
148/148 1s 8ms/step - accuracy: 0.8033 - loss: 0.4591 - val_accuracy: 0.7601 - val_loss: 0.5799
s: 0.5799
Epoch 144/600
148/148 1s 7ms/step - accuracy: 0.8021 - loss: 0.4596 - val_accuracy: 0.7589 - val_loss: 0.5791
s: 0.5791
Epoch 145/600
148/148 1s 7ms/step - accuracy: 0.7994 - loss: 0.4614 - val_accuracy: 0.7579 - val_loss: 0.5835
s: 0.5835
Epoch 146/600
148/148 1s 8ms/step - accuracy: 0.8010 - loss: 0.4600 - val_accuracy: 0.7583 - val_loss: 0.5838
s: 0.5838
Epoch 147/600
148/148 1s 8ms/step - accuracy: 0.8005 - loss: 0.4592 - val_accuracy: 0.7561 - val_loss: 0.5750
s: 0.5750
Epoch 148/600
148/148 1s 9ms/step - accuracy: 0.8020 - loss: 0.4621 - val_accuracy: 0.7604 - val_loss: 0.5855
s: 0.5855
Epoch 149/600
148/148 1s 10ms/step - accuracy: 0.8041 - loss: 0.4564 - val_accuracy: 0.7616 - val_loss: 0.5824
s: 0.5824
Epoch 150/600
148/148 3s 10ms/step - accuracy: 0.8021 - loss: 0.4583 - val_accuracy: 0.7618 - val_loss: 0.5826
s: 0.5826
Epoch 151/600
148/148 1s 10ms/step - accuracy: 0.8015 - loss: 0.4546 - val_accuracy: 0.7601 - val_loss: 0.5822
s: 0.5822
Epoch 152/600
148/148 1s 8ms/step - accuracy: 0.7987 - loss: 0.4575 - val_accuracy: 0.7616 - val_loss: 0.5818
s: 0.5818

Epoch 153/600
148/148 1s 9ms/step - accuracy: 0.8045 - loss: 0.4540 - val_accuracy: 0.7607 - val_loss: 0.5849
Epoch 154/600
148/148 1s 8ms/step - accuracy: 0.8073 - loss: 0.4495 - val_accuracy: 0.7611 - val_loss: 0.5837
Epoch 155/600
148/148 1s 8ms/step - accuracy: 0.8033 - loss: 0.4559 - val_accuracy: 0.7583 - val_loss: 0.5865
Epoch 156/600
148/148 1s 7ms/step - accuracy: 0.8052 - loss: 0.4532 - val_accuracy: 0.7608 - val_loss: 0.5817
Epoch 157/600
148/148 1s 8ms/step - accuracy: 0.8064 - loss: 0.4511 - val_accuracy: 0.7606 - val_loss: 0.5799
Epoch 158/600
148/148 1s 8ms/step - accuracy: 0.8037 - loss: 0.4557 - val_accuracy: 0.7635 - val_loss: 0.5818
Epoch 159/600
148/148 1s 7ms/step - accuracy: 0.8047 - loss: 0.4527 - val_accuracy: 0.7578 - val_loss: 0.5804
Epoch 160/600
148/148 2s 11ms/step - accuracy: 0.8064 - loss: 0.4543 - val_accuracy: 0.7608 - val_loss: 0.5847
Epoch 161/600
148/148 1s 9ms/step - accuracy: 0.8047 - loss: 0.4497 - val_accuracy: 0.7622 - val_loss: 0.5787
Epoch 162/600
148/148 1s 8ms/step - accuracy: 0.8049 - loss: 0.4529 - val_accuracy: 0.7620 - val_loss: 0.5816
Epoch 163/600
148/148 1s 9ms/step - accuracy: 0.8064 - loss: 0.4470 - val_accuracy: 0.7608 - val_loss: 0.5748
Epoch 164/600
148/148 1s 10ms/step - accuracy: 0.8060 - loss: 0.4513 - val_accuracy: 0.7637 - val_loss: 0.5816
Epoch 165/600
148/148 1s 8ms/step - accuracy: 0.8035 - loss: 0.4532 - val_accuracy: 0.7634 - val_loss: 0.5773
Epoch 166/600
148/148 1s 8ms/step - accuracy: 0.8051 - loss: 0.4501 - val_accuracy: 0.7613 - val_loss: 0.5869
Epoch 167/600
148/148 1s 7ms/step - accuracy: 0.8051 - loss: 0.4517 - val_accuracy: 0.7625 - val_loss: 0.5817
Epoch 168/600
148/148 1s 8ms/step - accuracy: 0.8090 - loss: 0.4439 - val_accuracy: 0.7621 - val_loss: 0.5809
Epoch 169/600
148/148 1s 8ms/step - accuracy: 0.8063 - loss: 0.4483 - val_accuracy: 0.7625 - val_loss: 0.5787
Epoch 170/600
148/148 1s 8ms/step - accuracy: 0.8068 - loss: 0.4492 - val_accuracy: 0.7591 - val_loss: 0.5738
Epoch 171/600
148/148 1s 9ms/step - accuracy: 0.8115 - loss: 0.4413 - val_accuracy: 0.7616 - val_loss: 0.5798
Epoch 172/600
148/148 1s 7ms/step - accuracy: 0.8065 - loss: 0.4524 - val_accuracy: 0.7629 - val_loss: 0.5801
Epoch 173/600
148/148 1s 8ms/step - accuracy: 0.8064 - loss: 0.4464 - val_accuracy: 0.7613 - val_loss: 0.5752
Epoch 174/600
148/148 1s 8ms/step - accuracy: 0.8111 - loss: 0.4453 - val_accuracy: 0.7610 - val_loss: 0.5710
Epoch 175/600
148/148 1s 9ms/step - accuracy: 0.8081 - loss: 0.4470 - val_accuracy: 0.7623 - val_loss: 0.5796
Epoch 176/600
148/148 1s 8ms/step - accuracy: 0.8041 - loss: 0.4498 - val_accuracy: 0.7626 - val_loss: 0.5803
Epoch 177/600
148/148 1s 8ms/step - accuracy: 0.8088 - loss: 0.4428 - val_accuracy: 0.7640 - val_loss: 0.5716
Epoch 178/600

148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8039 - loss: 0.4543 - val_accuracy: 0.7662 - val_loss: 0.5768
Epoch 179/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.8103 - loss: 0.4426 - val_accuracy: 0.7631 - val_loss: 0.5715
Epoch 180/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8093 - loss: 0.4429 - val_accuracy: 0.7658 - val_loss: 0.5784
Epoch 181/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.8094 - loss: 0.4472 - val_accuracy: 0.7651 - val_loss: 0.5758
Epoch 182/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8103 - loss: 0.4424 - val_accuracy: 0.7638 - val_loss: 0.5804
Epoch 183/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8064 - loss: 0.4463 - val_accuracy: 0.7634 - val_loss: 0.5859
Epoch 184/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8129 - loss: 0.4400 - val_accuracy: 0.7635 - val_loss: 0.5769
Epoch 185/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8130 - loss: 0.4381 - val_accuracy: 0.7668 - val_loss: 0.5757
Epoch 186/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8105 - loss: 0.4391 - val_accuracy: 0.7669 - val_loss: 0.5767
Epoch 187/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8079 - loss: 0.4432 - val_accuracy: 0.7636 - val_loss: 0.5822
Epoch 188/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8108 - loss: 0.4387 - val_accuracy: 0.7614 - val_loss: 0.5786
Epoch 189/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8110 - loss: 0.4352 - val_accuracy: 0.7633 - val_loss: 0.5763
Epoch 190/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8092 - loss: 0.4434 - val_accuracy: 0.7635 - val_loss: 0.5691
Epoch 191/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8109 - loss: 0.4391 - val_accuracy: 0.7626 - val_loss: 0.5781
Epoch 192/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8127 - loss: 0.4349 - val_accuracy: 0.7663 - val_loss: 0.5718
Epoch 193/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8142 - loss: 0.4378 - val_accuracy: 0.7638 - val_loss: 0.5742
Epoch 194/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8136 - loss: 0.4342 - val_accuracy: 0.7670 - val_loss: 0.5741
Epoch 195/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8137 - loss: 0.4375 - val_accuracy: 0.7643 - val_loss: 0.5702
Epoch 196/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8093 - loss: 0.4409 - val_accuracy: 0.7661 - val_loss: 0.5791
Epoch 197/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8131 - loss: 0.4382 - val_accuracy: 0.7657 - val_loss: 0.5780
Epoch 198/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8139 - loss: 0.4359 - val_accuracy: 0.7649 - val_loss: 0.5793
Epoch 199/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8089 - loss: 0.4402 - val_accuracy: 0.7635 - val_loss: 0.5773
Epoch 200/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8077 - loss: 0.4440 - val_accuracy: 0.7659 - val_loss: 0.5761
Epoch 201/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8154 - loss: 0.4341 - val_accuracy: 0.7650 - val_loss: 0.5760
Epoch 202/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8126 - loss: 0.4337 - val_accuracy: 0.7671 - val_loss: 0.5725
Epoch 203/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8139 - loss: 0.4341 - val_accuracy: 0.7652 - val_loss:

s: 0.5775
Epoch 204/600
148/148 1s 9ms/step - accuracy: 0.8144 - loss: 0.4349 - val_accuracy: 0.7619 - val_loss: 0.5741
Epoch 205/600
148/148 1s 7ms/step - accuracy: 0.8121 - loss: 0.4353 - val_accuracy: 0.7634 - val_loss: 0.5756
Epoch 206/600
148/148 1s 7ms/step - accuracy: 0.8132 - loss: 0.4356 - val_accuracy: 0.7661 - val_loss: 0.5731
Epoch 207/600
148/148 1s 7ms/step - accuracy: 0.8131 - loss: 0.4327 - val_accuracy: 0.7628 - val_loss: 0.5773
Epoch 208/600
148/148 1s 7ms/step - accuracy: 0.8148 - loss: 0.4335 - val_accuracy: 0.7634 - val_loss: 0.5770
Epoch 209/600
148/148 1s 7ms/step - accuracy: 0.8150 - loss: 0.4347 - val_accuracy: 0.7662 - val_loss: 0.5727
Epoch 210/600
148/148 1s 7ms/step - accuracy: 0.8134 - loss: 0.4340 - val_accuracy: 0.7646 - val_loss: 0.5799
Epoch 211/600
148/148 1s 7ms/step - accuracy: 0.8126 - loss: 0.4327 - val_accuracy: 0.7644 - val_loss: 0.5804
Epoch 212/600
148/148 1s 7ms/step - accuracy: 0.8140 - loss: 0.4326 - val_accuracy: 0.7657 - val_loss: 0.5747
Epoch 213/600
148/148 1s 7ms/step - accuracy: 0.8099 - loss: 0.4353 - val_accuracy: 0.7637 - val_loss: 0.5808
Epoch 214/600
148/148 1s 7ms/step - accuracy: 0.8139 - loss: 0.4350 - val_accuracy: 0.7665 - val_loss: 0.5817
Epoch 215/600
148/148 1s 7ms/step - accuracy: 0.8135 - loss: 0.4321 - val_accuracy: 0.7654 - val_loss: 0.5727
Epoch 216/600
148/148 1s 7ms/step - accuracy: 0.8141 - loss: 0.4325 - val_accuracy: 0.7693 - val_loss: 0.5708
Epoch 217/600
148/148 1s 7ms/step - accuracy: 0.8139 - loss: 0.4340 - val_accuracy: 0.7669 - val_loss: 0.5664
Epoch 218/600
148/148 1s 8ms/step - accuracy: 0.8138 - loss: 0.4314 - val_accuracy: 0.7672 - val_loss: 0.5700
Epoch 219/600
148/148 1s 8ms/step - accuracy: 0.8186 - loss: 0.4286 - val_accuracy: 0.7636 - val_loss: 0.5810
Epoch 220/600
148/148 1s 7ms/step - accuracy: 0.8161 - loss: 0.4294 - val_accuracy: 0.7661 - val_loss: 0.5791
Epoch 221/600
148/148 1s 7ms/step - accuracy: 0.8141 - loss: 0.4341 - val_accuracy: 0.7672 - val_loss: 0.5779
Epoch 222/600
148/148 1s 8ms/step - accuracy: 0.8167 - loss: 0.4275 - val_accuracy: 0.7641 - val_loss: 0.5772
Epoch 223/600
148/148 1s 7ms/step - accuracy: 0.8158 - loss: 0.4270 - val_accuracy: 0.7656 - val_loss: 0.5782
Epoch 224/600
148/148 1s 7ms/step - accuracy: 0.8175 - loss: 0.4286 - val_accuracy: 0.7637 - val_loss: 0.5775
Epoch 225/600
148/148 1s 9ms/step - accuracy: 0.8122 - loss: 0.4342 - val_accuracy: 0.7616 - val_loss: 0.5752
Epoch 226/600
148/148 1s 7ms/step - accuracy: 0.8150 - loss: 0.4310 - val_accuracy: 0.7666 - val_loss: 0.5778
Epoch 227/600
148/148 1s 7ms/step - accuracy: 0.8162 - loss: 0.4304 - val_accuracy: 0.7652 - val_loss: 0.5803
Epoch 228/600
148/148 1s 7ms/step - accuracy: 0.8177 - loss: 0.4273 - val_accuracy: 0.7636 - val_loss: 0.5779

Epoch 229/600
148/148 1s 7ms/step - accuracy: 0.8181 - loss: 0.4258 - val_accuracy: 0.7672 - val_loss: 0.5755
Epoch 230/600
148/148 1s 7ms/step - accuracy: 0.8167 - loss: 0.4297 - val_accuracy: 0.7664 - val_loss: 0.5712
Epoch 231/600
148/148 1s 8ms/step - accuracy: 0.8160 - loss: 0.4253 - val_accuracy: 0.7641 - val_loss: 0.5782
s: 0.5733
Epoch 232/600
148/148 1s 7ms/step - accuracy: 0.8151 - loss: 0.4286 - val_accuracy: 0.7656 - val_loss: 0.5787
s: 0.5809
Epoch 233/600
148/148 1s 7ms/step - accuracy: 0.8174 - loss: 0.4300 - val_accuracy: 0.7695 - val_loss: 0.5693
s: 0.5702
Epoch 234/600
148/148 1s 8ms/step - accuracy: 0.8149 - loss: 0.4292 - val_accuracy: 0.7661 - val_loss: 0.5762
s: 0.5649
Epoch 235/600
148/148 1s 7ms/step - accuracy: 0.8197 - loss: 0.4221 - val_accuracy: 0.7661 - val_loss: 0.5767
s: 0.5767
Epoch 236/600
148/148 1s 7ms/step - accuracy: 0.8166 - loss: 0.4287 - val_accuracy: 0.7670 - val_loss: 0.5700
s: 0.5700
Epoch 237/600
148/148 1s 8ms/step - accuracy: 0.8159 - loss: 0.4319 - val_accuracy: 0.7665 - val_loss: 0.5749
s: 0.5749
Epoch 238/600
148/148 1s 7ms/step - accuracy: 0.8188 - loss: 0.4233 - val_accuracy: 0.7656 - val_loss: 0.5754
s: 0.5754
Epoch 239/600
148/148 1s 7ms/step - accuracy: 0.8141 - loss: 0.4296 - val_accuracy: 0.7668 - val_loss: 0.5772
s: 0.5772
Epoch 240/600
148/148 1s 7ms/step - accuracy: 0.8180 - loss: 0.4245 - val_accuracy: 0.7676 - val_loss: 0.5796
s: 0.5796
Epoch 241/600
148/148 1s 7ms/step - accuracy: 0.8215 - loss: 0.4222 - val_accuracy: 0.7665 - val_loss: 0.5715
s: 0.5715
Epoch 242/600
148/148 1s 7ms/step - accuracy: 0.8201 - loss: 0.4227 - val_accuracy: 0.7672 - val_loss: 0.5735
s: 0.5735
Epoch 243/600
148/148 1s 7ms/step - accuracy: 0.8192 - loss: 0.4254 - val_accuracy: 0.7674 - val_loss: 0.5750
s: 0.5750
Epoch 244/600
148/148 1s 7ms/step - accuracy: 0.8183 - loss: 0.4233 - val_accuracy: 0.7665 - val_loss: 0.5745
s: 0.5745
Epoch 245/600
148/148 1s 8ms/step - accuracy: 0.8212 - loss: 0.4186 - val_accuracy: 0.7641 - val_loss: 0.5819
s: 0.5819
Epoch 246/600
148/148 1s 8ms/step - accuracy: 0.8221 - loss: 0.4212 - val_accuracy: 0.7655 - val_loss: 0.5806
s: 0.5806
Epoch 247/600
148/148 1s 8ms/step - accuracy: 0.8210 - loss: 0.4236 - val_accuracy: 0.7665 - val_loss: 0.5744
s: 0.5744
Epoch 248/600
148/148 1s 7ms/step - accuracy: 0.8192 - loss: 0.4197 - val_accuracy: 0.7662 - val_loss: 0.5741
s: 0.5741
Epoch 249/600
148/148 1s 7ms/step - accuracy: 0.8184 - loss: 0.4241 - val_accuracy: 0.7669 - val_loss: 0.5741
s: 0.5741
Epoch 250/600
148/148 1s 7ms/step - accuracy: 0.8176 - loss: 0.4246 - val_accuracy: 0.7665 - val_loss: 0.5741
s: 0.5741
Epoch 251/600
148/148 1s 7ms/step - accuracy: 0.8210 - loss: 0.4194 - val_accuracy: 0.7672 - val_loss: 0.5741
s: 0.5741
Epoch 252/600
148/148 1s 7ms/step - accuracy: 0.8210 - loss: 0.4194 - val_accuracy: 0.7672 - val_loss: 0.5741
s: 0.5741
Epoch 253/600
148/148 1s 7ms/step - accuracy: 0.8210 - loss: 0.4194 - val_accuracy: 0.7672 - val_loss: 0.5741
s: 0.5741
Epoch 254/600

148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8214 - loss: 0.4209 - val_accuracy: 0.7687 - val_loss: 0.5800
Epoch 255/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8218 - loss: 0.4196 - val_accuracy: 0.7668 - val_loss: 0.5733
Epoch 256/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8189 - loss: 0.4227 - val_accuracy: 0.7670 - val_loss: 0.5812
Epoch 257/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8239 - loss: 0.4142 - val_accuracy: 0.7669 - val_loss: 0.5740
Epoch 258/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8189 - loss: 0.4227 - val_accuracy: 0.7645 - val_loss: 0.5824
Epoch 259/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8219 - loss: 0.4183 - val_accuracy: 0.7702 - val_loss: 0.5725
Epoch 260/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8221 - loss: 0.4187 - val_accuracy: 0.7668 - val_loss: 0.5787
Epoch 261/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8227 - loss: 0.4149 - val_accuracy: 0.7658 - val_loss: 0.5702
Epoch 262/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8173 - loss: 0.4224 - val_accuracy: 0.7692 - val_loss: 0.5791
Epoch 263/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8211 - loss: 0.4189 - val_accuracy: 0.7669 - val_loss: 0.5725
Epoch 264/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8181 - loss: 0.4201 - val_accuracy: 0.7686 - val_loss: 0.5794
Epoch 265/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8213 - loss: 0.4212 - val_accuracy: 0.7686 - val_loss: 0.5767
Epoch 266/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8208 - loss: 0.4201 - val_accuracy: 0.7632 - val_loss: 0.5834
Epoch 267/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8206 - loss: 0.4179 - val_accuracy: 0.7686 - val_loss: 0.5723
Epoch 268/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8215 - loss: 0.4191 - val_accuracy: 0.7647 - val_loss: 0.5754
Epoch 269/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8218 - loss: 0.4225 - val_accuracy: 0.7660 - val_loss: 0.5711
Epoch 270/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8206 - loss: 0.4178 - val_accuracy: 0.7669 - val_loss: 0.5771
Epoch 271/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8197 - loss: 0.4165 - val_accuracy: 0.7693 - val_loss: 0.5771
Epoch 272/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8237 - loss: 0.4145 - val_accuracy: 0.7676 - val_loss: 0.5733
Epoch 273/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8205 - loss: 0.4193 - val_accuracy: 0.7642 - val_loss: 0.5819
Epoch 274/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8248 - loss: 0.4139 - val_accuracy: 0.7683 - val_loss: 0.5715
Epoch 275/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8206 - loss: 0.4152 - val_accuracy: 0.7667 - val_loss: 0.5810
Epoch 276/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8223 - loss: 0.4151 - val_accuracy: 0.7676 - val_loss: 0.5778
Epoch 277/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8216 - loss: 0.4193 - val_accuracy: 0.7668 - val_loss: 0.5788
Epoch 278/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8241 - loss: 0.4155 - val_accuracy: 0.7653 - val_loss: 0.5808
Epoch 279/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8240 - loss: 0.4170 - val_accuracy: 0.7665 - val_loss:

s: 0.5730
Epoch 280/600
148/148 1s 7ms/step - accuracy: 0.8224 - loss: 0.4144 - val_accuracy: 0.7676 - val_loss: 0.5751
Epoch 281/600
148/148 1s 7ms/step - accuracy: 0.8253 - loss: 0.4161 - val_accuracy: 0.7658 - val_loss: 0.5826
Epoch 282/600
148/148 1s 7ms/step - accuracy: 0.8237 - loss: 0.4159 - val_accuracy: 0.7670 - val_loss: 0.5811
s: 0.5755
Epoch 283/600
148/148 1s 8ms/step - accuracy: 0.8250 - loss: 0.4140 - val_accuracy: 0.7670 - val_loss: 0.5752
s: 0.5758
Epoch 284/600
148/148 1s 7ms/step - accuracy: 0.8221 - loss: 0.4159 - val_accuracy: 0.7697 - val_loss: 0.5758
s: 0.5735
Epoch 285/600
148/148 1s 7ms/step - accuracy: 0.8262 - loss: 0.4109 - val_accuracy: 0.7687 - val_loss: 0.5773
s: 0.5773
Epoch 286/600
148/148 1s 7ms/step - accuracy: 0.8236 - loss: 0.4136 - val_accuracy: 0.7673 - val_loss: 0.5720
s: 0.5790
Epoch 287/600
148/148 1s 8ms/step - accuracy: 0.8246 - loss: 0.4115 - val_accuracy: 0.7702 - val_loss: 0.5790
s: 0.5821
Epoch 288/600
148/148 1s 7ms/step - accuracy: 0.8251 - loss: 0.4119 - val_accuracy: 0.7707 - val_loss: 0.5822
s: 0.5773
Epoch 289/600
148/148 1s 7ms/step - accuracy: 0.8273 - loss: 0.4077 - val_accuracy: 0.7686 - val_loss: 0.5794
s: 0.5773
Epoch 290/600
148/148 1s 8ms/step - accuracy: 0.8236 - loss: 0.4105 - val_accuracy: 0.7684 - val_loss: 0.5773
s: 0.5795
Epoch 291/600
148/148 1s 7ms/step - accuracy: 0.8237 - loss: 0.4144 - val_accuracy: 0.7697 - val_loss: 0.5727
s: 0.5688
Epoch 292/600
148/148 1s 7ms/step - accuracy: 0.8236 - loss: 0.4129 - val_accuracy: 0.7696 - val_loss: 0.5721
s: 0.5780
Epoch 293/600
148/148 1s 7ms/step - accuracy: 0.8280 - loss: 0.4083 - val_accuracy: 0.7698 - val_loss: 0.5780
s: 0.5838
Epoch 294/600
148/148 1s 7ms/step - accuracy: 0.8215 - loss: 0.4156 - val_accuracy: 0.7667 - val_loss: 0.5765
s: 0.5702
Epoch 295/600
148/148 1s 7ms/step - accuracy: 0.8256 - loss: 0.4091 - val_accuracy: 0.7689 - val_loss: 0.5753
s: 0.5702
Epoch 296/600
148/148 1s 7ms/step - accuracy: 0.8247 - loss: 0.4134 - val_accuracy: 0.7671 - val_loss: 0.5839
s: 0.5839

Epoch 305/600
148/148 1s 7ms/step - accuracy: 0.8230 - loss: 0.4131 - val_accuracy: 0.7682 - val_loss: 0.5860
Epoch 306/600
148/148 1s 8ms/step - accuracy: 0.8272 - loss: 0.4081 - val_accuracy: 0.7686 - val_loss: 0.5777
Epoch 307/600
148/148 1s 8ms/step - accuracy: 0.8233 - loss: 0.4135 - val_accuracy: 0.7717 - val_loss: 0.5828
Epoch 308/600
148/148 1s 7ms/step - accuracy: 0.8252 - loss: 0.4069 - val_accuracy: 0.7714 - val_loss: 0.5748
Epoch 309/600
148/148 1s 7ms/step - accuracy: 0.8246 - loss: 0.4087 - val_accuracy: 0.7697 - val_loss: 0.5795
Epoch 310/600
148/148 1s 7ms/step - accuracy: 0.8247 - loss: 0.4103 - val_accuracy: 0.7690 - val_loss: 0.5747
Epoch 311/600
148/148 1s 7ms/step - accuracy: 0.8230 - loss: 0.4131 - val_accuracy: 0.7697 - val_loss: 0.5772
Epoch 312/600
148/148 1s 7ms/step - accuracy: 0.8246 - loss: 0.4100 - val_accuracy: 0.7727 - val_loss: 0.5780
Epoch 313/600
148/148 1s 7ms/step - accuracy: 0.8226 - loss: 0.4153 - val_accuracy: 0.7701 - val_loss: 0.5850
Epoch 314/600
148/148 1s 7ms/step - accuracy: 0.8276 - loss: 0.4050 - val_accuracy: 0.7712 - val_loss: 0.5784
Epoch 315/600
148/148 1s 7ms/step - accuracy: 0.8270 - loss: 0.4045 - val_accuracy: 0.7700 - val_loss: 0.5693
Epoch 316/600
148/148 1s 7ms/step - accuracy: 0.8287 - loss: 0.4050 - val_accuracy: 0.7681 - val_loss: 0.5733
Epoch 317/600
148/148 1s 7ms/step - accuracy: 0.8266 - loss: 0.4095 - val_accuracy: 0.7684 - val_loss: 0.5806
Epoch 318/600
148/148 1s 7ms/step - accuracy: 0.8282 - loss: 0.4091 - val_accuracy: 0.7682 - val_loss: 0.5734
Epoch 319/600
148/148 1s 7ms/step - accuracy: 0.8240 - loss: 0.4154 - val_accuracy: 0.7676 - val_loss: 0.5803
Epoch 320/600
148/148 1s 7ms/step - accuracy: 0.8293 - loss: 0.4009 - val_accuracy: 0.7684 - val_loss: 0.5658
Epoch 321/600
148/148 1s 7ms/step - accuracy: 0.8278 - loss: 0.4084 - val_accuracy: 0.7679 - val_loss: 0.5717
Epoch 322/600
148/148 1s 7ms/step - accuracy: 0.8272 - loss: 0.4118 - val_accuracy: 0.7695 - val_loss: 0.5730
Epoch 323/600
148/148 1s 7ms/step - accuracy: 0.8241 - loss: 0.4114 - val_accuracy: 0.7694 - val_loss: 0.5981
Epoch 324/600
148/148 1s 7ms/step - accuracy: 0.8309 - loss: 0.3983 - val_accuracy: 0.7711 - val_loss: 0.5827
Epoch 325/600
148/148 1s 7ms/step - accuracy: 0.8252 - loss: 0.4077 - val_accuracy: 0.7690 - val_loss: 0.5746
Epoch 326/600
148/148 1s 7ms/step - accuracy: 0.8264 - loss: 0.4097 - val_accuracy: 0.7696 - val_loss: 0.5740
Epoch 327/600
148/148 1s 8ms/step - accuracy: 0.8285 - loss: 0.4044 - val_accuracy: 0.7699 - val_loss: 0.5781
Epoch 328/600
148/148 1s 7ms/step - accuracy: 0.8253 - loss: 0.4085 - val_accuracy: 0.7676 - val_loss: 0.5789
Epoch 329/600
148/148 1s 7ms/step - accuracy: 0.8280 - loss: 0.4063 - val_accuracy: 0.7702 - val_loss: 0.5702
Epoch 330/600

148/148 1s 7ms/step - accuracy: 0.8287 - loss: 0.4054 - val_accuracy: 0.7674 - val_loss: 0.5720
Epoch 331/600
148/148 1s 7ms/step - accuracy: 0.8276 - loss: 0.4060 - val_accuracy: 0.7666 - val_loss: 0.5704
Epoch 332/600
148/148 1s 7ms/step - accuracy: 0.8281 - loss: 0.4026 - val_accuracy: 0.7676 - val_loss: 0.5726
Epoch 333/600
148/148 1s 8ms/step - accuracy: 0.8275 - loss: 0.4056 - val_accuracy: 0.7700 - val_loss: 0.5682
Epoch 334/600
148/148 1s 7ms/step - accuracy: 0.8258 - loss: 0.4095 - val_accuracy: 0.7688 - val_loss: 0.5814
Epoch 335/600
148/148 1s 7ms/step - accuracy: 0.8233 - loss: 0.4087 - val_accuracy: 0.7705 - val_loss: 0.5799
Epoch 336/600
148/148 1s 7ms/step - accuracy: 0.8291 - loss: 0.4029 - val_accuracy: 0.7669 - val_loss: 0.5827
Epoch 337/600
148/148 1s 8ms/step - accuracy: 0.8243 - loss: 0.4083 - val_accuracy: 0.7687 - val_loss: 0.5803
Epoch 338/600
148/148 1s 8ms/step - accuracy: 0.8298 - loss: 0.4022 - val_accuracy: 0.7671 - val_loss: 0.5823
Epoch 339/600
148/148 1s 7ms/step - accuracy: 0.8265 - loss: 0.4098 - val_accuracy: 0.7727 - val_loss: 0.5749
Epoch 340/600
148/148 1s 7ms/step - accuracy: 0.8269 - loss: 0.4079 - val_accuracy: 0.7668 - val_loss: 0.5908
Epoch 341/600
148/148 1s 7ms/step - accuracy: 0.8300 - loss: 0.4025 - val_accuracy: 0.7686 - val_loss: 0.5690
Epoch 342/600
148/148 1s 7ms/step - accuracy: 0.8283 - loss: 0.4036 - val_accuracy: 0.7697 - val_loss: 0.5721
Epoch 343/600
148/148 1s 7ms/step - accuracy: 0.8256 - loss: 0.4092 - val_accuracy: 0.7698 - val_loss: 0.5734
Epoch 344/600
148/148 1s 7ms/step - accuracy: 0.8298 - loss: 0.3976 - val_accuracy: 0.7703 - val_loss: 0.5801
Epoch 345/600
148/148 1s 7ms/step - accuracy: 0.8310 - loss: 0.4002 - val_accuracy: 0.7695 - val_loss: 0.5764
Epoch 346/600
148/148 1s 7ms/step - accuracy: 0.8314 - loss: 0.3988 - val_accuracy: 0.7715 - val_loss: 0.5841
Epoch 347/600
148/148 1s 8ms/step - accuracy: 0.8315 - loss: 0.3989 - val_accuracy: 0.7711 - val_loss: 0.5757
Epoch 348/600
148/148 1s 7ms/step - accuracy: 0.8283 - loss: 0.4082 - val_accuracy: 0.7669 - val_loss: 0.5768
Epoch 349/600
148/148 1s 7ms/step - accuracy: 0.8270 - loss: 0.4073 - val_accuracy: 0.7727 - val_loss: 0.5812
Epoch 350/600
148/148 1s 7ms/step - accuracy: 0.8312 - loss: 0.4007 - val_accuracy: 0.7708 - val_loss: 0.5761
Epoch 351/600
148/148 1s 8ms/step - accuracy: 0.8275 - loss: 0.4037 - val_accuracy: 0.7692 - val_loss: 0.5793
Epoch 352/600
148/148 1s 7ms/step - accuracy: 0.8286 - loss: 0.4066 - val_accuracy: 0.7687 - val_loss: 0.5776
Epoch 353/600
148/148 1s 8ms/step - accuracy: 0.8311 - loss: 0.3977 - val_accuracy: 0.7684 - val_loss: 0.5858
Epoch 354/600
148/148 1s 8ms/step - accuracy: 0.8290 - loss: 0.4027 - val_accuracy: 0.7686 - val_loss: 0.5767
Epoch 355/600
148/148 1s 7ms/step - accuracy: 0.8323 - loss: 0.3968 - val_accuracy: 0.7695 - val_loss:

s: 0.5833
Epoch 356/600
148/148 1s 7ms/step - accuracy: 0.8304 - loss: 0.3992 - val_accuracy: 0.7705 - val_loss: 0.5810
Epoch 357/600
148/148 1s 8ms/step - accuracy: 0.8269 - loss: 0.4061 - val_accuracy: 0.7715 - val_loss: 0.5882
Epoch 358/600
148/148 1s 8ms/step - accuracy: 0.8284 - loss: 0.4031 - val_accuracy: 0.7676 - val_loss: 0.5793
Epoch 359/600
148/148 1s 8ms/step - accuracy: 0.8352 - loss: 0.3969 - val_accuracy: 0.7689 - val_loss: 0.5766
Epoch 360/600
148/148 1s 9ms/step - accuracy: 0.8278 - loss: 0.4075 - val_accuracy: 0.7675 - val_loss: 0.5807
Epoch 361/600
148/148 1s 8ms/step - accuracy: 0.8310 - loss: 0.3988 - val_accuracy: 0.7691 - val_loss: 0.5770
Epoch 362/600
148/148 1s 8ms/step - accuracy: 0.8304 - loss: 0.3979 - val_accuracy: 0.7709 - val_loss: 0.5695
Epoch 363/600
148/148 1s 8ms/step - accuracy: 0.8292 - loss: 0.4008 - val_accuracy: 0.7726 - val_loss: 0.5713
Epoch 364/600
148/148 1s 8ms/step - accuracy: 0.8294 - loss: 0.4008 - val_accuracy: 0.7669 - val_loss: 0.5759
Epoch 365/600
148/148 1s 8ms/step - accuracy: 0.8291 - loss: 0.4020 - val_accuracy: 0.7684 - val_loss: 0.5797
Epoch 366/600
148/148 1s 8ms/step - accuracy: 0.8276 - loss: 0.4025 - val_accuracy: 0.7698 - val_loss: 0.5928
Epoch 367/600
148/148 1s 8ms/step - accuracy: 0.8293 - loss: 0.4056 - val_accuracy: 0.7671 - val_loss: 0.5796
Epoch 368/600
148/148 1s 7ms/step - accuracy: 0.8284 - loss: 0.4015 - val_accuracy: 0.7698 - val_loss: 0.5833
Epoch 369/600
148/148 1s 7ms/step - accuracy: 0.8303 - loss: 0.3997 - val_accuracy: 0.7695 - val_loss: 0.5836
Epoch 370/600
148/148 1s 8ms/step - accuracy: 0.8289 - loss: 0.4026 - val_accuracy: 0.7708 - val_loss: 0.5880
Epoch 371/600
148/148 1s 7ms/step - accuracy: 0.8317 - loss: 0.3979 - val_accuracy: 0.7693 - val_loss: 0.5801
Epoch 372/600
148/148 1s 7ms/step - accuracy: 0.8299 - loss: 0.3987 - val_accuracy: 0.7697 - val_loss: 0.5811
Epoch 373/600
148/148 1s 7ms/step - accuracy: 0.8319 - loss: 0.3984 - val_accuracy: 0.7656 - val_loss: 0.5837
Epoch 374/600
148/148 1s 7ms/step - accuracy: 0.8307 - loss: 0.4006 - val_accuracy: 0.7662 - val_loss: 0.5741
Epoch 375/600
148/148 1s 8ms/step - accuracy: 0.8320 - loss: 0.3979 - val_accuracy: 0.7709 - val_loss: 0.5682
Epoch 376/600
148/148 1s 7ms/step - accuracy: 0.8323 - loss: 0.3966 - val_accuracy: 0.7689 - val_loss: 0.5924
Epoch 377/600
148/148 1s 7ms/step - accuracy: 0.8308 - loss: 0.3994 - val_accuracy: 0.7690 - val_loss: 0.5887
Epoch 378/600
148/148 1s 7ms/step - accuracy: 0.8293 - loss: 0.3978 - val_accuracy: 0.7692 - val_loss: 0.5795
Epoch 379/600
148/148 1s 7ms/step - accuracy: 0.8328 - loss: 0.3962 - val_accuracy: 0.7714 - val_loss: 0.5872
Epoch 380/600
148/148 1s 7ms/step - accuracy: 0.8304 - loss: 0.3998 - val_accuracy: 0.7682 - val_loss: 0.5915

Epoch 381/600
148/148 1s 7ms/step - accuracy: 0.8309 - loss: 0.3963 - val_accuracy: 0.7657 - val_loss: 0.5904
Epoch 382/600
148/148 1s 7ms/step - accuracy: 0.8337 - loss: 0.3988 - val_accuracy: 0.7716 - val_loss: 0.5727
Epoch 383/600
148/148 1s 7ms/step - accuracy: 0.8328 - loss: 0.3949 - val_accuracy: 0.7688 - val_loss: 0.5907
Epoch 384/600
148/148 1s 7ms/step - accuracy: 0.8325 - loss: 0.3999 - val_accuracy: 0.7702 - val_loss: 0.5856
Epoch 385/600
148/148 1s 7ms/step - accuracy: 0.8297 - loss: 0.4015 - val_accuracy: 0.7669 - val_loss: 0.5761
Epoch 386/600
148/148 1s 7ms/step - accuracy: 0.8317 - loss: 0.3988 - val_accuracy: 0.7722 - val_loss: 0.5742
Epoch 387/600
148/148 1s 7ms/step - accuracy: 0.8313 - loss: 0.3977 - val_accuracy: 0.7688 - val_loss: 0.5788
Epoch 388/600
148/148 1s 8ms/step - accuracy: 0.8320 - loss: 0.3984 - val_accuracy: 0.7705 - val_loss: 0.5910
Epoch 389/600
148/148 1s 7ms/step - accuracy: 0.8309 - loss: 0.4006 - val_accuracy: 0.7691 - val_loss: 0.5858
Epoch 390/600
148/148 1s 8ms/step - accuracy: 0.8344 - loss: 0.3958 - val_accuracy: 0.7680 - val_loss: 0.5785
Epoch 391/600
148/148 1s 7ms/step - accuracy: 0.8330 - loss: 0.3951 - val_accuracy: 0.7706 - val_loss: 0.5825
Epoch 392/600
148/148 1s 8ms/step - accuracy: 0.8316 - loss: 0.4019 - val_accuracy: 0.7691 - val_loss: 0.5859
Epoch 393/600
148/148 1s 8ms/step - accuracy: 0.8325 - loss: 0.3958 - val_accuracy: 0.7684 - val_loss: 0.5808
Epoch 394/600
148/148 1s 7ms/step - accuracy: 0.8309 - loss: 0.3971 - val_accuracy: 0.7694 - val_loss: 0.5791
Epoch 395/600
148/148 1s 8ms/step - accuracy: 0.8327 - loss: 0.3952 - val_accuracy: 0.7689 - val_loss: 0.5802
Epoch 396/600
148/148 1s 8ms/step - accuracy: 0.8313 - loss: 0.3977 - val_accuracy: 0.7686 - val_loss: 0.5765
Epoch 397/600
148/148 1s 8ms/step - accuracy: 0.8339 - loss: 0.3953 - val_accuracy: 0.7672 - val_loss: 0.5871
Epoch 398/600
148/148 1s 7ms/step - accuracy: 0.8325 - loss: 0.3915 - val_accuracy: 0.7703 - val_loss: 0.5808
Epoch 399/600
148/148 1s 7ms/step - accuracy: 0.8299 - loss: 0.3996 - val_accuracy: 0.7704 - val_loss: 0.5924
Epoch 400/600
148/148 1s 7ms/step - accuracy: 0.8340 - loss: 0.3963 - val_accuracy: 0.7690 - val_loss: 0.5683
Epoch 401/600
148/148 1s 7ms/step - accuracy: 0.8327 - loss: 0.3966 - val_accuracy: 0.7698 - val_loss: 0.5837
Epoch 402/600
148/148 1s 7ms/step - accuracy: 0.8318 - loss: 0.3969 - val_accuracy: 0.7672 - val_loss: 0.5831
Epoch 403/600
148/148 1s 7ms/step - accuracy: 0.8327 - loss: 0.3949 - val_accuracy: 0.7696 - val_loss: 0.5858
Epoch 404/600
148/148 1s 9ms/step - accuracy: 0.8338 - loss: 0.3946 - val_accuracy: 0.7670 - val_loss: 0.5763
Epoch 405/600
148/148 1s 8ms/step - accuracy: 0.8330 - loss: 0.3911 - val_accuracy: 0.7681 - val_loss: 0.5825
Epoch 406/600

```
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.8370 - loss: 0.3900 - val_accuracy: 0.7700 - val_loss: 0.5729
Epoch 407/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.8326 - loss: 0.3927 - val_accuracy: 0.7709 - val_loss: 0.5793
Epoch 408/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.8335 - loss: 0.3945 - val_accuracy: 0.7692 - val_loss: 0.5853
Epoch 409/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.8339 - loss: 0.3928 - val_accuracy: 0.7691 - val_loss: 0.5821
```

```
In [251... ann4.evaluate(X_train, y_train)
```

```
2363/2363 ━━━━━━━━ 2s 1ms/step - accuracy: 0.9046 - loss: 0.2586
```

```
Out[251... [0.259221613407135, 0.9053961038589478]
```

```
In [252... ann4.summary()
```

Model: "sequential_6"

Layer (type)	Output Shape	Param #
dense_36 (Dense)	(None, 512)	23,040
dropout_30 (Dropout)	(None, 512)	0
dense_37 (Dense)	(None, 256)	131,328
dropout_31 (Dropout)	(None, 256)	0
dense_38 (Dense)	(None, 256)	65,792
dropout_32 (Dropout)	(None, 256)	0
dense_39 (Dense)	(None, 128)	32,896
dropout_33 (Dropout)	(None, 128)	0
dense_40 (Dense)	(None, 128)	16,512
dropout_34 (Dropout)	(None, 128)	0
dense_41 (Dense)	(None, 64)	8,256
dropout_35 (Dropout)	(None, 64)	0
dense_42 (Dense)	(None, 3)	195

Total params: 834,059 (3.18 MB)

Trainable params: 278,019 (1.06 MB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 556,040 (2.12 MB)

```
In [253... eval_metric(ann4, X_train, y_train, X_val, y_val)
```

2363/2363 ————— 3s 1ms/step
 591/591 ————— 1s 989us/step

Test Set:

```
[[4421 943 144]
 [1319 7628 1106]
 [ 63 721 2558]]
```

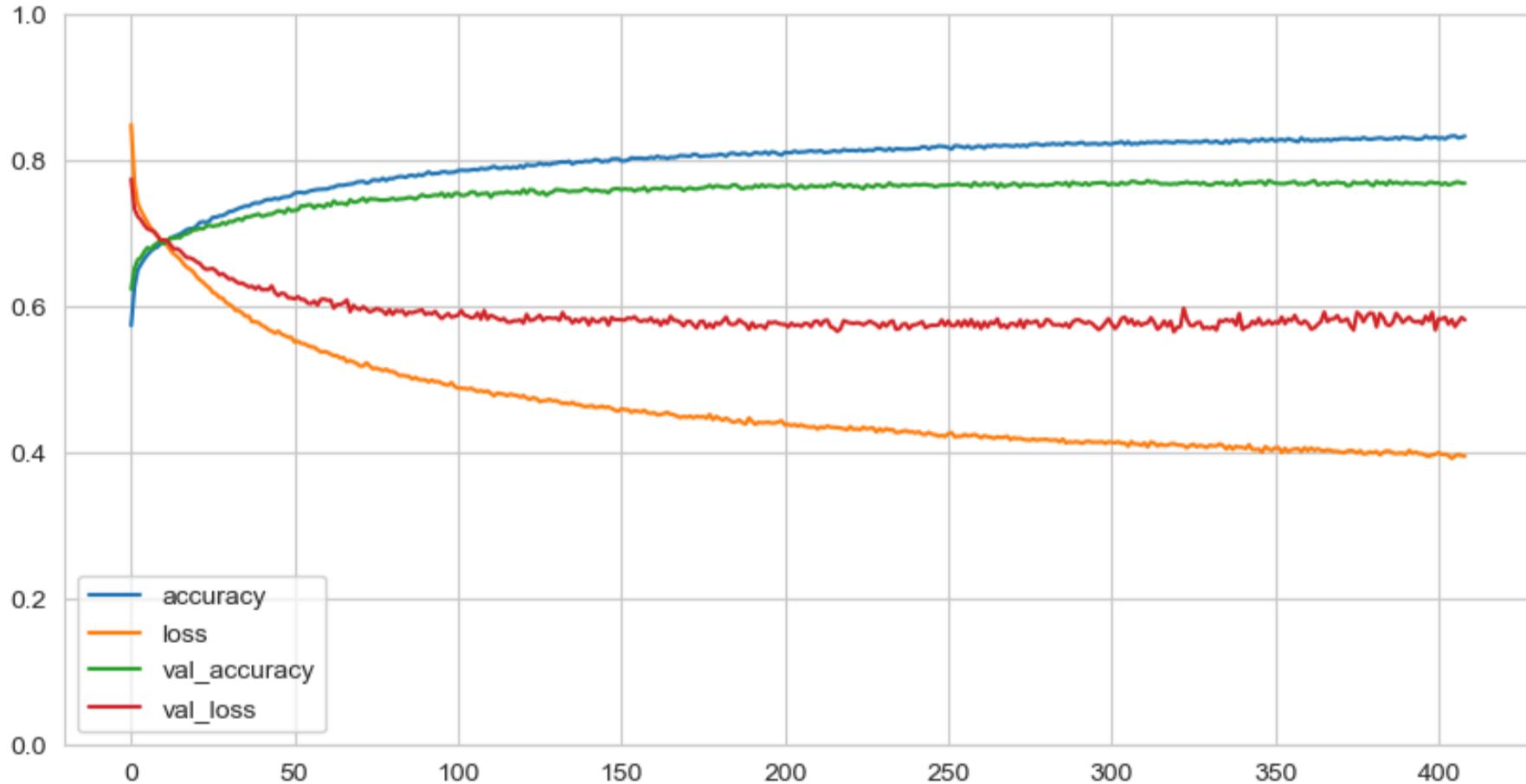
	precision	recall	f1-score	support
0	0.76	0.80	0.78	5508
1	0.82	0.76	0.79	10053
2	0.67	0.77	0.72	3342
accuracy			0.77	18903
macro avg	0.75	0.78	0.76	18903
weighted avg	0.78	0.77	0.77	18903

Train Set:

```
[[20875 1036 120]
 [ 2720 34939 2550]
 [ 23 704 12643]]
```

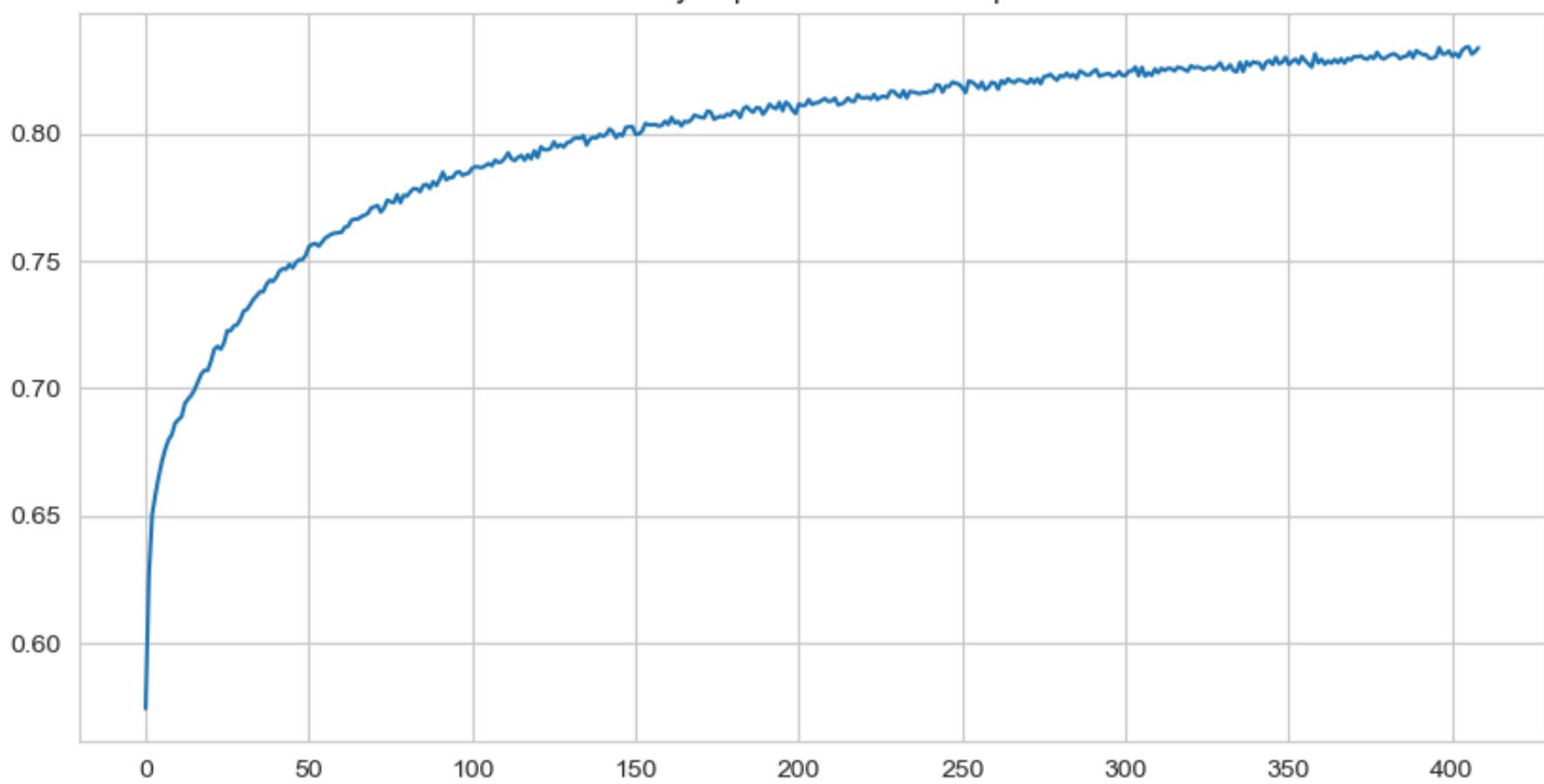
	precision	recall	f1-score	support
0	0.88	0.95	0.91	22031
1	0.95	0.87	0.91	40209
2	0.83	0.95	0.88	13370
accuracy			0.91	75610
macro avg	0.89	0.92	0.90	75610
weighted avg	0.91	0.91	0.91	75610

```
In [254...]: pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



```
In [255...]: pd.DataFrame(history.history)[“accuracy”].plot(figsize=(10, 5))
plt.title("Accuracy improvements with Epoch")
plt.show()
```

Accuracy improvements with Epoch



In [230...]

```
# Save the Model

from tensorflow.keras.models import load_model

# Save the model to 'model.h5' file
ann4.save('ann4_model.h5')

# To Load the model later for further use
loaded_ann4 = load_model('ann4_model.h5')
```

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)` . This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')` .
 WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.

ANN-4 Model Summary:

- **(Dense) layers: 7 / Neurons:** 512-256-256-128-128-64 / **Dropout:** 30-30-25-25-20-20% / **Learning Rate:** 0.001 / **Batch Size:** 512 / **Epochs:** 600 / **Early Stop (val_accuracy):** 60
- **Accuracy:** 0.9054 / **Val_Accuracy:** 0.7700 / **Loss:** 0.2586 / **Val_Loss:** 0.7300 / **Train Recall (Class 2):** 0.95 / **Test Recall (Class 2):** 0.77

Improvements: The model achieves high training accuracy (90.54%) and excellent recall for Class 2 (95% training, 77% testing), showing effective recognition of the minority class.

No Improvement: Validation accuracy remains relatively stable but does not show significant improvement, highlighting the challenge of generalizing the complex model to unseen data.

Got Worse: Validation loss indicates a potential overfitting issue, as it has increased to 0.7300 despite the improved performance metrics in training. This suggests that while the model is learning the training data well, it struggles to perform similarly on validation data.

ANN-4 Model with SMOTE (%89)

```
In [29]: print('Credi_Score Classes before Smote: ', y_train.value_counts())
print('\nCredi_Score Classes after Smote: ', y_train_smote.value_counts())
```

```
Credi_Score Classes before Smote: Credit_Score
1    40209
0    22031
2   13370
Name: count, dtype: int64
```

```
Credi_Score Classes after Smote: Credit_Score
1    40209
2    40209
0    40209
Name: count, dtype: int64
```

In [257]: # ANN-4 Model with SMOTE

```
# 1) Model Architecture:
# Fully connected (Dense)---> 7 Layers
ann4_smote = Sequential([
    Dense(512, input_dim=X_train.shape[1], activation='relu'),
    Dropout(0.3),
    Dense(256, activation='relu'),
    Dropout(0.3),
    Dense(256, activation='relu'),
    Dropout(0.25),
    Dense(128, activation='relu'),
    Dropout(0.25),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(3, activation='softmax')
])

# 2) Compiling the Model:
ann4_smote.compile(optimizer=Adam(learning_rate=0.001),
                    loss='sparse_categorical_crossentropy',
                    metrics=['accuracy'])

# 3) Early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=60,
                               mode="auto",
                               restore_best_weights=True)

# 4) Train the model
history = ann4_smote.fit(
    x=X_train,
    y=y_train,
    validation_data=(X_val, y_val),
    batch_size=512, # Increased batch_size from 256 to --> 512
    epochs=600,      # Increased Epoch from 400 to --> 600
    verbose=1,
    callbacks=[early_stopping])
```

Epoch 1/600
148/148 4s 10ms/step - accuracy: 0.5477 - loss: 0.9011 - val_accuracy: 0.6357 - val_loss: 0.7679
Epoch 2/600
148/148 1s 8ms/step - accuracy: 0.6229 - loss: 0.7777 - val_accuracy: 0.6507 - val_loss: 0.7362
Epoch 3/600
148/148 1s 9ms/step - accuracy: 0.6448 - loss: 0.7457 - val_accuracy: 0.6645 - val_loss: 0.7195
Epoch 4/600
148/148 1s 8ms/step - accuracy: 0.6585 - loss: 0.7310 - val_accuracy: 0.6742 - val_loss: 0.7128
Epoch 5/600
148/148 1s 9ms/step - accuracy: 0.6659 - loss: 0.7198 - val_accuracy: 0.6787 - val_loss: 0.7077
Epoch 6/600
148/148 1s 8ms/step - accuracy: 0.6701 - loss: 0.7142 - val_accuracy: 0.6813 - val_loss: 0.7020
Epoch 7/600
148/148 1s 8ms/step - accuracy: 0.6803 - loss: 0.7067 - val_accuracy: 0.6817 - val_loss: 0.7035
Epoch 8/600
148/148 1s 8ms/step - accuracy: 0.6805 - loss: 0.7029 - val_accuracy: 0.6826 - val_loss: 0.6992
Epoch 9/600
148/148 1s 8ms/step - accuracy: 0.6826 - loss: 0.6969 - val_accuracy: 0.6849 - val_loss: 0.6951
Epoch 10/600
148/148 1s 7ms/step - accuracy: 0.6872 - loss: 0.6911 - val_accuracy: 0.6881 - val_loss: 0.6920
Epoch 11/600
148/148 1s 7ms/step - accuracy: 0.6892 - loss: 0.6833 - val_accuracy: 0.6886 - val_loss: 0.6889
Epoch 12/600
148/148 1s 8ms/step - accuracy: 0.6913 - loss: 0.6804 - val_accuracy: 0.6909 - val_loss: 0.6850
Epoch 13/600
148/148 1s 8ms/step - accuracy: 0.6935 - loss: 0.6757 - val_accuracy: 0.6905 - val_loss: 0.6835
Epoch 14/600
148/148 1s 8ms/step - accuracy: 0.6966 - loss: 0.6746 - val_accuracy: 0.6952 - val_loss: 0.6793
Epoch 15/600
148/148 2s 10ms/step - accuracy: 0.6960 - loss: 0.6685 - val_accuracy: 0.6953 - val_loss: 0.6757
Epoch 16/600
148/148 1s 8ms/step - accuracy: 0.7040 - loss: 0.6590 - val_accuracy: 0.6952 - val_loss: 0.6770
Epoch 17/600
148/148 1s 9ms/step - accuracy: 0.7053 - loss: 0.6582 - val_accuracy: 0.6973 - val_loss: 0.6717
Epoch 18/600
148/148 2s 8ms/step - accuracy: 0.7068 - loss: 0.6565 - val_accuracy: 0.7020 - val_loss: 0.6664
Epoch 19/600
148/148 1s 10ms/step - accuracy: 0.7109 - loss: 0.6429 - val_accuracy: 0.7030 - val_loss: 0.6632
Epoch 20/600
148/148 1s 8ms/step - accuracy: 0.7096 - loss: 0.6438 - val_accuracy: 0.7031 - val_loss: 0.6593
Epoch 21/600
148/148 1s 9ms/step - accuracy: 0.7151 - loss: 0.6337 - val_accuracy: 0.7048 - val_loss: 0.6618
Epoch 22/600
148/148 1s 9ms/step - accuracy: 0.7114 - loss: 0.6390 - val_accuracy: 0.7058 - val_loss: 0.6587
Epoch 23/600
148/148 1s 8ms/step - accuracy: 0.7155 - loss: 0.6325 - val_accuracy: 0.7082 - val_loss: 0.6554
Epoch 24/600
148/148 1s 8ms/step - accuracy: 0.7176 - loss: 0.6268 - val_accuracy: 0.7079 - val_loss: 0.6508
Epoch 25/600
148/148 1s 7ms/step - accuracy: 0.7171 - loss: 0.6250 - val_accuracy: 0.7118 - val_loss: 0.6468
Epoch 26/600

148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.7210 - loss: 0.6183 - val_accuracy: 0.7098 - val_loss: 0.6443
Epoch 27/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.7252 - loss: 0.6125 - val_accuracy: 0.7112 - val_loss: 0.6450
Epoch 28/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7265 - loss: 0.6103 - val_accuracy: 0.7154 - val_loss: 0.6429
Epoch 29/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7255 - loss: 0.6115 - val_accuracy: 0.7152 - val_loss: 0.6403
Epoch 30/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.7332 - loss: 0.6032 - val_accuracy: 0.7171 - val_loss: 0.6392
Epoch 31/600
148/148 ━━━━━━ 1s 10ms/step - accuracy: 0.7343 - loss: 0.5973 - val_accuracy: 0.7182 - val_loss: 0.6396
Epoch 32/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.7321 - loss: 0.6009 - val_accuracy: 0.7180 - val_loss: 0.6390
Epoch 33/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.7336 - loss: 0.5978 - val_accuracy: 0.7205 - val_loss: 0.6317
Epoch 34/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.7335 - loss: 0.5954 - val_accuracy: 0.7236 - val_loss: 0.6280
Epoch 35/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7380 - loss: 0.5870 - val_accuracy: 0.7216 - val_loss: 0.6289
Epoch 36/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7416 - loss: 0.5832 - val_accuracy: 0.7225 - val_loss: 0.6250
Epoch 37/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7379 - loss: 0.5842 - val_accuracy: 0.7244 - val_loss: 0.6240
Epoch 38/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7419 - loss: 0.5786 - val_accuracy: 0.7219 - val_loss: 0.6255
Epoch 39/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.7417 - loss: 0.5796 - val_accuracy: 0.7251 - val_loss: 0.6223
Epoch 40/600
148/148 ━━━━━━ 2s 10ms/step - accuracy: 0.7433 - loss: 0.5774 - val_accuracy: 0.7214 - val_loss: 0.6260
Epoch 41/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.7426 - loss: 0.5773 - val_accuracy: 0.7250 - val_loss: 0.6224
Epoch 42/600
148/148 ━━━━━━ 2s 11ms/step - accuracy: 0.7458 - loss: 0.5692 - val_accuracy: 0.7258 - val_loss: 0.6173
Epoch 43/600
148/148 ━━━━━━ 1s 10ms/step - accuracy: 0.7486 - loss: 0.5650 - val_accuracy: 0.7297 - val_loss: 0.6137
Epoch 44/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.7469 - loss: 0.5673 - val_accuracy: 0.7302 - val_loss: 0.6141
Epoch 45/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7525 - loss: 0.5594 - val_accuracy: 0.7321 - val_loss: 0.6134
Epoch 46/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7506 - loss: 0.5641 - val_accuracy: 0.7330 - val_loss: 0.6151
Epoch 47/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7504 - loss: 0.5637 - val_accuracy: 0.7324 - val_loss: 0.6121
Epoch 48/600
148/148 ━━━━━━ 2s 10ms/step - accuracy: 0.7502 - loss: 0.5614 - val_accuracy: 0.7328 - val_loss: 0.6102
Epoch 49/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.7499 - loss: 0.5607 - val_accuracy: 0.7349 - val_loss: 0.6102
Epoch 50/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.7576 - loss: 0.5521 - val_accuracy: 0.7318 - val_loss: 0.6098
Epoch 51/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.7538 - loss: 0.5525 - val_accuracy: 0.7378 - val_loss:

s: 0.6062
Epoch 52/600
148/148 1s 8ms/step - accuracy: 0.7531 - loss: 0.5530 - val_accuracy: 0.7355 - val_loss: 0.5530
s: 0.6043
Epoch 53/600
148/148 1s 8ms/step - accuracy: 0.7608 - loss: 0.5416 - val_accuracy: 0.7370 - val_loss: 0.5416
s: 0.6023
Epoch 54/600
148/148 1s 8ms/step - accuracy: 0.7592 - loss: 0.5424 - val_accuracy: 0.7372 - val_loss: 0.5424
s: 0.6024
Epoch 55/600
148/148 1s 8ms/step - accuracy: 0.7606 - loss: 0.5422 - val_accuracy: 0.7386 - val_loss: 0.5422
s: 0.6023
Epoch 56/600
148/148 1s 8ms/step - accuracy: 0.7603 - loss: 0.5406 - val_accuracy: 0.7378 - val_loss: 0.5406
s: 0.6030
Epoch 57/600
148/148 1s 7ms/step - accuracy: 0.7627 - loss: 0.5406 - val_accuracy: 0.7388 - val_loss: 0.5406
s: 0.6034
Epoch 58/600
148/148 1s 9ms/step - accuracy: 0.7659 - loss: 0.5354 - val_accuracy: 0.7410 - val_loss: 0.5354
s: 0.6040
Epoch 59/600
148/148 1s 9ms/step - accuracy: 0.7655 - loss: 0.5326 - val_accuracy: 0.7384 - val_loss: 0.5326
s: 0.6024
Epoch 60/600
148/148 1s 8ms/step - accuracy: 0.7631 - loss: 0.5320 - val_accuracy: 0.7434 - val_loss: 0.5320
s: 0.5986
Epoch 61/600
148/148 1s 9ms/step - accuracy: 0.7634 - loss: 0.5332 - val_accuracy: 0.7417 - val_loss: 0.5332
s: 0.6034
Epoch 62/600
148/148 1s 8ms/step - accuracy: 0.7654 - loss: 0.5315 - val_accuracy: 0.7424 - val_loss: 0.5315
s: 0.5992
Epoch 63/600
148/148 1s 9ms/step - accuracy: 0.7637 - loss: 0.5340 - val_accuracy: 0.7435 - val_loss: 0.5340
s: 0.5955
Epoch 64/600
148/148 1s 8ms/step - accuracy: 0.7652 - loss: 0.5293 - val_accuracy: 0.7422 - val_loss: 0.5293
s: 0.5951
Epoch 65/600
148/148 1s 7ms/step - accuracy: 0.7689 - loss: 0.5297 - val_accuracy: 0.7434 - val_loss: 0.5297
s: 0.6001
Epoch 66/600
148/148 1s 9ms/step - accuracy: 0.7682 - loss: 0.5275 - val_accuracy: 0.7387 - val_loss: 0.5275
s: 0.6000
Epoch 67/600
148/148 1s 8ms/step - accuracy: 0.7707 - loss: 0.5277 - val_accuracy: 0.7453 - val_loss: 0.5277
s: 0.5957
Epoch 68/600
148/148 1s 9ms/step - accuracy: 0.7688 - loss: 0.5249 - val_accuracy: 0.7440 - val_loss: 0.5249
s: 0.5950
Epoch 69/600
148/148 1s 8ms/step - accuracy: 0.7683 - loss: 0.5237 - val_accuracy: 0.7434 - val_loss: 0.5237
s: 0.5912
Epoch 70/600
148/148 1s 8ms/step - accuracy: 0.7730 - loss: 0.5177 - val_accuracy: 0.7460 - val_loss: 0.5177
s: 0.5930
Epoch 71/600
148/148 1s 9ms/step - accuracy: 0.7714 - loss: 0.5201 - val_accuracy: 0.7427 - val_loss: 0.5201
s: 0.5929
Epoch 72/600
148/148 1s 8ms/step - accuracy: 0.7728 - loss: 0.5182 - val_accuracy: 0.7468 - val_loss: 0.5182
s: 0.5925
Epoch 73/600
148/148 1s 7ms/step - accuracy: 0.7724 - loss: 0.5161 - val_accuracy: 0.7448 - val_loss: 0.5161
s: 0.5966
Epoch 74/600
148/148 1s 8ms/step - accuracy: 0.7762 - loss: 0.5133 - val_accuracy: 0.7450 - val_loss: 0.5133
s: 0.5937
Epoch 75/600
148/148 1s 8ms/step - accuracy: 0.7765 - loss: 0.5123 - val_accuracy: 0.7505 - val_loss: 0.5123
s: 0.5918
Epoch 76/600
148/148 1s 9ms/step - accuracy: 0.7739 - loss: 0.5141 - val_accuracy: 0.7478 - val_loss: 0.5141
s: 0.5876

Epoch 77/600
148/148 2s 10ms/step - accuracy: 0.7743 - loss: 0.5141 - val_accuracy: 0.7489 - val_loss: 0.5872
Epoch 78/600
148/148 1s 9ms/step - accuracy: 0.7774 - loss: 0.5087 - val_accuracy: 0.7480 - val_loss: 0.5852
Epoch 79/600
148/148 1s 9ms/step - accuracy: 0.7748 - loss: 0.5112 - val_accuracy: 0.7505 - val_loss: 0.5887
Epoch 80/600
148/148 1s 10ms/step - accuracy: 0.7777 - loss: 0.5078 - val_accuracy: 0.7505 - val_loss: 0.5915
Epoch 81/600
148/148 1s 8ms/step - accuracy: 0.7788 - loss: 0.5020 - val_accuracy: 0.7493 - val_loss: 0.5935
Epoch 82/600
148/148 1s 8ms/step - accuracy: 0.7792 - loss: 0.5070 - val_accuracy: 0.7484 - val_loss: 0.5917
Epoch 83/600
148/148 1s 8ms/step - accuracy: 0.7771 - loss: 0.5043 - val_accuracy: 0.7500 - val_loss: 0.5921
Epoch 84/600
148/148 1s 7ms/step - accuracy: 0.7786 - loss: 0.5021 - val_accuracy: 0.7509 - val_loss: 0.5915
Epoch 85/600
148/148 1s 7ms/step - accuracy: 0.7775 - loss: 0.5051 - val_accuracy: 0.7508 - val_loss: 0.5930
Epoch 86/600
148/148 1s 7ms/step - accuracy: 0.7803 - loss: 0.5024 - val_accuracy: 0.7462 - val_loss: 0.5909
Epoch 87/600
148/148 1s 7ms/step - accuracy: 0.7807 - loss: 0.5002 - val_accuracy: 0.7485 - val_loss: 0.5906
Epoch 88/600
148/148 1s 8ms/step - accuracy: 0.7832 - loss: 0.4985 - val_accuracy: 0.7519 - val_loss: 0.5943
Epoch 89/600
148/148 1s 7ms/step - accuracy: 0.7858 - loss: 0.4948 - val_accuracy: 0.7536 - val_loss: 0.5912
Epoch 90/600
148/148 1s 7ms/step - accuracy: 0.7856 - loss: 0.4923 - val_accuracy: 0.7518 - val_loss: 0.5880
Epoch 91/600
148/148 1s 7ms/step - accuracy: 0.7854 - loss: 0.4953 - val_accuracy: 0.7520 - val_loss: 0.5847
Epoch 92/600
148/148 1s 8ms/step - accuracy: 0.7830 - loss: 0.4994 - val_accuracy: 0.7513 - val_loss: 0.5876
Epoch 93/600
148/148 1s 8ms/step - accuracy: 0.7876 - loss: 0.4909 - val_accuracy: 0.7534 - val_loss: 0.5865
Epoch 94/600
148/148 1s 9ms/step - accuracy: 0.7821 - loss: 0.4991 - val_accuracy: 0.7508 - val_loss: 0.5900
Epoch 95/600
148/148 1s 7ms/step - accuracy: 0.7867 - loss: 0.4881 - val_accuracy: 0.7564 - val_loss: 0.5854
Epoch 96/600
148/148 1s 7ms/step - accuracy: 0.7848 - loss: 0.4919 - val_accuracy: 0.7545 - val_loss: 0.5854
Epoch 97/600
148/148 1s 7ms/step - accuracy: 0.7869 - loss: 0.4898 - val_accuracy: 0.7538 - val_loss: 0.5857
Epoch 98/600
148/148 1s 7ms/step - accuracy: 0.7869 - loss: 0.4880 - val_accuracy: 0.7551 - val_loss: 0.5858
Epoch 99/600
148/148 1s 7ms/step - accuracy: 0.7863 - loss: 0.4881 - val_accuracy: 0.7529 - val_loss: 0.5873
Epoch 100/600
148/148 1s 7ms/step - accuracy: 0.7871 - loss: 0.4913 - val_accuracy: 0.7508 - val_loss: 0.5891
Epoch 101/600
148/148 1s 7ms/step - accuracy: 0.7886 - loss: 0.4879 - val_accuracy: 0.7542 - val_loss: 0.5823
Epoch 102/600

148/148 1s 8ms/step - accuracy: 0.7866 - loss: 0.4873 - val_accuracy: 0.7532 - val_loss: 0.5894
Epoch 103/600
148/148 1s 7ms/step - accuracy: 0.7877 - loss: 0.4886 - val_accuracy: 0.7546 - val_loss: 0.5897
Epoch 104/600
148/148 1s 7ms/step - accuracy: 0.7907 - loss: 0.4831 - val_accuracy: 0.7556 - val_loss: 0.5842
Epoch 105/600
148/148 1s 9ms/step - accuracy: 0.7920 - loss: 0.4815 - val_accuracy: 0.7568 - val_loss: 0.5773
Epoch 106/600
148/148 1s 9ms/step - accuracy: 0.7923 - loss: 0.4811 - val_accuracy: 0.7568 - val_loss: 0.5845
Epoch 107/600
148/148 1s 8ms/step - accuracy: 0.7898 - loss: 0.4845 - val_accuracy: 0.7528 - val_loss: 0.5877
Epoch 108/600
148/148 1s 9ms/step - accuracy: 0.7914 - loss: 0.4767 - val_accuracy: 0.7552 - val_loss: 0.5806
Epoch 109/600
148/148 1s 8ms/step - accuracy: 0.7925 - loss: 0.4781 - val_accuracy: 0.7572 - val_loss: 0.5790
Epoch 110/600
148/148 1s 7ms/step - accuracy: 0.7927 - loss: 0.4763 - val_accuracy: 0.7567 - val_loss: 0.5844
Epoch 111/600
148/148 1s 7ms/step - accuracy: 0.7903 - loss: 0.4808 - val_accuracy: 0.7569 - val_loss: 0.5739
Epoch 112/600
148/148 1s 7ms/step - accuracy: 0.7923 - loss: 0.4773 - val_accuracy: 0.7576 - val_loss: 0.5817
Epoch 113/600
148/148 1s 8ms/step - accuracy: 0.7916 - loss: 0.4804 - val_accuracy: 0.7574 - val_loss: 0.5821
Epoch 114/600
148/148 1s 8ms/step - accuracy: 0.7951 - loss: 0.4778 - val_accuracy: 0.7546 - val_loss: 0.5825
Epoch 115/600
148/148 1s 7ms/step - accuracy: 0.7915 - loss: 0.4796 - val_accuracy: 0.7587 - val_loss: 0.5806
Epoch 116/600
148/148 1s 7ms/step - accuracy: 0.7921 - loss: 0.4788 - val_accuracy: 0.7570 - val_loss: 0.5812
Epoch 117/600
148/148 1s 7ms/step - accuracy: 0.7930 - loss: 0.4748 - val_accuracy: 0.7564 - val_loss: 0.5795
Epoch 118/600
148/148 1s 7ms/step - accuracy: 0.7957 - loss: 0.4694 - val_accuracy: 0.7526 - val_loss: 0.5821
Epoch 119/600
148/148 1s 9ms/step - accuracy: 0.7943 - loss: 0.4738 - val_accuracy: 0.7530 - val_loss: 0.5791
Epoch 120/600
148/148 1s 8ms/step - accuracy: 0.7948 - loss: 0.4728 - val_accuracy: 0.7534 - val_loss: 0.5825
Epoch 121/600
148/148 1s 9ms/step - accuracy: 0.7955 - loss: 0.4681 - val_accuracy: 0.7579 - val_loss: 0.5846
Epoch 122/600
148/148 1s 8ms/step - accuracy: 0.7974 - loss: 0.4689 - val_accuracy: 0.7571 - val_loss: 0.5827
Epoch 123/600
148/148 1s 8ms/step - accuracy: 0.7942 - loss: 0.4724 - val_accuracy: 0.7578 - val_loss: 0.5762
Epoch 124/600
148/148 1s 9ms/step - accuracy: 0.7973 - loss: 0.4709 - val_accuracy: 0.7581 - val_loss: 0.5765
Epoch 125/600
148/148 1s 8ms/step - accuracy: 0.7969 - loss: 0.4670 - val_accuracy: 0.7589 - val_loss: 0.5737
Epoch 126/600
148/148 1s 8ms/step - accuracy: 0.7979 - loss: 0.4699 - val_accuracy: 0.7574 - val_loss: 0.5783
Epoch 127/600
148/148 1s 8ms/step - accuracy: 0.7938 - loss: 0.4732 - val_accuracy: 0.7568 - val_loss:

s: 0.5861
Epoch 128/600
148/148 1s 8ms/step - accuracy: 0.7982 - loss: 0.4664 - val_accuracy: 0.7578 - val_loss: 0.5886
s: 0.5886
Epoch 129/600
148/148 1s 9ms/step - accuracy: 0.7952 - loss: 0.4699 - val_accuracy: 0.7576 - val_loss: 0.5871
s: 0.5871
Epoch 130/600
148/148 1s 7ms/step - accuracy: 0.7968 - loss: 0.4695 - val_accuracy: 0.7592 - val_loss: 0.5775
s: 0.5775
Epoch 131/600
148/148 1s 8ms/step - accuracy: 0.8019 - loss: 0.4636 - val_accuracy: 0.7588 - val_loss: 0.5812
s: 0.5812
Epoch 132/600
148/148 1s 9ms/step - accuracy: 0.7985 - loss: 0.4629 - val_accuracy: 0.7567 - val_loss: 0.5829
s: 0.5829
Epoch 133/600
148/148 1s 9ms/step - accuracy: 0.7994 - loss: 0.4627 - val_accuracy: 0.7578 - val_loss: 0.5854
s: 0.5854
Epoch 134/600
148/148 1s 7ms/step - accuracy: 0.7979 - loss: 0.4661 - val_accuracy: 0.7579 - val_loss: 0.5905
s: 0.5905
Epoch 135/600
148/148 1s 8ms/step - accuracy: 0.7976 - loss: 0.4650 - val_accuracy: 0.7611 - val_loss: 0.5812
s: 0.5812
Epoch 136/600
148/148 1s 7ms/step - accuracy: 0.8002 - loss: 0.4622 - val_accuracy: 0.7564 - val_loss: 0.5833
s: 0.5833
Epoch 137/600
148/148 1s 7ms/step - accuracy: 0.8016 - loss: 0.4596 - val_accuracy: 0.7577 - val_loss: 0.5792
s: 0.5792
Epoch 138/600
148/148 1s 7ms/step - accuracy: 0.7997 - loss: 0.4611 - val_accuracy: 0.7573 - val_loss: 0.5779
s: 0.5779
Epoch 139/600
148/148 1s 7ms/step - accuracy: 0.7989 - loss: 0.4649 - val_accuracy: 0.7591 - val_loss: 0.5849
s: 0.5849
Epoch 140/600
148/148 1s 7ms/step - accuracy: 0.7989 - loss: 0.4644 - val_accuracy: 0.7587 - val_loss: 0.5767
s: 0.5767
Epoch 141/600
148/148 1s 7ms/step - accuracy: 0.7990 - loss: 0.4617 - val_accuracy: 0.7577 - val_loss: 0.5790
s: 0.5790
Epoch 142/600
148/148 1s 7ms/step - accuracy: 0.8005 - loss: 0.4630 - val_accuracy: 0.7605 - val_loss: 0.5785
s: 0.5785
Epoch 143/600
148/148 1s 7ms/step - accuracy: 0.8010 - loss: 0.4569 - val_accuracy: 0.7586 - val_loss: 0.5803
s: 0.5803
Epoch 144/600
148/148 1s 7ms/step - accuracy: 0.8008 - loss: 0.4632 - val_accuracy: 0.7572 - val_loss: 0.5858
s: 0.5858
Epoch 145/600
148/148 1s 8ms/step - accuracy: 0.8026 - loss: 0.4562 - val_accuracy: 0.7574 - val_loss: 0.5823
s: 0.5823
Epoch 146/600
148/148 1s 7ms/step - accuracy: 0.7998 - loss: 0.4623 - val_accuracy: 0.7600 - val_loss: 0.5744
s: 0.5744
Epoch 147/600
148/148 1s 8ms/step - accuracy: 0.8014 - loss: 0.4596 - val_accuracy: 0.7563 - val_loss: 0.5827
s: 0.5827
Epoch 148/600
148/148 1s 7ms/step - accuracy: 0.7999 - loss: 0.4608 - val_accuracy: 0.7587 - val_loss: 0.5810
s: 0.5810
Epoch 149/600
148/148 1s 8ms/step - accuracy: 0.8066 - loss: 0.4518 - val_accuracy: 0.7620 - val_loss: 0.5787
s: 0.5787
Epoch 150/600
148/148 1s 8ms/step - accuracy: 0.8045 - loss: 0.4548 - val_accuracy: 0.7606 - val_loss: 0.5763
s: 0.5763
Epoch 151/600
148/148 1s 8ms/step - accuracy: 0.8002 - loss: 0.4582 - val_accuracy: 0.7622 - val_loss: 0.5925
s: 0.5925
Epoch 152/600
148/148 1s 7ms/step - accuracy: 0.8037 - loss: 0.4568 - val_accuracy: 0.7614 - val_loss: 0.5814
s: 0.5814

Epoch 153/600
148/148 1s 7ms/step - accuracy: 0.8063 - loss: 0.4500 - val_accuracy: 0.7607 - val_loss: 0.5840
Epoch 154/600
148/148 1s 7ms/step - accuracy: 0.8083 - loss: 0.4519 - val_accuracy: 0.7608 - val_loss: 0.5736
Epoch 155/600
148/148 1s 7ms/step - accuracy: 0.8055 - loss: 0.4500 - val_accuracy: 0.7598 - val_loss: 0.5812
Epoch 156/600
148/148 1s 7ms/step - accuracy: 0.8037 - loss: 0.4541 - val_accuracy: 0.7605 - val_loss: 0.5780
Epoch 157/600
148/148 1s 7ms/step - accuracy: 0.8073 - loss: 0.4494 - val_accuracy: 0.7600 - val_loss: 0.5786
Epoch 158/600
148/148 1s 7ms/step - accuracy: 0.8069 - loss: 0.4501 - val_accuracy: 0.7592 - val_loss: 0.5745
Epoch 159/600
148/148 1s 7ms/step - accuracy: 0.8056 - loss: 0.4513 - val_accuracy: 0.7616 - val_loss: 0.5763
Epoch 160/600
148/148 1s 7ms/step - accuracy: 0.8050 - loss: 0.4509 - val_accuracy: 0.7596 - val_loss: 0.5810
Epoch 161/600
148/148 1s 8ms/step - accuracy: 0.8055 - loss: 0.4527 - val_accuracy: 0.7610 - val_loss: 0.5779
Epoch 162/600
148/148 1s 7ms/step - accuracy: 0.8083 - loss: 0.4477 - val_accuracy: 0.7562 - val_loss: 0.5826
Epoch 163/600
148/148 1s 8ms/step - accuracy: 0.8036 - loss: 0.4518 - val_accuracy: 0.7636 - val_loss: 0.5738
Epoch 164/600
148/148 1s 7ms/step - accuracy: 0.8071 - loss: 0.4490 - val_accuracy: 0.7663 - val_loss: 0.5809
Epoch 165/600
148/148 1s 8ms/step - accuracy: 0.8062 - loss: 0.4512 - val_accuracy: 0.7603 - val_loss: 0.5829
Epoch 166/600
148/148 1s 7ms/step - accuracy: 0.8048 - loss: 0.4512 - val_accuracy: 0.7653 - val_loss: 0.5774
Epoch 167/600
148/148 1s 8ms/step - accuracy: 0.8032 - loss: 0.4527 - val_accuracy: 0.7636 - val_loss: 0.5819
Epoch 168/600
148/148 1s 7ms/step - accuracy: 0.8065 - loss: 0.4485 - val_accuracy: 0.7631 - val_loss: 0.5859
Epoch 169/600
148/148 1s 7ms/step - accuracy: 0.8071 - loss: 0.4494 - val_accuracy: 0.7614 - val_loss: 0.5796
Epoch 170/600
148/148 1s 7ms/step - accuracy: 0.8083 - loss: 0.4479 - val_accuracy: 0.7637 - val_loss: 0.5742
Epoch 171/600
148/148 1s 8ms/step - accuracy: 0.8073 - loss: 0.4461 - val_accuracy: 0.7650 - val_loss: 0.5770
Epoch 172/600
148/148 1s 7ms/step - accuracy: 0.8074 - loss: 0.4473 - val_accuracy: 0.7652 - val_loss: 0.5811
Epoch 173/600
148/148 1s 8ms/step - accuracy: 0.8085 - loss: 0.4444 - val_accuracy: 0.7614 - val_loss: 0.5736
Epoch 174/600
148/148 1s 8ms/step - accuracy: 0.8071 - loss: 0.4448 - val_accuracy: 0.7640 - val_loss: 0.5860
Epoch 175/600
148/148 1s 8ms/step - accuracy: 0.8059 - loss: 0.4455 - val_accuracy: 0.7630 - val_loss: 0.5832
Epoch 176/600
148/148 1s 8ms/step - accuracy: 0.8092 - loss: 0.4459 - val_accuracy: 0.7645 - val_loss: 0.5756
Epoch 177/600
148/148 1s 9ms/step - accuracy: 0.8116 - loss: 0.4400 - val_accuracy: 0.7646 - val_loss: 0.5770
Epoch 178/600

148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8073 - loss: 0.4468 - val_accuracy: 0.7618 - val_loss: 0.5884
Epoch 179/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8087 - loss: 0.4431 - val_accuracy: 0.7619 - val_loss: 0.5787
Epoch 180/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.8061 - loss: 0.4456 - val_accuracy: 0.7611 - val_loss: 0.5834
Epoch 181/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8082 - loss: 0.4459 - val_accuracy: 0.7664 - val_loss: 0.5834
Epoch 182/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8101 - loss: 0.4426 - val_accuracy: 0.7654 - val_loss: 0.5805
Epoch 183/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8076 - loss: 0.4431 - val_accuracy: 0.7639 - val_loss: 0.5735
Epoch 184/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8085 - loss: 0.4456 - val_accuracy: 0.7631 - val_loss: 0.5734
Epoch 185/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8108 - loss: 0.4399 - val_accuracy: 0.7631 - val_loss: 0.5769
Epoch 186/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.8097 - loss: 0.4430 - val_accuracy: 0.7617 - val_loss: 0.5830
Epoch 187/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8139 - loss: 0.4371 - val_accuracy: 0.7623 - val_loss: 0.5848
Epoch 188/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8112 - loss: 0.4405 - val_accuracy: 0.7647 - val_loss: 0.5780
Epoch 189/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.8110 - loss: 0.4404 - val_accuracy: 0.7657 - val_loss: 0.5717
Epoch 190/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8144 - loss: 0.4372 - val_accuracy: 0.7651 - val_loss: 0.5790
Epoch 191/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8129 - loss: 0.4365 - val_accuracy: 0.7644 - val_loss: 0.5740
Epoch 192/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8128 - loss: 0.4381 - val_accuracy: 0.7650 - val_loss: 0.5820
Epoch 193/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8110 - loss: 0.4402 - val_accuracy: 0.7643 - val_loss: 0.5760
Epoch 194/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8130 - loss: 0.4350 - val_accuracy: 0.7622 - val_loss: 0.5833
Epoch 195/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8128 - loss: 0.4380 - val_accuracy: 0.7630 - val_loss: 0.5847
Epoch 196/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8092 - loss: 0.4407 - val_accuracy: 0.7635 - val_loss: 0.5810
Epoch 197/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8119 - loss: 0.4357 - val_accuracy: 0.7622 - val_loss: 0.5778
Epoch 198/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8109 - loss: 0.4366 - val_accuracy: 0.7634 - val_loss: 0.5742
Epoch 199/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8106 - loss: 0.4369 - val_accuracy: 0.7659 - val_loss: 0.5752
Epoch 200/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8152 - loss: 0.4381 - val_accuracy: 0.7652 - val_loss: 0.5763
Epoch 201/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8107 - loss: 0.4378 - val_accuracy: 0.7654 - val_loss: 0.5799
Epoch 202/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8105 - loss: 0.4401 - val_accuracy: 0.7617 - val_loss: 0.5755
Epoch 203/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8119 - loss: 0.4362 - val_accuracy: 0.7634 - val_loss:

s: 0.5886
Epoch 204/600
148/148 1s 8ms/step - accuracy: 0.8147 - loss: 0.4325 - val_accuracy: 0.7644 - val_loss: 0.5800
Epoch 205/600
148/148 1s 8ms/step - accuracy: 0.8121 - loss: 0.4359 - val_accuracy: 0.7619 - val_loss: 0.5723
Epoch 206/600
148/148 1s 8ms/step - accuracy: 0.8130 - loss: 0.4363 - val_accuracy: 0.7617 - val_loss: 0.5859
Epoch 207/600
148/148 1s 7ms/step - accuracy: 0.8124 - loss: 0.4358 - val_accuracy: 0.7603 - val_loss: 0.5835
Epoch 208/600
148/148 1s 7ms/step - accuracy: 0.8149 - loss: 0.4329 - val_accuracy: 0.7625 - val_loss: 0.5844
Epoch 209/600
148/148 1s 7ms/step - accuracy: 0.8118 - loss: 0.4367 - val_accuracy: 0.7597 - val_loss: 0.5812
Epoch 210/600
148/148 1s 7ms/step - accuracy: 0.8139 - loss: 0.4309 - val_accuracy: 0.7667 - val_loss: 0.5773
Epoch 211/600
148/148 1s 7ms/step - accuracy: 0.8163 - loss: 0.4311 - val_accuracy: 0.7637 - val_loss: 0.5811
Epoch 212/600
148/148 1s 7ms/step - accuracy: 0.8131 - loss: 0.4323 - val_accuracy: 0.7636 - val_loss: 0.5831
Epoch 213/600
148/148 1s 7ms/step - accuracy: 0.8136 - loss: 0.4350 - val_accuracy: 0.7645 - val_loss: 0.5811
Epoch 214/600
148/148 1s 7ms/step - accuracy: 0.8121 - loss: 0.4347 - val_accuracy: 0.7623 - val_loss: 0.5823
Epoch 215/600
148/148 1s 8ms/step - accuracy: 0.8136 - loss: 0.4330 - val_accuracy: 0.7642 - val_loss: 0.5798
Epoch 216/600
148/148 1s 8ms/step - accuracy: 0.8165 - loss: 0.4288 - val_accuracy: 0.7628 - val_loss: 0.5783
Epoch 217/600
148/148 1s 10ms/step - accuracy: 0.8138 - loss: 0.4333 - val_accuracy: 0.7615 - val_loss: 0.5883
Epoch 218/600
148/148 1s 9ms/step - accuracy: 0.8164 - loss: 0.4290 - val_accuracy: 0.7633 - val_loss: 0.5853
Epoch 219/600
148/148 1s 10ms/step - accuracy: 0.8181 - loss: 0.4265 - val_accuracy: 0.7648 - val_loss: 0.5917
Epoch 220/600
148/148 1s 9ms/step - accuracy: 0.8163 - loss: 0.4306 - val_accuracy: 0.7665 - val_loss: 0.5733
Epoch 221/600
148/148 1s 9ms/step - accuracy: 0.8169 - loss: 0.4304 - val_accuracy: 0.7635 - val_loss: 0.5728
Epoch 222/600
148/148 1s 8ms/step - accuracy: 0.8194 - loss: 0.4224 - val_accuracy: 0.7673 - val_loss: 0.5765
Epoch 223/600
148/148 1s 8ms/step - accuracy: 0.8170 - loss: 0.4267 - val_accuracy: 0.7655 - val_loss: 0.5874
Epoch 224/600
148/148 1s 8ms/step - accuracy: 0.8142 - loss: 0.4320 - val_accuracy: 0.7636 - val_loss: 0.5839
Epoch 225/600
148/148 1s 8ms/step - accuracy: 0.8174 - loss: 0.4308 - val_accuracy: 0.7628 - val_loss: 0.5768
Epoch 226/600
148/148 1s 7ms/step - accuracy: 0.8162 - loss: 0.4286 - val_accuracy: 0.7665 - val_loss: 0.5840
Epoch 227/600
148/148 1s 8ms/step - accuracy: 0.8157 - loss: 0.4317 - val_accuracy: 0.7669 - val_loss: 0.5737
Epoch 228/600
148/148 2s 11ms/step - accuracy: 0.8161 - loss: 0.4260 - val_accuracy: 0.7640 - val_loss: 0.5867

Epoch 229/600
148/148 1s 8ms/step - accuracy: 0.8168 - loss: 0.4260 - val_accuracy: 0.7652 - val_loss: 0.5837
Epoch 230/600
148/148 1s 8ms/step - accuracy: 0.8157 - loss: 0.4299 - val_accuracy: 0.7650 - val_loss: 0.5833
Epoch 231/600
148/148 1s 8ms/step - accuracy: 0.8211 - loss: 0.4242 - val_accuracy: 0.7653 - val_loss: 0.5822
Epoch 232/600
148/148 1s 7ms/step - accuracy: 0.8149 - loss: 0.4293 - val_accuracy: 0.7634 - val_loss: 0.5827
Epoch 233/600
148/148 1s 8ms/step - accuracy: 0.8185 - loss: 0.4229 - val_accuracy: 0.7654 - val_loss: 0.5885
Epoch 234/600
148/148 1s 7ms/step - accuracy: 0.8172 - loss: 0.4237 - val_accuracy: 0.7650 - val_loss: 0.5766
Epoch 235/600
148/148 1s 8ms/step - accuracy: 0.8183 - loss: 0.4249 - val_accuracy: 0.7627 - val_loss: 0.5801
Epoch 236/600
148/148 1s 7ms/step - accuracy: 0.8175 - loss: 0.4243 - val_accuracy: 0.7647 - val_loss: 0.5776
Epoch 237/600
148/148 1s 7ms/step - accuracy: 0.8176 - loss: 0.4297 - val_accuracy: 0.7633 - val_loss: 0.5814
Epoch 238/600
148/148 1s 8ms/step - accuracy: 0.8215 - loss: 0.4198 - val_accuracy: 0.7608 - val_loss: 0.5790
Epoch 239/600
148/148 1s 7ms/step - accuracy: 0.8197 - loss: 0.4224 - val_accuracy: 0.7645 - val_loss: 0.5906
Epoch 240/600
148/148 1s 7ms/step - accuracy: 0.8199 - loss: 0.4211 - val_accuracy: 0.7627 - val_loss: 0.5806
Epoch 241/600
148/148 1s 8ms/step - accuracy: 0.8177 - loss: 0.4246 - val_accuracy: 0.7638 - val_loss: 0.5853
Epoch 242/600
148/148 1s 8ms/step - accuracy: 0.8165 - loss: 0.4257 - val_accuracy: 0.7627 - val_loss: 0.5953
Epoch 243/600
148/148 1s 8ms/step - accuracy: 0.8199 - loss: 0.4279 - val_accuracy: 0.7648 - val_loss: 0.5808
Epoch 244/600
148/148 1s 8ms/step - accuracy: 0.8166 - loss: 0.4259 - val_accuracy: 0.7639 - val_loss: 0.5850
Epoch 245/600
148/148 1s 8ms/step - accuracy: 0.8209 - loss: 0.4226 - val_accuracy: 0.7649 - val_loss: 0.5847
Epoch 246/600
148/148 1s 8ms/step - accuracy: 0.8200 - loss: 0.4217 - val_accuracy: 0.7641 - val_loss: 0.5813
Epoch 247/600
148/148 1s 7ms/step - accuracy: 0.8236 - loss: 0.4157 - val_accuracy: 0.7647 - val_loss: 0.5754
Epoch 248/600
148/148 1s 8ms/step - accuracy: 0.8207 - loss: 0.4216 - val_accuracy: 0.7654 - val_loss: 0.5764
Epoch 249/600
148/148 1s 8ms/step - accuracy: 0.8197 - loss: 0.4204 - val_accuracy: 0.7631 - val_loss: 0.5785
Epoch 250/600
148/148 1s 8ms/step - accuracy: 0.8218 - loss: 0.4186 - val_accuracy: 0.7675 - val_loss: 0.5748
Epoch 251/600
148/148 1s 8ms/step - accuracy: 0.8200 - loss: 0.4218 - val_accuracy: 0.7670 - val_loss: 0.5816
Epoch 252/600
148/148 1s 9ms/step - accuracy: 0.8185 - loss: 0.4262 - val_accuracy: 0.7668 - val_loss: 0.5824
Epoch 253/600
148/148 1s 9ms/step - accuracy: 0.8207 - loss: 0.4235 - val_accuracy: 0.7658 - val_loss: 0.5754
Epoch 254/600

148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8166 - loss: 0.4259 - val_accuracy: 0.7665 - val_loss: 0.5797
Epoch 255/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8195 - loss: 0.4211 - val_accuracy: 0.7671 - val_loss: 0.5715
Epoch 256/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.8194 - loss: 0.4226 - val_accuracy: 0.7668 - val_loss: 0.5795
Epoch 257/600
148/148 ━━━━━━ 2s 10ms/step - accuracy: 0.8175 - loss: 0.4254 - val_accuracy: 0.7680 - val_loss: 0.5829
Epoch 258/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.8191 - loss: 0.4201 - val_accuracy: 0.7677 - val_loss: 0.5805
Epoch 259/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8212 - loss: 0.4205 - val_accuracy: 0.7654 - val_loss: 0.5846
Epoch 260/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8216 - loss: 0.4204 - val_accuracy: 0.7677 - val_loss: 0.5816
Epoch 261/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8213 - loss: 0.4202 - val_accuracy: 0.7635 - val_loss: 0.5864
Epoch 262/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.8211 - loss: 0.4187 - val_accuracy: 0.7644 - val_loss: 0.5852
Epoch 263/600
148/148 ━━━━━━ 1s 10ms/step - accuracy: 0.8232 - loss: 0.4144 - val_accuracy: 0.7651 - val_loss: 0.5811
Epoch 264/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8202 - loss: 0.4195 - val_accuracy: 0.7645 - val_loss: 0.5951
Epoch 265/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.8181 - loss: 0.4252 - val_accuracy: 0.7667 - val_loss: 0.5859
Epoch 266/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8197 - loss: 0.4230 - val_accuracy: 0.7644 - val_loss: 0.5862
Epoch 267/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8162 - loss: 0.4251 - val_accuracy: 0.7644 - val_loss: 0.5749
Epoch 268/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8202 - loss: 0.4184 - val_accuracy: 0.7650 - val_loss: 0.5807
Epoch 269/600
148/148 ━━━━━━ 2s 10ms/step - accuracy: 0.8227 - loss: 0.4173 - val_accuracy: 0.7622 - val_loss: 0.5910
Epoch 270/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.8217 - loss: 0.4137 - val_accuracy: 0.7643 - val_loss: 0.5780
Epoch 271/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.8226 - loss: 0.4185 - val_accuracy: 0.7636 - val_loss: 0.5829
Epoch 272/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8189 - loss: 0.4200 - val_accuracy: 0.7670 - val_loss: 0.5765
Epoch 273/600
148/148 ━━━━━━ 1s 9ms/step - accuracy: 0.8214 - loss: 0.4182 - val_accuracy: 0.7653 - val_loss: 0.5875
Epoch 274/600
148/148 ━━━━━━ 1s 10ms/step - accuracy: 0.8216 - loss: 0.4176 - val_accuracy: 0.7632 - val_loss: 0.5736
Epoch 275/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8220 - loss: 0.4160 - val_accuracy: 0.7677 - val_loss: 0.5899
Epoch 276/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8212 - loss: 0.4187 - val_accuracy: 0.7681 - val_loss: 0.5722
Epoch 277/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8206 - loss: 0.4188 - val_accuracy: 0.7688 - val_loss: 0.5783
Epoch 278/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.8210 - loss: 0.4197 - val_accuracy: 0.7672 - val_loss: 0.5804
Epoch 279/600
148/148 ━━━━━━ 1s 8ms/step - accuracy: 0.8208 - loss: 0.4190 - val_accuracy: 0.7634 - val_loss:

s: 0.5792
Epoch 280/600
148/148 1s 8ms/step - accuracy: 0.8213 - loss: 0.4153 - val_accuracy: 0.7677 - val_loss: 0.5789
Epoch 281/600
148/148 1s 9ms/step - accuracy: 0.8211 - loss: 0.4173 - val_accuracy: 0.7671 - val_loss: 0.5775
Epoch 282/600
148/148 1s 9ms/step - accuracy: 0.8208 - loss: 0.4186 - val_accuracy: 0.7651 - val_loss: 0.5728
Epoch 283/600
148/148 1s 9ms/step - accuracy: 0.8223 - loss: 0.4143 - val_accuracy: 0.7646 - val_loss: 0.5749
Epoch 284/600
148/148 1s 8ms/step - accuracy: 0.8225 - loss: 0.4159 - val_accuracy: 0.7663 - val_loss: 0.5778
Epoch 285/600
148/148 1s 8ms/step - accuracy: 0.8223 - loss: 0.4122 - val_accuracy: 0.7669 - val_loss: 0.5859
Epoch 286/600
148/148 1s 8ms/step - accuracy: 0.8202 - loss: 0.4173 - val_accuracy: 0.7663 - val_loss: 0.5809
Epoch 287/600
148/148 1s 7ms/step - accuracy: 0.8217 - loss: 0.4164 - val_accuracy: 0.7637 - val_loss: 0.5776
Epoch 288/600
148/148 1s 9ms/step - accuracy: 0.8211 - loss: 0.4197 - val_accuracy: 0.7662 - val_loss: 0.5856
Epoch 289/600
148/148 1s 8ms/step - accuracy: 0.8217 - loss: 0.4119 - val_accuracy: 0.7655 - val_loss: 0.5821
Epoch 290/600
148/148 1s 8ms/step - accuracy: 0.8246 - loss: 0.4152 - val_accuracy: 0.7644 - val_loss: 0.5819
Epoch 291/600
148/148 1s 7ms/step - accuracy: 0.8199 - loss: 0.4172 - val_accuracy: 0.7664 - val_loss: 0.5771
Epoch 292/600
148/148 1s 7ms/step - accuracy: 0.8258 - loss: 0.4103 - val_accuracy: 0.7676 - val_loss: 0.5796
Epoch 293/600
148/148 1s 7ms/step - accuracy: 0.8233 - loss: 0.4143 - val_accuracy: 0.7638 - val_loss: 0.5930
Epoch 294/600
148/148 1s 7ms/step - accuracy: 0.8248 - loss: 0.4085 - val_accuracy: 0.7631 - val_loss: 0.5799
Epoch 295/600
148/148 1s 8ms/step - accuracy: 0.8187 - loss: 0.4170 - val_accuracy: 0.7645 - val_loss: 0.5833
Epoch 296/600
148/148 1s 7ms/step - accuracy: 0.8252 - loss: 0.4084 - val_accuracy: 0.7656 - val_loss: 0.5873
Epoch 297/600
148/148 1s 7ms/step - accuracy: 0.8240 - loss: 0.4090 - val_accuracy: 0.7662 - val_loss: 0.5795
Epoch 298/600
148/148 1s 7ms/step - accuracy: 0.8244 - loss: 0.4157 - val_accuracy: 0.7674 - val_loss: 0.5922
Epoch 299/600
148/148 1s 7ms/step - accuracy: 0.8232 - loss: 0.4151 - val_accuracy: 0.7656 - val_loss: 0.5902
Epoch 300/600
148/148 1s 7ms/step - accuracy: 0.8247 - loss: 0.4103 - val_accuracy: 0.7666 - val_loss: 0.5775
Epoch 301/600
148/148 1s 8ms/step - accuracy: 0.8236 - loss: 0.4118 - val_accuracy: 0.7672 - val_loss: 0.5709
Epoch 302/600
148/148 1s 7ms/step - accuracy: 0.8288 - loss: 0.4060 - val_accuracy: 0.7665 - val_loss: 0.5753
Epoch 303/600
148/148 1s 7ms/step - accuracy: 0.8264 - loss: 0.4091 - val_accuracy: 0.7664 - val_loss: 0.5841
Epoch 304/600
148/148 1s 7ms/step - accuracy: 0.8239 - loss: 0.4149 - val_accuracy: 0.7654 - val_loss: 0.5724

Epoch 305/600
148/148 1s 8ms/step - accuracy: 0.8240 - loss: 0.4112 - val_accuracy: 0.7640 - val_loss: 0.5786
Epoch 306/600
148/148 1s 8ms/step - accuracy: 0.8225 - loss: 0.4163 - val_accuracy: 0.7676 - val_loss: 0.5728
Epoch 307/600
148/148 1s 8ms/step - accuracy: 0.8250 - loss: 0.4136 - val_accuracy: 0.7669 - val_loss: 0.5757
Epoch 308/600
148/148 1s 8ms/step - accuracy: 0.8248 - loss: 0.4112 - val_accuracy: 0.7661 - val_loss: 0.5868
Epoch 309/600
148/148 1s 10ms/step - accuracy: 0.8287 - loss: 0.4015 - val_accuracy: 0.7643 - val_loss: 0.5803
Epoch 310/600
148/148 2s 8ms/step - accuracy: 0.8284 - loss: 0.4040 - val_accuracy: 0.7664 - val_loss: 0.5835
Epoch 311/600
148/148 1s 8ms/step - accuracy: 0.8291 - loss: 0.4038 - val_accuracy: 0.7678 - val_loss: 0.5759
Epoch 312/600
148/148 1s 8ms/step - accuracy: 0.8260 - loss: 0.4095 - val_accuracy: 0.7659 - val_loss: 0.5784
Epoch 313/600
148/148 1s 8ms/step - accuracy: 0.8259 - loss: 0.4095 - val_accuracy: 0.7634 - val_loss: 0.5890
Epoch 314/600
148/148 1s 7ms/step - accuracy: 0.8250 - loss: 0.4094 - val_accuracy: 0.7669 - val_loss: 0.5753
Epoch 315/600
148/148 1s 7ms/step - accuracy: 0.8268 - loss: 0.4058 - val_accuracy: 0.7640 - val_loss: 0.5800
Epoch 316/600
148/148 1s 7ms/step - accuracy: 0.8262 - loss: 0.4086 - val_accuracy: 0.7638 - val_loss: 0.5831
Epoch 317/600
148/148 1s 7ms/step - accuracy: 0.8259 - loss: 0.4071 - val_accuracy: 0.7649 - val_loss: 0.5754
Epoch 318/600
148/148 1s 7ms/step - accuracy: 0.8238 - loss: 0.4142 - val_accuracy: 0.7664 - val_loss: 0.5844
Epoch 319/600
148/148 1s 7ms/step - accuracy: 0.8239 - loss: 0.4098 - val_accuracy: 0.7665 - val_loss: 0.5800
Epoch 320/600
148/148 1s 7ms/step - accuracy: 0.8273 - loss: 0.4072 - val_accuracy: 0.7629 - val_loss: 0.5734
Epoch 321/600
148/148 1s 7ms/step - accuracy: 0.8281 - loss: 0.4052 - val_accuracy: 0.7654 - val_loss: 0.5754
Epoch 322/600
148/148 1s 8ms/step - accuracy: 0.8233 - loss: 0.4080 - val_accuracy: 0.7643 - val_loss: 0.5775
Epoch 323/600
148/148 1s 8ms/step - accuracy: 0.8274 - loss: 0.4051 - val_accuracy: 0.7672 - val_loss: 0.5795
Epoch 324/600
148/148 1s 7ms/step - accuracy: 0.8265 - loss: 0.4108 - val_accuracy: 0.7672 - val_loss: 0.5781
Epoch 325/600
148/148 1s 7ms/step - accuracy: 0.8280 - loss: 0.4047 - val_accuracy: 0.7645 - val_loss: 0.5779
Epoch 326/600
148/148 1s 7ms/step - accuracy: 0.8274 - loss: 0.4091 - val_accuracy: 0.7660 - val_loss: 0.5766
Epoch 327/600
148/148 1s 7ms/step - accuracy: 0.8276 - loss: 0.4044 - val_accuracy: 0.7659 - val_loss: 0.5734
Epoch 328/600
148/148 1s 7ms/step - accuracy: 0.8259 - loss: 0.4058 - val_accuracy: 0.7674 - val_loss: 0.5878
Epoch 329/600
148/148 1s 7ms/step - accuracy: 0.8270 - loss: 0.4039 - val_accuracy: 0.7657 - val_loss: 0.5896
Epoch 330/600

```
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.8274 - loss: 0.4068 - val_accuracy: 0.7678 - val_loss: 0.5797
Epoch 331/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.8258 - loss: 0.4071 - val_accuracy: 0.7666 - val_loss: 0.5755
Epoch 332/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.8287 - loss: 0.4036 - val_accuracy: 0.7662 - val_loss: 0.5803
Epoch 333/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.8269 - loss: 0.4099 - val_accuracy: 0.7646 - val_loss: 0.5766
Epoch 334/600
148/148 ━━━━━━━━ 1s 8ms/step - accuracy: 0.8239 - loss: 0.4077 - val_accuracy: 0.7657 - val_loss: 0.5787
Epoch 335/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.8293 - loss: 0.4000 - val_accuracy: 0.7663 - val_loss: 0.5879
Epoch 336/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.8256 - loss: 0.4054 - val_accuracy: 0.7635 - val_loss: 0.5885
Epoch 337/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.8299 - loss: 0.4039 - val_accuracy: 0.7680 - val_loss: 0.5748
```

In [258]: ann4_smote.evaluate(X_train_smote, y_train_smote)

```
3770/3770 ━━━━━━━━ 4s 964us/step - accuracy: 0.8971 - loss: 0.2805
```

Out[258]: [0.2933151125907898, 0.9028409719467163]

In [259]: ann4_smote.summary()

Model: "sequential_8"

Layer (type)	Output Shape	Param #
dense_50 (Dense)	(None, 512)	23,040
dropout_42 (Dropout)	(None, 512)	0
dense_51 (Dense)	(None, 256)	131,328
dropout_43 (Dropout)	(None, 256)	0
dense_52 (Dense)	(None, 256)	65,792
dropout_44 (Dropout)	(None, 256)	0
dense_53 (Dense)	(None, 128)	32,896
dropout_45 (Dropout)	(None, 128)	0
dense_54 (Dense)	(None, 128)	16,512
dropout_46 (Dropout)	(None, 128)	0
dense_55 (Dense)	(None, 64)	8,256
dropout_47 (Dropout)	(None, 64)	0
dense_56 (Dense)	(None, 3)	195

Total params: 834,059 (3.18 MB)

Trainable params: 278,019 (1.06 MB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 556,040 (2.12 MB)

In [260]: eval_metric(ann4_smote, X_train_smote, y_train_smote, X_val, y_val)

3770/3770 ————— 4s 1ms/step
 591/591 ————— 1s 970us/step

Test Set:

```
[[4421 966 121]
 [1383 7716 954]
 [ 84 862 2396]]
```

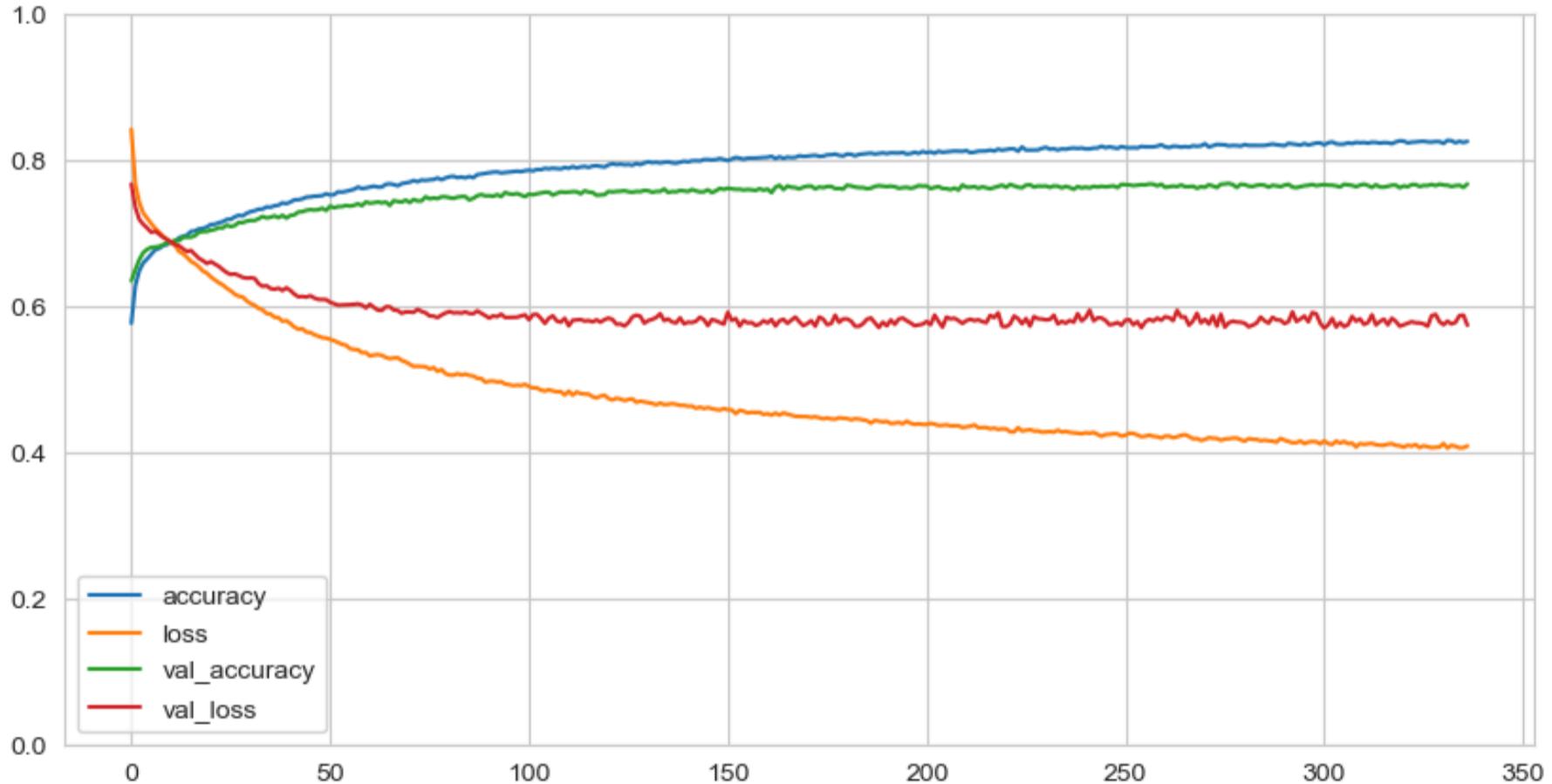
	precision	recall	f1-score	support
0	0.75	0.80	0.78	5508
1	0.81	0.77	0.79	10053
2	0.69	0.72	0.70	3342
accuracy			0.77	18903
macro avg	0.75	0.76	0.76	18903
weighted avg	0.77	0.77	0.77	18903

Train Set:

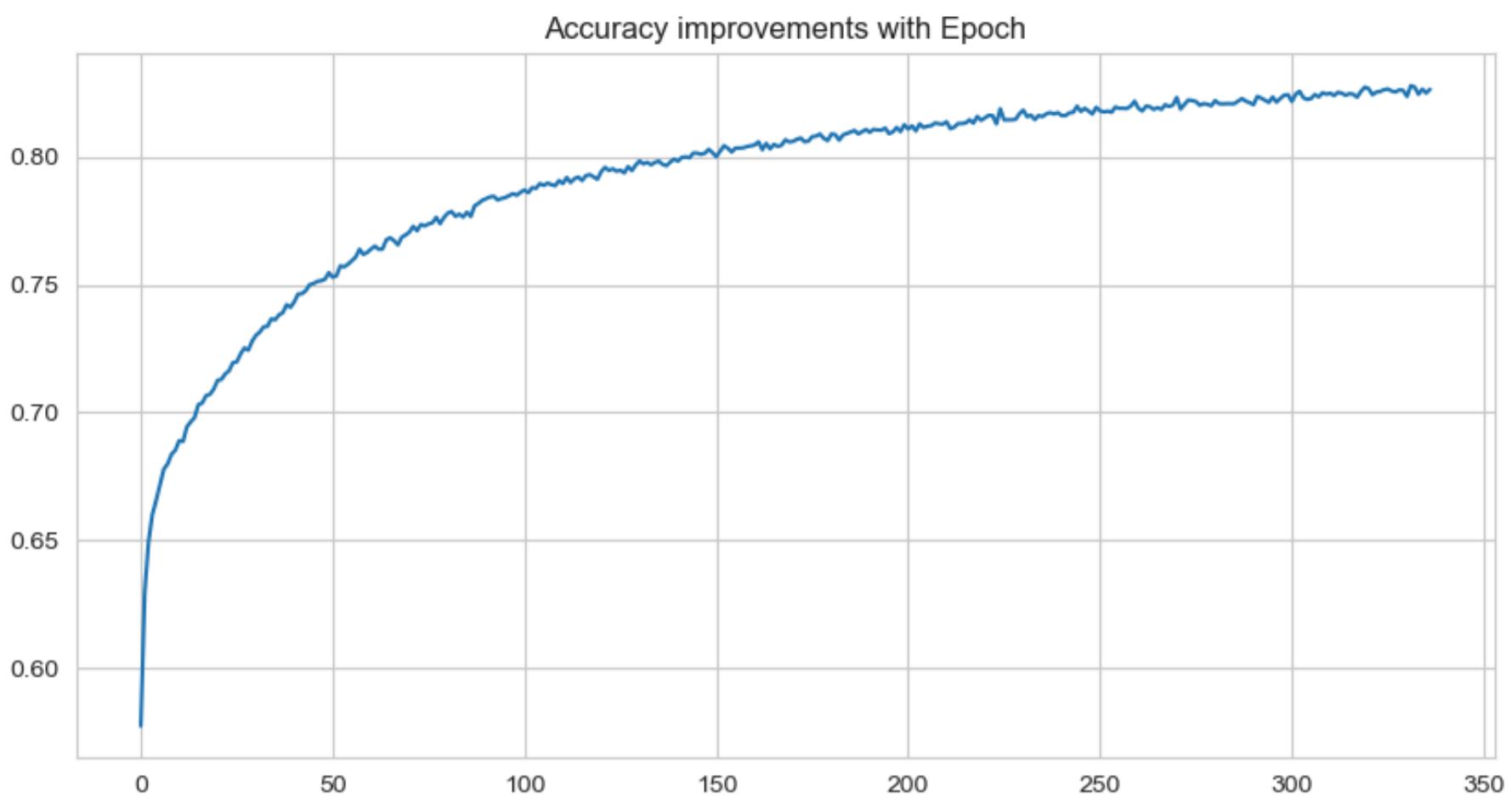
```
[[37392 2561 256]
 [ 2933 35067 2209]
 [ 271 3490 36448]]
```

	precision	recall	f1-score	support
0	0.92	0.93	0.93	40209
1	0.85	0.87	0.86	40209
2	0.94	0.91	0.92	40209
accuracy			0.90	120627
macro avg	0.90	0.90	0.90	120627
weighted avg	0.90	0.90	0.90	120627

```
In [261]: pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



```
In [262]: pd.DataFrame(history.history)[“accuracy”].plot(figsize=(10, 5))
plt.title(“Accuracy improvements with Epoch”)
plt.show()
```



```
In [ ]: # Save the Model

from tensorflow.keras.models import load_model

# Save the model to 'model.h5' file
ann4_smote.save('ann4_smote_model.h5')

# To Load the model Later for further use
loaded_ann4_smote = load_model('ann4_smote_model.h5')
```

ANN-4 Model with SMOTE Summary:

- **(Dense) layers:** 7 / **Neurons:** 512-256-256-128-128-64 / **Dropout:** 30-30-25-25-20-20% / **Learning Rate:** 0.001 / **Batch Size:** 512 / **Epochs:** 600 / **Early Stop (val_accuracy):** 60
- **Accuracy:** 0.9028 / **Val_Accuracy:** 0.7700 / **Loss:** 0.2933 / **Val_Loss:** 0.7300 / **Train Recall (Class 2):** 0.91 / **Test Recall (Class 2):** 0.72

Improvements: The model shows strong training performance, with accuracy reaching 90.28% and excellent recall for class 2 at 91%. The inclusion of SMOTE appears to have positively influenced the model's ability to identify the minority class effectively during training.

No Improvement: Validation accuracy has not improved significantly, indicating ongoing challenges with model generalization to new data. This suggests the need for further adjustments to enhance the model's robustness against unseen data.

Got Worse: Validation loss remains high, consistent with the possibility of overfitting. This indicates that while the model is performing well on training data, it may be too tailored to the specific patterns of the training set, leading to reduced performance on the validation set. Future iterations might focus on further optimizing dropout rates or experimenting with different layer configurations to address this issue.

ANN-5 Model +BatchNormalization(%90)

```
In [256...]: # ANN-5 Model
# BatchNormalization() is added!

# 1) Model Architecture:
# Fully connected (Dense) --> 7 Layers
ann5 = Sequential([
    Dense(512, input_dim=X_train.shape[1], activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(256, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
```

```
Dense(256, activation='relu'),
BatchNormalization(),
Dropout(0.25),
Dense(128, activation='relu'),
BatchNormalization(),
Dropout(0.25),
Dense(128, activation='relu'),
BatchNormalization(),
Dropout(0.2),
Dense(64, activation='relu'),
BatchNormalization(),
Dropout(0.2),
Dense(3, activation='softmax')
])

# 2) Compiling the Model:
ann5.compile(optimizer=Adam(learning_rate=0.001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 3) Early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                                patience=60,
                                mode="auto",
                                restore_best_weights=True)

# 4) Train the model
history = ann5.fit(
    x=X_train,
    y=y_train,
    validation_data=(X_val, y_val),
    batch_size=512,
    epochs=600,
    verbose=1,
    callbacks=[early_stopping])
```

Epoch 1/600
148/148 8s 14ms/step - accuracy: 0.4477 - loss: 1.2138 - val_accuracy: 0.5322 - val_loss: 0.9386
Epoch 2/600
148/148 2s 12ms/step - accuracy: 0.5832 - loss: 0.8568 - val_accuracy: 0.5732 - val_loss: 0.8475
Epoch 3/600
148/148 2s 12ms/step - accuracy: 0.6162 - loss: 0.7976 - val_accuracy: 0.6411 - val_loss: 0.7437
Epoch 4/600
148/148 2s 12ms/step - accuracy: 0.6401 - loss: 0.7591 - val_accuracy: 0.6654 - val_loss: 0.7255
Epoch 5/600
148/148 2s 11ms/step - accuracy: 0.6466 - loss: 0.7444 - val_accuracy: 0.6729 - val_loss: 0.7177
Epoch 6/600
148/148 2s 11ms/step - accuracy: 0.6521 - loss: 0.7368 - val_accuracy: 0.6757 - val_loss: 0.7146
Epoch 7/600
148/148 2s 11ms/step - accuracy: 0.6638 - loss: 0.7234 - val_accuracy: 0.6784 - val_loss: 0.7086
Epoch 8/600
148/148 2s 14ms/step - accuracy: 0.6644 - loss: 0.7229 - val_accuracy: 0.6805 - val_loss: 0.7074
Epoch 9/600
148/148 2s 11ms/step - accuracy: 0.6705 - loss: 0.7167 - val_accuracy: 0.6799 - val_loss: 0.7053
Epoch 10/600
148/148 2s 11ms/step - accuracy: 0.6692 - loss: 0.7141 - val_accuracy: 0.6816 - val_loss: 0.7023
Epoch 11/600
148/148 2s 11ms/step - accuracy: 0.6747 - loss: 0.7082 - val_accuracy: 0.6836 - val_loss: 0.7004
Epoch 12/600
148/148 2s 10ms/step - accuracy: 0.6800 - loss: 0.7027 - val_accuracy: 0.6843 - val_loss: 0.6968
Epoch 13/600
148/148 2s 10ms/step - accuracy: 0.6819 - loss: 0.6977 - val_accuracy: 0.6879 - val_loss: 0.6946
Epoch 14/600
148/148 2s 11ms/step - accuracy: 0.6839 - loss: 0.6918 - val_accuracy: 0.6905 - val_loss: 0.6931
Epoch 15/600
148/148 2s 10ms/step - accuracy: 0.6872 - loss: 0.6874 - val_accuracy: 0.6888 - val_loss: 0.6951
Epoch 16/600
148/148 2s 11ms/step - accuracy: 0.6907 - loss: 0.6868 - val_accuracy: 0.6899 - val_loss: 0.6928
Epoch 17/600
148/148 2s 11ms/step - accuracy: 0.6896 - loss: 0.6872 - val_accuracy: 0.6916 - val_loss: 0.6883
Epoch 18/600
148/148 2s 13ms/step - accuracy: 0.6951 - loss: 0.6762 - val_accuracy: 0.6896 - val_loss: 0.6875
Epoch 19/600
148/148 2s 12ms/step - accuracy: 0.6908 - loss: 0.6785 - val_accuracy: 0.6897 - val_loss: 0.6879
Epoch 20/600
148/148 2s 11ms/step - accuracy: 0.6970 - loss: 0.6721 - val_accuracy: 0.6942 - val_loss: 0.6825
Epoch 21/600
148/148 2s 12ms/step - accuracy: 0.6981 - loss: 0.6702 - val_accuracy: 0.6959 - val_loss: 0.6822
Epoch 22/600
148/148 2s 13ms/step - accuracy: 0.7012 - loss: 0.6684 - val_accuracy: 0.6973 - val_loss: 0.6801
Epoch 23/600
148/148 2s 11ms/step - accuracy: 0.7010 - loss: 0.6646 - val_accuracy: 0.6968 - val_loss: 0.6774
Epoch 24/600
148/148 2s 12ms/step - accuracy: 0.7044 - loss: 0.6590 - val_accuracy: 0.6994 - val_loss: 0.6735
Epoch 25/600
148/148 2s 11ms/step - accuracy: 0.7040 - loss: 0.6575 - val_accuracy: 0.7001 - val_loss: 0.6772
Epoch 26/600

148/148 2s 11ms/step - accuracy: 0.7084 - loss: 0.6556 - val_accuracy: 0.7002 - val_loss: 0.6772
Epoch 27/600
148/148 2s 12ms/step - accuracy: 0.7074 - loss: 0.6538 - val_accuracy: 0.6996 - val_loss: 0.6715
Epoch 28/600
148/148 2s 13ms/step - accuracy: 0.7140 - loss: 0.6431 - val_accuracy: 0.7041 - val_loss: 0.6671
Epoch 29/600
148/148 2s 11ms/step - accuracy: 0.7145 - loss: 0.6422 - val_accuracy: 0.7062 - val_loss: 0.6686
Epoch 30/600
148/148 2s 12ms/step - accuracy: 0.7113 - loss: 0.6435 - val_accuracy: 0.7067 - val_loss: 0.6615
Epoch 31/600
148/148 2s 12ms/step - accuracy: 0.7190 - loss: 0.6319 - val_accuracy: 0.7066 - val_loss: 0.6630
Epoch 32/600
148/148 2s 13ms/step - accuracy: 0.7206 - loss: 0.6275 - val_accuracy: 0.7062 - val_loss: 0.6618
Epoch 33/600
148/148 2s 12ms/step - accuracy: 0.7209 - loss: 0.6287 - val_accuracy: 0.7099 - val_loss: 0.6551
Epoch 34/600
148/148 2s 11ms/step - accuracy: 0.7212 - loss: 0.6252 - val_accuracy: 0.7132 - val_loss: 0.6511
Epoch 35/600
148/148 2s 11ms/step - accuracy: 0.7216 - loss: 0.6235 - val_accuracy: 0.7146 - val_loss: 0.6519
Epoch 36/600
148/148 2s 11ms/step - accuracy: 0.7254 - loss: 0.6167 - val_accuracy: 0.7146 - val_loss: 0.6480
Epoch 37/600
148/148 2s 12ms/step - accuracy: 0.7292 - loss: 0.6150 - val_accuracy: 0.7120 - val_loss: 0.6501
Epoch 38/600
148/148 2s 14ms/step - accuracy: 0.7340 - loss: 0.6090 - val_accuracy: 0.7203 - val_loss: 0.6424
Epoch 39/600
148/148 2s 12ms/step - accuracy: 0.7317 - loss: 0.6059 - val_accuracy: 0.7168 - val_loss: 0.6432
Epoch 40/600
148/148 2s 13ms/step - accuracy: 0.7314 - loss: 0.6031 - val_accuracy: 0.7210 - val_loss: 0.6403
Epoch 41/600
148/148 2s 11ms/step - accuracy: 0.7330 - loss: 0.6051 - val_accuracy: 0.7210 - val_loss: 0.6420
Epoch 42/600
148/148 2s 10ms/step - accuracy: 0.7396 - loss: 0.5947 - val_accuracy: 0.7233 - val_loss: 0.6348
Epoch 43/600
148/148 2s 10ms/step - accuracy: 0.7358 - loss: 0.5972 - val_accuracy: 0.7240 - val_loss: 0.6325
Epoch 44/600
148/148 2s 10ms/step - accuracy: 0.7376 - loss: 0.5927 - val_accuracy: 0.7262 - val_loss: 0.6339
Epoch 45/600
148/148 2s 13ms/step - accuracy: 0.7381 - loss: 0.5913 - val_accuracy: 0.7228 - val_loss: 0.6327
Epoch 46/600
148/148 2s 11ms/step - accuracy: 0.7409 - loss: 0.5864 - val_accuracy: 0.7272 - val_loss: 0.6368
Epoch 47/600
148/148 2s 13ms/step - accuracy: 0.7405 - loss: 0.5819 - val_accuracy: 0.7273 - val_loss: 0.6298
Epoch 48/600
148/148 2s 11ms/step - accuracy: 0.7447 - loss: 0.5799 - val_accuracy: 0.7298 - val_loss: 0.6296
Epoch 49/600
148/148 2s 11ms/step - accuracy: 0.7479 - loss: 0.5765 - val_accuracy: 0.7309 - val_loss: 0.6246
Epoch 50/600
148/148 2s 12ms/step - accuracy: 0.7477 - loss: 0.5766 - val_accuracy: 0.7327 - val_loss: 0.6224
Epoch 51/600
148/148 2s 12ms/step - accuracy: 0.7500 - loss: 0.5718 - val_accuracy: 0.7348 - val_loss:

s: 0.6293
Epoch 52/600
148/148 2s 11ms/step - accuracy: 0.7461 - loss: 0.5701 - val_accuracy: 0.7319 - val_loss: 0.5970
s: 0.6219
Epoch 53/600
148/148 2s 12ms/step - accuracy: 0.7517 - loss: 0.5655 - val_accuracy: 0.7322 - val_loss: 0.5970
s: 0.6212
Epoch 54/600
148/148 2s 11ms/step - accuracy: 0.7546 - loss: 0.5621 - val_accuracy: 0.7345 - val_loss: 0.5970
s: 0.6151
Epoch 55/600
148/148 2s 12ms/step - accuracy: 0.7572 - loss: 0.5577 - val_accuracy: 0.7348 - val_loss: 0.5970
s: 0.6132
Epoch 56/600
148/148 2s 14ms/step - accuracy: 0.7547 - loss: 0.5609 - val_accuracy: 0.7379 - val_loss: 0.5970
s: 0.6108
Epoch 57/600
148/148 2s 12ms/step - accuracy: 0.7568 - loss: 0.5603 - val_accuracy: 0.7381 - val_loss: 0.5970
s: 0.6153
Epoch 58/600
148/148 2s 12ms/step - accuracy: 0.7565 - loss: 0.5557 - val_accuracy: 0.7371 - val_loss: 0.5970
s: 0.6123
Epoch 59/600
148/148 2s 11ms/step - accuracy: 0.7591 - loss: 0.5517 - val_accuracy: 0.7413 - val_loss: 0.5970
s: 0.6118
Epoch 60/600
148/148 2s 12ms/step - accuracy: 0.7591 - loss: 0.5526 - val_accuracy: 0.7398 - val_loss: 0.5970
s: 0.6143
Epoch 61/600
148/148 2s 11ms/step - accuracy: 0.7595 - loss: 0.5508 - val_accuracy: 0.7417 - val_loss: 0.5970
s: 0.6107
Epoch 62/600
148/148 2s 11ms/step - accuracy: 0.7588 - loss: 0.5477 - val_accuracy: 0.7422 - val_loss: 0.5970
s: 0.6054
Epoch 63/600
148/148 2s 12ms/step - accuracy: 0.7633 - loss: 0.5408 - val_accuracy: 0.7398 - val_loss: 0.5970
s: 0.6094
Epoch 64/600
148/148 2s 13ms/step - accuracy: 0.7611 - loss: 0.5455 - val_accuracy: 0.7396 - val_loss: 0.5970
s: 0.6057
Epoch 65/600
148/148 2s 12ms/step - accuracy: 0.7606 - loss: 0.5458 - val_accuracy: 0.7458 - val_loss: 0.5970
s: 0.6025
Epoch 66/600
148/148 2s 14ms/step - accuracy: 0.7660 - loss: 0.5357 - val_accuracy: 0.7441 - val_loss: 0.5970
s: 0.6003
Epoch 67/600
148/148 2s 12ms/step - accuracy: 0.7634 - loss: 0.5399 - val_accuracy: 0.7450 - val_loss: 0.5970
s: 0.6041
Epoch 68/600
148/148 2s 12ms/step - accuracy: 0.7658 - loss: 0.5390 - val_accuracy: 0.7450 - val_loss: 0.5970
s: 0.6057
Epoch 69/600
148/148 2s 11ms/step - accuracy: 0.7679 - loss: 0.5341 - val_accuracy: 0.7486 - val_loss: 0.5970
s: 0.6025
Epoch 70/600
148/148 2s 11ms/step - accuracy: 0.7696 - loss: 0.5267 - val_accuracy: 0.7488 - val_loss: 0.5958
s: 0.5958
Epoch 71/600
148/148 2s 11ms/step - accuracy: 0.7692 - loss: 0.5292 - val_accuracy: 0.7468 - val_loss: 0.5984
s: 0.5984
Epoch 72/600
148/148 2s 12ms/step - accuracy: 0.7652 - loss: 0.5333 - val_accuracy: 0.7468 - val_loss: 0.5971
s: 0.5971
Epoch 73/600
148/148 2s 12ms/step - accuracy: 0.7737 - loss: 0.5265 - val_accuracy: 0.7489 - val_loss: 0.6056
s: 0.6056
Epoch 74/600
148/148 2s 11ms/step - accuracy: 0.7708 - loss: 0.5270 - val_accuracy: 0.7512 - val_loss: 0.5970
s: 0.5970
Epoch 75/600
148/148 2s 11ms/step - accuracy: 0.7738 - loss: 0.5239 - val_accuracy: 0.7471 - val_loss: 0.5921
s: 0.5921
Epoch 76/600
148/148 2s 13ms/step - accuracy: 0.7719 - loss: 0.5215 - val_accuracy: 0.7500 - val_loss: 0.5988
s: 0.5988

Epoch 77/600
148/148 2s 11ms/step - accuracy: 0.7726 - loss: 0.5193 - val_accuracy: 0.7495 - val_loss: 0.5972
Epoch 78/600
148/148 2s 10ms/step - accuracy: 0.7701 - loss: 0.5209 - val_accuracy: 0.7494 - val_loss: 0.5958
Epoch 79/600
148/148 2s 11ms/step - accuracy: 0.7752 - loss: 0.5162 - val_accuracy: 0.7505 - val_loss: 0.5884
Epoch 80/600
148/148 3s 11ms/step - accuracy: 0.7750 - loss: 0.5186 - val_accuracy: 0.7518 - val_loss: 0.6011
Epoch 81/600
148/148 2s 12ms/step - accuracy: 0.7765 - loss: 0.5159 - val_accuracy: 0.7545 - val_loss: 0.5932
Epoch 82/600
148/148 2s 12ms/step - accuracy: 0.7778 - loss: 0.5140 - val_accuracy: 0.7529 - val_loss: 0.5938
Epoch 83/600
148/148 2s 12ms/step - accuracy: 0.7765 - loss: 0.5124 - val_accuracy: 0.7519 - val_loss: 0.5922
Epoch 84/600
148/148 2s 11ms/step - accuracy: 0.7787 - loss: 0.5100 - val_accuracy: 0.7542 - val_loss: 0.5991
Epoch 85/600
148/148 2s 11ms/step - accuracy: 0.7772 - loss: 0.5093 - val_accuracy: 0.7561 - val_loss: 0.5984
Epoch 86/600
148/148 2s 11ms/step - accuracy: 0.7794 - loss: 0.5096 - val_accuracy: 0.7512 - val_loss: 0.5985
Epoch 87/600
148/148 2s 11ms/step - accuracy: 0.7813 - loss: 0.5055 - val_accuracy: 0.7556 - val_loss: 0.5892
Epoch 88/600
148/148 2s 11ms/step - accuracy: 0.7811 - loss: 0.5074 - val_accuracy: 0.7546 - val_loss: 0.5962
Epoch 89/600
148/148 2s 11ms/step - accuracy: 0.7799 - loss: 0.5121 - val_accuracy: 0.7558 - val_loss: 0.5942
Epoch 90/600
148/148 2s 12ms/step - accuracy: 0.7772 - loss: 0.5127 - val_accuracy: 0.7536 - val_loss: 0.6033
Epoch 91/600
148/148 2s 13ms/step - accuracy: 0.7814 - loss: 0.5054 - val_accuracy: 0.7542 - val_loss: 0.5917
Epoch 92/600
148/148 2s 12ms/step - accuracy: 0.7848 - loss: 0.4989 - val_accuracy: 0.7581 - val_loss: 0.5905
Epoch 93/600
148/148 2s 12ms/step - accuracy: 0.7850 - loss: 0.4980 - val_accuracy: 0.7546 - val_loss: 0.5911
Epoch 94/600
148/148 2s 11ms/step - accuracy: 0.7812 - loss: 0.5034 - val_accuracy: 0.7572 - val_loss: 0.5879
Epoch 95/600
148/148 2s 12ms/step - accuracy: 0.7825 - loss: 0.4995 - val_accuracy: 0.7579 - val_loss: 0.5898
Epoch 96/600
148/148 2s 12ms/step - accuracy: 0.7856 - loss: 0.4964 - val_accuracy: 0.7568 - val_loss: 0.5926
Epoch 97/600
148/148 2s 12ms/step - accuracy: 0.7905 - loss: 0.4908 - val_accuracy: 0.7568 - val_loss: 0.5954
Epoch 98/600
148/148 2s 11ms/step - accuracy: 0.7831 - loss: 0.4968 - val_accuracy: 0.7552 - val_loss: 0.5909
Epoch 99/600
148/148 2s 12ms/step - accuracy: 0.7860 - loss: 0.4969 - val_accuracy: 0.7604 - val_loss: 0.5898
Epoch 100/600
148/148 2s 11ms/step - accuracy: 0.7874 - loss: 0.4906 - val_accuracy: 0.7614 - val_loss: 0.5879
Epoch 101/600
148/148 2s 11ms/step - accuracy: 0.7875 - loss: 0.4912 - val_accuracy: 0.7619 - val_loss: 0.5847
Epoch 102/600

148/148 2s 11ms/step - accuracy: 0.7859 - loss: 0.4925 - val_accuracy: 0.7604 - val_loss: 0.5878
Epoch 103/600
148/148 2s 11ms/step - accuracy: 0.7864 - loss: 0.4910 - val_accuracy: 0.7573 - val_loss: 0.5853
Epoch 104/600
148/148 2s 11ms/step - accuracy: 0.7873 - loss: 0.4904 - val_accuracy: 0.7606 - val_loss: 0.5906
Epoch 105/600
148/148 2s 11ms/step - accuracy: 0.7921 - loss: 0.4805 - val_accuracy: 0.7590 - val_loss: 0.5839
Epoch 106/600
148/148 2s 10ms/step - accuracy: 0.7874 - loss: 0.4920 - val_accuracy: 0.7606 - val_loss: 0.5924
Epoch 107/600
148/148 2s 12ms/step - accuracy: 0.7880 - loss: 0.4884 - val_accuracy: 0.7644 - val_loss: 0.5867
Epoch 108/600
148/148 2s 12ms/step - accuracy: 0.7897 - loss: 0.4860 - val_accuracy: 0.7630 - val_loss: 0.5886
Epoch 109/600
148/148 2s 12ms/step - accuracy: 0.7896 - loss: 0.4852 - val_accuracy: 0.7619 - val_loss: 0.5896
Epoch 110/600
148/148 2s 13ms/step - accuracy: 0.7911 - loss: 0.4864 - val_accuracy: 0.7623 - val_loss: 0.5812
Epoch 111/600
148/148 2s 11ms/step - accuracy: 0.7906 - loss: 0.4836 - val_accuracy: 0.7634 - val_loss: 0.5824
Epoch 112/600
148/148 2s 11ms/step - accuracy: 0.7883 - loss: 0.4865 - val_accuracy: 0.7607 - val_loss: 0.5881
Epoch 113/600
148/148 2s 10ms/step - accuracy: 0.7913 - loss: 0.4836 - val_accuracy: 0.7625 - val_loss: 0.5806
Epoch 114/600
148/148 2s 11ms/step - accuracy: 0.7964 - loss: 0.4807 - val_accuracy: 0.7641 - val_loss: 0.5805
Epoch 115/600
148/148 2s 10ms/step - accuracy: 0.7917 - loss: 0.4839 - val_accuracy: 0.7586 - val_loss: 0.5914
Epoch 116/600
148/148 2s 11ms/step - accuracy: 0.7912 - loss: 0.4828 - val_accuracy: 0.7622 - val_loss: 0.5851
Epoch 117/600
148/148 2s 11ms/step - accuracy: 0.7927 - loss: 0.4779 - val_accuracy: 0.7633 - val_loss: 0.5859
Epoch 118/600
148/148 2s 12ms/step - accuracy: 0.7957 - loss: 0.4766 - val_accuracy: 0.7628 - val_loss: 0.5867
Epoch 119/600
148/148 2s 12ms/step - accuracy: 0.7928 - loss: 0.4782 - val_accuracy: 0.7625 - val_loss: 0.5846
Epoch 120/600
148/148 2s 12ms/step - accuracy: 0.7945 - loss: 0.4751 - val_accuracy: 0.7642 - val_loss: 0.5825
Epoch 121/600
148/148 2s 13ms/step - accuracy: 0.7971 - loss: 0.4741 - val_accuracy: 0.7622 - val_loss: 0.5840
Epoch 122/600
148/148 2s 12ms/step - accuracy: 0.7981 - loss: 0.4708 - val_accuracy: 0.7622 - val_loss: 0.5793
Epoch 123/600
148/148 2s 13ms/step - accuracy: 0.7964 - loss: 0.4773 - val_accuracy: 0.7622 - val_loss: 0.5862
Epoch 124/600
148/148 2s 11ms/step - accuracy: 0.7938 - loss: 0.4756 - val_accuracy: 0.7653 - val_loss: 0.5811
Epoch 125/600
148/148 2s 11ms/step - accuracy: 0.7947 - loss: 0.4759 - val_accuracy: 0.7655 - val_loss: 0.5833
Epoch 126/600
148/148 2s 12ms/step - accuracy: 0.8000 - loss: 0.4694 - val_accuracy: 0.7663 - val_loss: 0.5821
Epoch 127/600
148/148 2s 11ms/step - accuracy: 0.7999 - loss: 0.4673 - val_accuracy: 0.7668 - val_loss:

s: 0.5761
Epoch 128/600
148/148 2s 13ms/step - accuracy: 0.7980 - loss: 0.4691 - val_accuracy: 0.7658 - val_loss: 0.5779
Epoch 129/600
148/148 2s 12ms/step - accuracy: 0.7987 - loss: 0.4710 - val_accuracy: 0.7653 - val_loss: 0.5816
Epoch 130/600
148/148 2s 11ms/step - accuracy: 0.7985 - loss: 0.4694 - val_accuracy: 0.7646 - val_loss: 0.5845
Epoch 131/600
148/148 2s 13ms/step - accuracy: 0.7972 - loss: 0.4711 - val_accuracy: 0.7671 - val_loss: 0.5822
Epoch 132/600
148/148 2s 11ms/step - accuracy: 0.7970 - loss: 0.4691 - val_accuracy: 0.7668 - val_loss: 0.5773
Epoch 133/600
148/148 2s 11ms/step - accuracy: 0.7961 - loss: 0.4706 - val_accuracy: 0.7664 - val_loss: 0.5855
Epoch 134/600
148/148 2s 11ms/step - accuracy: 0.8015 - loss: 0.4642 - val_accuracy: 0.7638 - val_loss: 0.5909
Epoch 135/600
148/148 2s 11ms/step - accuracy: 0.8023 - loss: 0.4634 - val_accuracy: 0.7623 - val_loss: 0.5811
Epoch 136/600
148/148 2s 11ms/step - accuracy: 0.8022 - loss: 0.4642 - val_accuracy: 0.7663 - val_loss: 0.5788
Epoch 137/600
148/148 2s 12ms/step - accuracy: 0.7983 - loss: 0.4679 - val_accuracy: 0.7665 - val_loss: 0.5757
Epoch 138/600
148/148 2s 13ms/step - accuracy: 0.7990 - loss: 0.4663 - val_accuracy: 0.7690 - val_loss: 0.5837
Epoch 139/600
148/148 2s 12ms/step - accuracy: 0.8012 - loss: 0.4656 - val_accuracy: 0.7663 - val_loss: 0.5727
Epoch 140/600
148/148 2s 11ms/step - accuracy: 0.8048 - loss: 0.4601 - val_accuracy: 0.7663 - val_loss: 0.5782
Epoch 141/600
148/148 2s 11ms/step - accuracy: 0.7974 - loss: 0.4691 - val_accuracy: 0.7677 - val_loss: 0.5830
Epoch 142/600
148/148 2s 12ms/step - accuracy: 0.8028 - loss: 0.4640 - val_accuracy: 0.7665 - val_loss: 0.5843
Epoch 143/600
148/148 2s 12ms/step - accuracy: 0.8029 - loss: 0.4573 - val_accuracy: 0.7683 - val_loss: 0.5870
Epoch 144/600
148/148 2s 11ms/step - accuracy: 0.8015 - loss: 0.4633 - val_accuracy: 0.7683 - val_loss: 0.5827
Epoch 145/600
148/148 2s 12ms/step - accuracy: 0.8025 - loss: 0.4621 - val_accuracy: 0.7670 - val_loss: 0.5917
Epoch 146/600
148/148 2s 12ms/step - accuracy: 0.8041 - loss: 0.4576 - val_accuracy: 0.7668 - val_loss: 0.5806
Epoch 147/600
148/148 2s 11ms/step - accuracy: 0.8025 - loss: 0.4595 - val_accuracy: 0.7657 - val_loss: 0.5776
Epoch 148/600
148/148 2s 11ms/step - accuracy: 0.8052 - loss: 0.4568 - val_accuracy: 0.7665 - val_loss: 0.5811
Epoch 149/600
148/148 2s 12ms/step - accuracy: 0.8071 - loss: 0.4505 - val_accuracy: 0.7674 - val_loss: 0.5746
Epoch 150/600
148/148 2s 13ms/step - accuracy: 0.8056 - loss: 0.4575 - val_accuracy: 0.7675 - val_loss: 0.5851
Epoch 151/600
148/148 2s 12ms/step - accuracy: 0.8029 - loss: 0.4579 - val_accuracy: 0.7683 - val_loss: 0.5803
Epoch 152/600
148/148 2s 12ms/step - accuracy: 0.8034 - loss: 0.4588 - val_accuracy: 0.7668 - val_loss: 0.5732

Epoch 153/600
148/148 2s 11ms/step - accuracy: 0.8032 - loss: 0.4592 - val_accuracy: 0.7694 - val_loss: 0.5805
Epoch 154/600
148/148 2s 12ms/step - accuracy: 0.8028 - loss: 0.4578 - val_accuracy: 0.7676 - val_loss: 0.5841
Epoch 155/600
148/148 2s 12ms/step - accuracy: 0.8050 - loss: 0.4555 - val_accuracy: 0.7630 - val_loss: 0.5843
Epoch 156/600
148/148 2s 13ms/step - accuracy: 0.8048 - loss: 0.4580 - val_accuracy: 0.7684 - val_loss: 0.5738
Epoch 157/600
148/148 2s 12ms/step - accuracy: 0.8065 - loss: 0.4527 - val_accuracy: 0.7698 - val_loss: 0.5818
Epoch 158/600
148/148 2s 11ms/step - accuracy: 0.8062 - loss: 0.4497 - val_accuracy: 0.7678 - val_loss: 0.5781
Epoch 159/600
148/148 2s 12ms/step - accuracy: 0.8074 - loss: 0.4478 - val_accuracy: 0.7696 - val_loss: 0.5820
Epoch 160/600
148/148 2s 12ms/step - accuracy: 0.8072 - loss: 0.4511 - val_accuracy: 0.7678 - val_loss: 0.5766
Epoch 161/600
148/148 2s 11ms/step - accuracy: 0.8060 - loss: 0.4549 - val_accuracy: 0.7679 - val_loss: 0.5861
Epoch 162/600
148/148 2s 11ms/step - accuracy: 0.8021 - loss: 0.4600 - val_accuracy: 0.7690 - val_loss: 0.5826
Epoch 163/600
148/148 2s 11ms/step - accuracy: 0.8083 - loss: 0.4469 - val_accuracy: 0.7696 - val_loss: 0.5765
Epoch 164/600
148/148 2s 11ms/step - accuracy: 0.8092 - loss: 0.4537 - val_accuracy: 0.7667 - val_loss: 0.5857
Epoch 165/600
148/148 2s 11ms/step - accuracy: 0.8059 - loss: 0.4503 - val_accuracy: 0.7695 - val_loss: 0.5804
Epoch 166/600
148/148 2s 12ms/step - accuracy: 0.8085 - loss: 0.4471 - val_accuracy: 0.7673 - val_loss: 0.5850
Epoch 167/600
148/148 2s 11ms/step - accuracy: 0.8068 - loss: 0.4542 - val_accuracy: 0.7665 - val_loss: 0.5808
Epoch 168/600
148/148 2s 11ms/step - accuracy: 0.8046 - loss: 0.4534 - val_accuracy: 0.7682 - val_loss: 0.5818
Epoch 169/600
148/148 2s 11ms/step - accuracy: 0.8079 - loss: 0.4466 - val_accuracy: 0.7677 - val_loss: 0.5822
Epoch 170/600
148/148 2s 11ms/step - accuracy: 0.8058 - loss: 0.4532 - val_accuracy: 0.7702 - val_loss: 0.5723
Epoch 171/600
148/148 2s 10ms/step - accuracy: 0.8091 - loss: 0.4482 - val_accuracy: 0.7692 - val_loss: 0.5781
Epoch 172/600
148/148 2s 11ms/step - accuracy: 0.8080 - loss: 0.4484 - val_accuracy: 0.7697 - val_loss: 0.5783
Epoch 173/600
148/148 2s 10ms/step - accuracy: 0.8096 - loss: 0.4436 - val_accuracy: 0.7696 - val_loss: 0.5836
Epoch 174/600
148/148 2s 12ms/step - accuracy: 0.8079 - loss: 0.4489 - val_accuracy: 0.7714 - val_loss: 0.5822
Epoch 175/600
148/148 2s 11ms/step - accuracy: 0.8073 - loss: 0.4461 - val_accuracy: 0.7683 - val_loss: 0.5830
Epoch 176/600
148/148 2s 11ms/step - accuracy: 0.8124 - loss: 0.4440 - val_accuracy: 0.7691 - val_loss: 0.5815
Epoch 177/600
148/148 2s 10ms/step - accuracy: 0.8072 - loss: 0.4489 - val_accuracy: 0.7708 - val_loss: 0.5775
Epoch 178/600

148/148 2s 11ms/step - accuracy: 0.8080 - loss: 0.4486 - val_accuracy: 0.7700 - val_loss: 0.5848
Epoch 179/600
148/148 2s 10ms/step - accuracy: 0.8086 - loss: 0.4477 - val_accuracy: 0.7676 - val_loss: 0.5840
Epoch 180/600
148/148 2s 10ms/step - accuracy: 0.8096 - loss: 0.4452 - val_accuracy: 0.7697 - val_loss: 0.5747
Epoch 181/600
148/148 2s 10ms/step - accuracy: 0.8094 - loss: 0.4466 - val_accuracy: 0.7719 - val_loss: 0.5776
Epoch 182/600
148/148 2s 10ms/step - accuracy: 0.8093 - loss: 0.4449 - val_accuracy: 0.7707 - val_loss: 0.5853
Epoch 183/600
148/148 2s 10ms/step - accuracy: 0.8119 - loss: 0.4410 - val_accuracy: 0.7692 - val_loss: 0.5872
Epoch 184/600
148/148 2s 11ms/step - accuracy: 0.8117 - loss: 0.4389 - val_accuracy: 0.7716 - val_loss: 0.5813
Epoch 185/600
148/148 2s 11ms/step - accuracy: 0.8114 - loss: 0.4419 - val_accuracy: 0.7687 - val_loss: 0.5827
Epoch 186/600
148/148 2s 11ms/step - accuracy: 0.8106 - loss: 0.4429 - val_accuracy: 0.7703 - val_loss: 0.5847
Epoch 187/600
148/148 2s 11ms/step - accuracy: 0.8162 - loss: 0.4373 - val_accuracy: 0.7690 - val_loss: 0.5819
Epoch 188/600
148/148 2s 11ms/step - accuracy: 0.8128 - loss: 0.4405 - val_accuracy: 0.7704 - val_loss: 0.5822
Epoch 189/600
148/148 2s 11ms/step - accuracy: 0.8119 - loss: 0.4414 - val_accuracy: 0.7687 - val_loss: 0.5804
Epoch 190/600
148/148 2s 10ms/step - accuracy: 0.8160 - loss: 0.4342 - val_accuracy: 0.7713 - val_loss: 0.5818
Epoch 191/600
148/148 2s 11ms/step - accuracy: 0.8131 - loss: 0.4414 - val_accuracy: 0.7709 - val_loss: 0.5814
Epoch 192/600
148/148 2s 10ms/step - accuracy: 0.8111 - loss: 0.4405 - val_accuracy: 0.7707 - val_loss: 0.5854
Epoch 193/600
148/148 2s 10ms/step - accuracy: 0.8117 - loss: 0.4399 - val_accuracy: 0.7688 - val_loss: 0.5848
Epoch 194/600
148/148 2s 11ms/step - accuracy: 0.8092 - loss: 0.4471 - val_accuracy: 0.7725 - val_loss: 0.5839
Epoch 195/600
148/148 2s 11ms/step - accuracy: 0.8148 - loss: 0.4378 - val_accuracy: 0.7692 - val_loss: 0.5809
Epoch 196/600
148/148 2s 12ms/step - accuracy: 0.8124 - loss: 0.4393 - val_accuracy: 0.7704 - val_loss: 0.5798
Epoch 197/600
148/148 2s 10ms/step - accuracy: 0.8163 - loss: 0.4349 - val_accuracy: 0.7681 - val_loss: 0.5845
Epoch 198/600
148/148 2s 11ms/step - accuracy: 0.8157 - loss: 0.4376 - val_accuracy: 0.7697 - val_loss: 0.5850
Epoch 199/600
148/148 2s 10ms/step - accuracy: 0.8167 - loss: 0.4346 - val_accuracy: 0.7713 - val_loss: 0.5883
Epoch 200/600
148/148 2s 11ms/step - accuracy: 0.8092 - loss: 0.4418 - val_accuracy: 0.7707 - val_loss: 0.5874
Epoch 201/600
148/148 2s 10ms/step - accuracy: 0.8128 - loss: 0.4397 - val_accuracy: 0.7708 - val_loss: 0.5815
Epoch 202/600
148/148 2s 10ms/step - accuracy: 0.8158 - loss: 0.4356 - val_accuracy: 0.7711 - val_loss: 0.5771
Epoch 203/600
148/148 2s 10ms/step - accuracy: 0.8163 - loss: 0.4345 - val_accuracy: 0.7698 - val_loss:

s: 0.5773
Epoch 204/600
148/148 2s 11ms/step - accuracy: 0.8107 - loss: 0.4393 - val_accuracy: 0.7714 - val_loss: 0.5879
Epoch 205/600
148/148 2s 11ms/step - accuracy: 0.8142 - loss: 0.4382 - val_accuracy: 0.7727 - val_loss: 0.5839
Epoch 206/600
148/148 2s 11ms/step - accuracy: 0.8143 - loss: 0.4347 - val_accuracy: 0.7720 - val_loss: 0.5771
Epoch 207/600
148/148 2s 10ms/step - accuracy: 0.8133 - loss: 0.4384 - val_accuracy: 0.7690 - val_loss: 0.5862
Epoch 208/600
148/148 2s 10ms/step - accuracy: 0.8125 - loss: 0.4373 - val_accuracy: 0.7719 - val_loss: 0.5758
Epoch 209/600
148/148 2s 10ms/step - accuracy: 0.8144 - loss: 0.4349 - val_accuracy: 0.7690 - val_loss: 0.5840
Epoch 210/600
148/148 2s 10ms/step - accuracy: 0.8153 - loss: 0.4356 - val_accuracy: 0.7705 - val_loss: 0.5819
Epoch 211/600
148/148 2s 11ms/step - accuracy: 0.8148 - loss: 0.4331 - val_accuracy: 0.7725 - val_loss: 0.5875
Epoch 212/600
148/148 2s 10ms/step - accuracy: 0.8165 - loss: 0.4327 - val_accuracy: 0.7719 - val_loss: 0.5849
Epoch 213/600
148/148 2s 11ms/step - accuracy: 0.8167 - loss: 0.4302 - val_accuracy: 0.7709 - val_loss: 0.5773
Epoch 214/600
148/148 2s 11ms/step - accuracy: 0.8143 - loss: 0.4342 - val_accuracy: 0.7715 - val_loss: 0.5749
Epoch 215/600
148/148 2s 11ms/step - accuracy: 0.8160 - loss: 0.4327 - val_accuracy: 0.7702 - val_loss: 0.5861
Epoch 216/600
148/148 2s 13ms/step - accuracy: 0.8166 - loss: 0.4285 - val_accuracy: 0.7675 - val_loss: 0.5950
Epoch 217/600
148/148 2s 11ms/step - accuracy: 0.8151 - loss: 0.4352 - val_accuracy: 0.7718 - val_loss: 0.5892
Epoch 218/600
148/148 2s 10ms/step - accuracy: 0.8166 - loss: 0.4342 - val_accuracy: 0.7713 - val_loss: 0.5793
Epoch 219/600
148/148 2s 10ms/step - accuracy: 0.8167 - loss: 0.4350 - val_accuracy: 0.7700 - val_loss: 0.5862
Epoch 220/600
148/148 2s 10ms/step - accuracy: 0.8165 - loss: 0.4302 - val_accuracy: 0.7707 - val_loss: 0.5883
Epoch 221/600
148/148 2s 11ms/step - accuracy: 0.8177 - loss: 0.4287 - val_accuracy: 0.7733 - val_loss: 0.5813
Epoch 222/600
148/148 2s 11ms/step - accuracy: 0.8187 - loss: 0.4301 - val_accuracy: 0.7729 - val_loss: 0.5900
Epoch 223/600
148/148 2s 12ms/step - accuracy: 0.8175 - loss: 0.4294 - val_accuracy: 0.7711 - val_loss: 0.5863
Epoch 224/600
148/148 2s 11ms/step - accuracy: 0.8184 - loss: 0.4255 - val_accuracy: 0.7728 - val_loss: 0.5859
Epoch 225/600
148/148 2s 11ms/step - accuracy: 0.8182 - loss: 0.4307 - val_accuracy: 0.7707 - val_loss: 0.5920
Epoch 226/600
148/148 2s 11ms/step - accuracy: 0.8191 - loss: 0.4250 - val_accuracy: 0.7698 - val_loss: 0.5847
Epoch 227/600
148/148 2s 10ms/step - accuracy: 0.8169 - loss: 0.4295 - val_accuracy: 0.7703 - val_loss: 0.5882
Epoch 228/600
148/148 2s 12ms/step - accuracy: 0.8154 - loss: 0.4304 - val_accuracy: 0.7733 - val_loss: 0.5770

Epoch 229/600
148/148 2s 11ms/step - accuracy: 0.8181 - loss: 0.4287 - val_accuracy: 0.7696 - val_loss: 0.5800
Epoch 230/600
148/148 2s 13ms/step - accuracy: 0.8168 - loss: 0.4294 - val_accuracy: 0.7705 - val_loss: 0.5862
Epoch 231/600
148/148 2s 11ms/step - accuracy: 0.8185 - loss: 0.4264 - val_accuracy: 0.7683 - val_loss: 0.5855
Epoch 232/600
148/148 2s 11ms/step - accuracy: 0.8179 - loss: 0.4270 - val_accuracy: 0.7705 - val_loss: 0.5864
Epoch 233/600
148/148 2s 13ms/step - accuracy: 0.8154 - loss: 0.4267 - val_accuracy: 0.7708 - val_loss: 0.5876
Epoch 234/600
148/148 2s 13ms/step - accuracy: 0.8163 - loss: 0.4302 - val_accuracy: 0.7700 - val_loss: 0.5942
Epoch 235/600
148/148 2s 12ms/step - accuracy: 0.8176 - loss: 0.4271 - val_accuracy: 0.7734 - val_loss: 0.5849
Epoch 236/600
148/148 2s 12ms/step - accuracy: 0.8199 - loss: 0.4272 - val_accuracy: 0.7714 - val_loss: 0.5929
Epoch 237/600
148/148 2s 12ms/step - accuracy: 0.8162 - loss: 0.4273 - val_accuracy: 0.7714 - val_loss: 0.5930
Epoch 238/600
148/148 2s 10ms/step - accuracy: 0.8169 - loss: 0.4269 - val_accuracy: 0.7731 - val_loss: 0.5864
Epoch 239/600
148/148 2s 10ms/step - accuracy: 0.8186 - loss: 0.4246 - val_accuracy: 0.7715 - val_loss: 0.5847
Epoch 240/600
148/148 2s 11ms/step - accuracy: 0.8166 - loss: 0.4290 - val_accuracy: 0.7734 - val_loss: 0.6030
Epoch 241/600
148/148 2s 11ms/step - accuracy: 0.8195 - loss: 0.4262 - val_accuracy: 0.7697 - val_loss: 0.5965
Epoch 242/600
148/148 2s 11ms/step - accuracy: 0.8188 - loss: 0.4255 - val_accuracy: 0.7731 - val_loss: 0.5900
Epoch 243/600
148/148 2s 12ms/step - accuracy: 0.8200 - loss: 0.4253 - val_accuracy: 0.7713 - val_loss: 0.5949
Epoch 244/600
148/148 2s 12ms/step - accuracy: 0.8222 - loss: 0.4205 - val_accuracy: 0.7702 - val_loss: 0.5856
Epoch 245/600
148/148 2s 12ms/step - accuracy: 0.8179 - loss: 0.4285 - val_accuracy: 0.7698 - val_loss: 0.5875
Epoch 246/600
148/148 2s 12ms/step - accuracy: 0.8199 - loss: 0.4195 - val_accuracy: 0.7695 - val_loss: 0.5893
Epoch 247/600
148/148 2s 11ms/step - accuracy: 0.8195 - loss: 0.4252 - val_accuracy: 0.7724 - val_loss: 0.5930
Epoch 248/600
148/148 2s 11ms/step - accuracy: 0.8212 - loss: 0.4209 - val_accuracy: 0.7702 - val_loss: 0.5878
Epoch 249/600
148/148 2s 11ms/step - accuracy: 0.8183 - loss: 0.4278 - val_accuracy: 0.7739 - val_loss: 0.5926
Epoch 250/600
148/148 2s 11ms/step - accuracy: 0.8190 - loss: 0.4260 - val_accuracy: 0.7708 - val_loss: 0.5903
Epoch 251/600
148/148 2s 12ms/step - accuracy: 0.8185 - loss: 0.4283 - val_accuracy: 0.7715 - val_loss: 0.5870
Epoch 252/600
148/148 2s 11ms/step - accuracy: 0.8202 - loss: 0.4254 - val_accuracy: 0.7721 - val_loss: 0.5846
Epoch 253/600
148/148 2s 12ms/step - accuracy: 0.8208 - loss: 0.4184 - val_accuracy: 0.7717 - val_loss: 0.5852
Epoch 254/600

148/148 2s 10ms/step - accuracy: 0.8241 - loss: 0.4157 - val_accuracy: 0.7713 - val_loss: 0.5884
Epoch 255/600
148/148 2s 11ms/step - accuracy: 0.8200 - loss: 0.4220 - val_accuracy: 0.7713 - val_loss: 0.5869
Epoch 256/600
148/148 2s 11ms/step - accuracy: 0.8205 - loss: 0.4288 - val_accuracy: 0.7710 - val_loss: 0.5857
Epoch 257/600
148/148 2s 10ms/step - accuracy: 0.8205 - loss: 0.4206 - val_accuracy: 0.7723 - val_loss: 0.5879
Epoch 258/600
148/148 2s 11ms/step - accuracy: 0.8202 - loss: 0.4202 - val_accuracy: 0.7704 - val_loss: 0.5874
Epoch 259/600
148/148 2s 11ms/step - accuracy: 0.8210 - loss: 0.4228 - val_accuracy: 0.7727 - val_loss: 0.5857
Epoch 260/600
148/148 2s 11ms/step - accuracy: 0.8208 - loss: 0.4215 - val_accuracy: 0.7701 - val_loss: 0.5931
Epoch 261/600
148/148 2s 12ms/step - accuracy: 0.8224 - loss: 0.4198 - val_accuracy: 0.7742 - val_loss: 0.5857
Epoch 262/600
148/148 2s 11ms/step - accuracy: 0.8225 - loss: 0.4176 - val_accuracy: 0.7706 - val_loss: 0.5975
Epoch 263/600
148/148 2s 10ms/step - accuracy: 0.8217 - loss: 0.4179 - val_accuracy: 0.7745 - val_loss: 0.5851
Epoch 264/600
148/148 2s 11ms/step - accuracy: 0.8236 - loss: 0.4185 - val_accuracy: 0.7743 - val_loss: 0.5941
Epoch 265/600
148/148 2s 11ms/step - accuracy: 0.8187 - loss: 0.4216 - val_accuracy: 0.7725 - val_loss: 0.5880
Epoch 266/600
148/148 2s 10ms/step - accuracy: 0.8244 - loss: 0.4200 - val_accuracy: 0.7715 - val_loss: 0.5897
Epoch 267/600
148/148 2s 13ms/step - accuracy: 0.8215 - loss: 0.4152 - val_accuracy: 0.7715 - val_loss: 0.5966
Epoch 268/600
148/148 2s 10ms/step - accuracy: 0.8212 - loss: 0.4168 - val_accuracy: 0.7716 - val_loss: 0.5986
Epoch 269/600
148/148 2s 11ms/step - accuracy: 0.8257 - loss: 0.4125 - val_accuracy: 0.7711 - val_loss: 0.5832
Epoch 270/600
148/148 2s 12ms/step - accuracy: 0.8230 - loss: 0.4169 - val_accuracy: 0.7728 - val_loss: 0.5851
Epoch 271/600
148/148 2s 12ms/step - accuracy: 0.8245 - loss: 0.4166 - val_accuracy: 0.7727 - val_loss: 0.5884
Epoch 272/600
148/148 2s 11ms/step - accuracy: 0.8216 - loss: 0.4188 - val_accuracy: 0.7716 - val_loss: 0.5902
Epoch 273/600
148/148 2s 12ms/step - accuracy: 0.8225 - loss: 0.4211 - val_accuracy: 0.7695 - val_loss: 0.5879
Epoch 274/600
148/148 2s 13ms/step - accuracy: 0.8227 - loss: 0.4179 - val_accuracy: 0.7737 - val_loss: 0.5955
Epoch 275/600
148/148 2s 12ms/step - accuracy: 0.8226 - loss: 0.4203 - val_accuracy: 0.7717 - val_loss: 0.5914
Epoch 276/600
148/148 2s 12ms/step - accuracy: 0.8232 - loss: 0.4160 - val_accuracy: 0.7753 - val_loss: 0.5970
Epoch 277/600
148/148 2s 11ms/step - accuracy: 0.8275 - loss: 0.4101 - val_accuracy: 0.7705 - val_loss: 0.5889
Epoch 278/600
148/148 2s 11ms/step - accuracy: 0.8211 - loss: 0.4223 - val_accuracy: 0.7727 - val_loss: 0.5979
Epoch 279/600
148/148 2s 12ms/step - accuracy: 0.8224 - loss: 0.4186 - val_accuracy: 0.7716 - val_loss:

s: 0.5886
Epoch 280/600
148/148 2s 11ms/step - accuracy: 0.8222 - loss: 0.4136 - val_accuracy: 0.7713 - val_loss: 0.5907
Epoch 281/600
148/148 2s 11ms/step - accuracy: 0.8268 - loss: 0.4147 - val_accuracy: 0.7707 - val_loss: 0.5815
Epoch 282/600
148/148 2s 13ms/step - accuracy: 0.8244 - loss: 0.4136 - val_accuracy: 0.7735 - val_loss: 0.5986
s: 0.5862
Epoch 283/600
148/148 2s 10ms/step - accuracy: 0.8236 - loss: 0.4163 - val_accuracy: 0.7719 - val_loss: 0.6035
s: 0.5830
Epoch 285/600
148/148 2s 12ms/step - accuracy: 0.8231 - loss: 0.4153 - val_accuracy: 0.7707 - val_loss: 0.5959
s: 0.5869
Epoch 286/600
148/148 2s 13ms/step - accuracy: 0.8232 - loss: 0.4143 - val_accuracy: 0.7722 - val_loss: 0.5902
s: 0.5877
Epoch 288/600
148/148 2s 12ms/step - accuracy: 0.8229 - loss: 0.4165 - val_accuracy: 0.7723 - val_loss: 0.5909
s: 0.5915
Epoch 291/600
148/148 2s 10ms/step - accuracy: 0.8246 - loss: 0.4119 - val_accuracy: 0.7733 - val_loss: 0.5897
s: 0.5903
Epoch 293/600
148/148 2s 11ms/step - accuracy: 0.8269 - loss: 0.4142 - val_accuracy: 0.7740 - val_loss: 0.5926
s: 0.5867
Epoch 296/600
148/148 2s 10ms/step - accuracy: 0.8244 - loss: 0.4126 - val_accuracy: 0.7738 - val_loss: 0.5915
s: 0.5862
Epoch 298/600
148/148 2s 11ms/step - accuracy: 0.8256 - loss: 0.4151 - val_accuracy: 0.7722 - val_loss: 0.5893
s: 0.5867
Epoch 299/600
148/148 2s 11ms/step - accuracy: 0.8240 - loss: 0.4128 - val_accuracy: 0.7722 - val_loss: 0.5910
s: 0.5893
Epoch 301/600
148/148 2s 10ms/step - accuracy: 0.8255 - loss: 0.4086 - val_accuracy: 0.7741 - val_loss: 0.5910
s: 0.5851
Epoch 302/600
148/148 1s 10ms/step - accuracy: 0.8262 - loss: 0.4114 - val_accuracy: 0.7715 - val_loss: 0.5910
s: 0.5890
Epoch 303/600
148/148 2s 10ms/step - accuracy: 0.8259 - loss: 0.4097 - val_accuracy: 0.7728 - val_loss: 0.5890
s: 0.5890
Epoch 304/600
148/148 2s 12ms/step - accuracy: 0.8252 - loss: 0.4087 - val_accuracy: 0.7726 - val_loss: 0.5890
s: 0.5890

Epoch 305/600
148/148 2s 13ms/step - accuracy: 0.8243 - loss: 0.4117 - val_accuracy: 0.7734 - val_loss: 0.5984
Epoch 306/600
148/148 2s 11ms/step - accuracy: 0.8233 - loss: 0.4142 - val_accuracy: 0.7727 - val_loss: 0.5849
Epoch 307/600
148/148 2s 14ms/step - accuracy: 0.8287 - loss: 0.4082 - val_accuracy: 0.7727 - val_loss: 0.5897
s: 0.5935
Epoch 308/600
148/148 2s 11ms/step - accuracy: 0.8265 - loss: 0.4089 - val_accuracy: 0.7717 - val_loss: 0.5939
s: 0.5933
Epoch 310/600
148/148 2s 11ms/step - accuracy: 0.8269 - loss: 0.4088 - val_accuracy: 0.7727 - val_loss: 0.5933
s: 0.5995
Epoch 311/600
148/148 2s 10ms/step - accuracy: 0.8271 - loss: 0.4099 - val_accuracy: 0.7735 - val_loss: 0.5934
s: 0.5881
Epoch 312/600
148/148 2s 10ms/step - accuracy: 0.8280 - loss: 0.4089 - val_accuracy: 0.7734 - val_loss: 0.5881
s: 0.5911
Epoch 313/600
148/148 2s 10ms/step - accuracy: 0.8282 - loss: 0.4082 - val_accuracy: 0.7737 - val_loss: 0.5911
s: 0.5895
Epoch 315/600
148/148 2s 10ms/step - accuracy: 0.8274 - loss: 0.4095 - val_accuracy: 0.7761 - val_loss: 0.5895
s: 0.5897
Epoch 316/600
148/148 2s 11ms/step - accuracy: 0.8235 - loss: 0.4127 - val_accuracy: 0.7745 - val_loss: 0.5884
s: 0.5911
Epoch 317/600
148/148 2s 11ms/step - accuracy: 0.8260 - loss: 0.4095 - val_accuracy: 0.7742 - val_loss: 0.5883
s: 0.5883
Epoch 318/600
148/148 2s 11ms/step - accuracy: 0.8258 - loss: 0.4105 - val_accuracy: 0.7746 - val_loss: 0.5883
s: 0.5911
Epoch 319/600
148/148 2s 10ms/step - accuracy: 0.8300 - loss: 0.4036 - val_accuracy: 0.7746 - val_loss: 0.5883
s: 0.6035
Epoch 321/600
148/148 2s 11ms/step - accuracy: 0.8276 - loss: 0.4094 - val_accuracy: 0.7763 - val_loss: 0.5886
s: 0.5859
Epoch 322/600
148/148 2s 11ms/step - accuracy: 0.8272 - loss: 0.4068 - val_accuracy: 0.7723 - val_loss: 0.5878
s: 0.5878
Epoch 324/600
148/148 2s 11ms/step - accuracy: 0.8275 - loss: 0.4069 - val_accuracy: 0.7746 - val_loss: 0.6002
s: 0.5915
Epoch 325/600
148/148 2s 12ms/step - accuracy: 0.8297 - loss: 0.4046 - val_accuracy: 0.7730 - val_loss: 0.6043
s: 0.6043
Epoch 327/600
148/148 2s 11ms/step - accuracy: 0.8297 - loss: 0.4057 - val_accuracy: 0.7761 - val_loss: 0.5952
s: 0.6043
Epoch 328/600
148/148 2s 13ms/step - accuracy: 0.8310 - loss: 0.4049 - val_accuracy: 0.7741 - val_loss: 0.5896
s: 0.5942
Epoch 329/600
148/148 2s 12ms/step - accuracy: 0.8272 - loss: 0.4098 - val_accuracy: 0.7745 - val_loss: 0.5942
s: 0.5942
Epoch 330/600

148/148 2s 12ms/step - accuracy: 0.8269 - loss: 0.4048 - val_accuracy: 0.7750 - val_loss: 0.5876
Epoch 331/600
148/148 2s 10ms/step - accuracy: 0.8262 - loss: 0.4079 - val_accuracy: 0.7711 - val_loss: 0.6024
Epoch 332/600
148/148 2s 10ms/step - accuracy: 0.8272 - loss: 0.4063 - val_accuracy: 0.7752 - val_loss: 0.5894
Epoch 333/600
148/148 2s 10ms/step - accuracy: 0.8295 - loss: 0.4014 - val_accuracy: 0.7764 - val_loss: 0.5912
Epoch 334/600
148/148 2s 11ms/step - accuracy: 0.8309 - loss: 0.4016 - val_accuracy: 0.7738 - val_loss: 0.6032
Epoch 335/600
148/148 2s 11ms/step - accuracy: 0.8306 - loss: 0.4017 - val_accuracy: 0.7733 - val_loss: 0.5892
Epoch 336/600
148/148 2s 12ms/step - accuracy: 0.8286 - loss: 0.4060 - val_accuracy: 0.7752 - val_loss: 0.5953
Epoch 337/600
148/148 2s 11ms/step - accuracy: 0.8314 - loss: 0.4006 - val_accuracy: 0.7750 - val_loss: 0.5940
Epoch 338/600
148/148 2s 11ms/step - accuracy: 0.8299 - loss: 0.4037 - val_accuracy: 0.7742 - val_loss: 0.5897
Epoch 339/600
148/148 2s 11ms/step - accuracy: 0.8293 - loss: 0.4042 - val_accuracy: 0.7751 - val_loss: 0.5902
Epoch 340/600
148/148 2s 11ms/step - accuracy: 0.8280 - loss: 0.4046 - val_accuracy: 0.7738 - val_loss: 0.5911
Epoch 341/600
148/148 2s 11ms/step - accuracy: 0.8292 - loss: 0.4054 - val_accuracy: 0.7738 - val_loss: 0.5950
Epoch 342/600
148/148 2s 11ms/step - accuracy: 0.8283 - loss: 0.4028 - val_accuracy: 0.7744 - val_loss: 0.5909
Epoch 343/600
148/148 2s 11ms/step - accuracy: 0.8311 - loss: 0.3981 - val_accuracy: 0.7750 - val_loss: 0.5871
Epoch 344/600
148/148 2s 11ms/step - accuracy: 0.8332 - loss: 0.3980 - val_accuracy: 0.7737 - val_loss: 0.5916
Epoch 345/600
148/148 2s 11ms/step - accuracy: 0.8298 - loss: 0.4016 - val_accuracy: 0.7749 - val_loss: 0.5878
Epoch 346/600
148/148 2s 11ms/step - accuracy: 0.8289 - loss: 0.4048 - val_accuracy: 0.7726 - val_loss: 0.6017
Epoch 347/600
148/148 2s 11ms/step - accuracy: 0.8290 - loss: 0.4011 - val_accuracy: 0.7755 - val_loss: 0.5973
Epoch 348/600
148/148 2s 11ms/step - accuracy: 0.8294 - loss: 0.4075 - val_accuracy: 0.7756 - val_loss: 0.5932
Epoch 349/600
148/148 2s 11ms/step - accuracy: 0.8308 - loss: 0.3993 - val_accuracy: 0.7724 - val_loss: 0.5966
Epoch 350/600
148/148 2s 11ms/step - accuracy: 0.8334 - loss: 0.4007 - val_accuracy: 0.7755 - val_loss: 0.5982
Epoch 351/600
148/148 2s 11ms/step - accuracy: 0.8315 - loss: 0.4023 - val_accuracy: 0.7752 - val_loss: 0.5857
Epoch 352/600
148/148 2s 12ms/step - accuracy: 0.8285 - loss: 0.4086 - val_accuracy: 0.7738 - val_loss: 0.6025
Epoch 353/600
148/148 2s 11ms/step - accuracy: 0.8326 - loss: 0.3966 - val_accuracy: 0.7729 - val_loss: 0.5892
Epoch 354/600
148/148 2s 14ms/step - accuracy: 0.8284 - loss: 0.4012 - val_accuracy: 0.7747 - val_loss: 0.5957
Epoch 355/600
148/148 2s 12ms/step - accuracy: 0.8295 - loss: 0.3977 - val_accuracy: 0.7742 - val_loss:

s: 0.5956
Epoch 356/600
148/148 2s 13ms/step - accuracy: 0.8294 - loss: 0.4008 - val_accuracy: 0.7755 - val_loss: 0.5933
Epoch 357/600
148/148 2s 11ms/step - accuracy: 0.8289 - loss: 0.4026 - val_accuracy: 0.7756 - val_loss: 0.6001
Epoch 358/600
148/148 2s 11ms/step - accuracy: 0.8308 - loss: 0.4008 - val_accuracy: 0.7728 - val_loss: 0.5980
Epoch 359/600
148/148 2s 11ms/step - accuracy: 0.8308 - loss: 0.4009 - val_accuracy: 0.7733 - val_loss: 0.5940
Epoch 360/600
148/148 2s 11ms/step - accuracy: 0.8317 - loss: 0.4012 - val_accuracy: 0.7742 - val_loss: 0.6016
Epoch 361/600
148/148 2s 12ms/step - accuracy: 0.8296 - loss: 0.4029 - val_accuracy: 0.7756 - val_loss: 0.5926
Epoch 362/600
148/148 2s 11ms/step - accuracy: 0.8324 - loss: 0.3997 - val_accuracy: 0.7747 - val_loss: 0.5898
Epoch 363/600
148/148 2s 12ms/step - accuracy: 0.8301 - loss: 0.3977 - val_accuracy: 0.7727 - val_loss: 0.6127
Epoch 364/600
148/148 2s 12ms/step - accuracy: 0.8334 - loss: 0.3973 - val_accuracy: 0.7747 - val_loss: 0.6018
Epoch 365/600
148/148 2s 12ms/step - accuracy: 0.8335 - loss: 0.3962 - val_accuracy: 0.7724 - val_loss: 0.6017
Epoch 366/600
148/148 2s 12ms/step - accuracy: 0.8276 - loss: 0.4046 - val_accuracy: 0.7728 - val_loss: 0.5980
Epoch 367/600
148/148 2s 10ms/step - accuracy: 0.8298 - loss: 0.3999 - val_accuracy: 0.7741 - val_loss: 0.6009
Epoch 368/600
148/148 2s 10ms/step - accuracy: 0.8309 - loss: 0.3981 - val_accuracy: 0.7730 - val_loss: 0.6009
Epoch 369/600
148/148 2s 10ms/step - accuracy: 0.8337 - loss: 0.3999 - val_accuracy: 0.7731 - val_loss: 0.6028
Epoch 370/600
148/148 2s 14ms/step - accuracy: 0.8338 - loss: 0.3961 - val_accuracy: 0.7716 - val_loss: 0.6063
Epoch 371/600
148/148 2s 11ms/step - accuracy: 0.8334 - loss: 0.3940 - val_accuracy: 0.7728 - val_loss: 0.5997
Epoch 372/600
148/148 2s 11ms/step - accuracy: 0.8312 - loss: 0.4029 - val_accuracy: 0.7732 - val_loss: 0.5942
Epoch 373/600
148/148 2s 11ms/step - accuracy: 0.8291 - loss: 0.4020 - val_accuracy: 0.7731 - val_loss: 0.6015
Epoch 374/600
148/148 2s 12ms/step - accuracy: 0.8296 - loss: 0.3973 - val_accuracy: 0.7752 - val_loss: 0.6156
Epoch 375/600
148/148 2s 11ms/step - accuracy: 0.8346 - loss: 0.3953 - val_accuracy: 0.7754 - val_loss: 0.5996
Epoch 376/600
148/148 2s 11ms/step - accuracy: 0.8303 - loss: 0.4026 - val_accuracy: 0.7761 - val_loss: 0.5966
Epoch 377/600
148/148 2s 11ms/step - accuracy: 0.8339 - loss: 0.3928 - val_accuracy: 0.7739 - val_loss: 0.5960
Epoch 378/600
148/148 2s 10ms/step - accuracy: 0.8319 - loss: 0.3989 - val_accuracy: 0.7741 - val_loss: 0.5974
Epoch 379/600
148/148 2s 10ms/step - accuracy: 0.8301 - loss: 0.4024 - val_accuracy: 0.7727 - val_loss: 0.5917
Epoch 380/600
148/148 2s 11ms/step - accuracy: 0.8324 - loss: 0.3943 - val_accuracy: 0.7729 - val_loss: 0.5959

```
Epoch 381/600
148/148 2s 11ms/step - accuracy: 0.8323 - loss: 0.3999 - val_accuracy: 0.7753 - val_loss: 0.5900
Epoch 382/600
148/148 2s 11ms/step - accuracy: 0.8323 - loss: 0.3953 - val_accuracy: 0.7758 - val_loss: 0.5968
Epoch 383/600
148/148 2s 11ms/step - accuracy: 0.8305 - loss: 0.4017 - val_accuracy: 0.7739 - val_loss: 0.5982
Epoch 384/600
148/148 2s 11ms/step - accuracy: 0.8311 - loss: 0.3967 - val_accuracy: 0.7755 - val_loss: 0.6047
Epoch 385/600
148/148 2s 11ms/step - accuracy: 0.8310 - loss: 0.3985 - val_accuracy: 0.7733 - val_loss: 0.5994
Epoch 386/600
148/148 2s 12ms/step - accuracy: 0.8339 - loss: 0.3962 - val_accuracy: 0.7737 - val_loss: 0.5956
Epoch 387/600
148/148 2s 11ms/step - accuracy: 0.8337 - loss: 0.3945 - val_accuracy: 0.7724 - val_loss: 0.5935
Epoch 388/600
148/148 2s 11ms/step - accuracy: 0.8316 - loss: 0.3993 - val_accuracy: 0.7748 - val_loss: 0.5987
Epoch 389/600
148/148 2s 11ms/step - accuracy: 0.8336 - loss: 0.3955 - val_accuracy: 0.7728 - val_loss: 0.6014
Epoch 390/600
148/148 2s 11ms/step - accuracy: 0.8324 - loss: 0.3959 - val_accuracy: 0.7744 - val_loss: 0.6107
Epoch 391/600
148/148 2s 10ms/step - accuracy: 0.8318 - loss: 0.3968 - val_accuracy: 0.7749 - val_loss: 0.5937
Epoch 392/600
148/148 2s 11ms/step - accuracy: 0.8331 - loss: 0.3966 - val_accuracy: 0.7757 - val_loss: 0.5960
Epoch 393/600
148/148 2s 11ms/step - accuracy: 0.8307 - loss: 0.3989 - val_accuracy: 0.7746 - val_loss: 0.6011
```

```
In [251... ann5.evaluate(X_train, y_train)
```

```
2363/2363 2s 1ms/step - accuracy: 0.9046 - loss: 0.2586
```

```
Out[251... [0.259221613407135, 0.9053961038589478]
```

```
In [252... ann5.summary()
```

```
Model: "sequential_6"
```

Layer (type)	Output Shape	Param #
dense_36 (Dense)	(None, 512)	23,040
dropout_30 (Dropout)	(None, 512)	0
dense_37 (Dense)	(None, 256)	131,328
dropout_31 (Dropout)	(None, 256)	0
dense_38 (Dense)	(None, 256)	65,792
dropout_32 (Dropout)	(None, 256)	0
dense_39 (Dense)	(None, 128)	32,896
dropout_33 (Dropout)	(None, 128)	0
dense_40 (Dense)	(None, 128)	16,512
dropout_34 (Dropout)	(None, 128)	0
dense_41 (Dense)	(None, 64)	8,256
dropout_35 (Dropout)	(None, 64)	0
dense_42 (Dense)	(None, 3)	195

Total params: 834,059 (3.18 MB)

Trainable params: 278,019 (1.06 MB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 556,040 (2.12 MB)

```
In [253]: eval_metric(ann5, X_train, y_train, X_val, y_val)
```

```
2363/2363 ━━━━━━━━ 3s 1ms/step
591/591 ━━━━━━ 1s 989us/step
```

Test Set:

```
[[4421 943 144]
 [1319 7628 1106]
 [ 63 721 2558]]
```

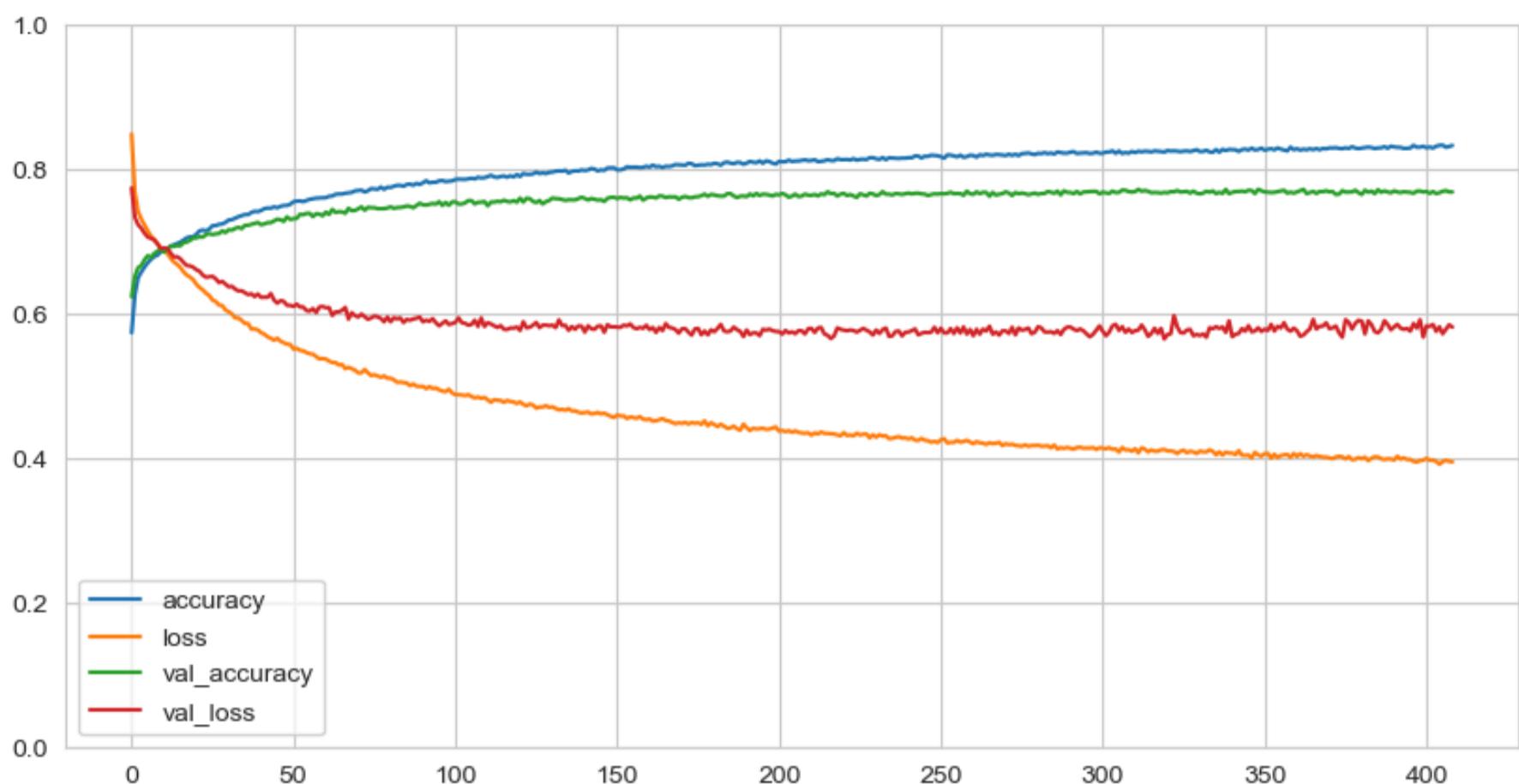
	precision	recall	f1-score	support
0	0.76	0.80	0.78	5508
1	0.82	0.76	0.79	10053
2	0.67	0.77	0.72	3342
accuracy			0.77	18903
macro avg	0.75	0.78	0.76	18903
weighted avg	0.78	0.77	0.77	18903

Train Set:

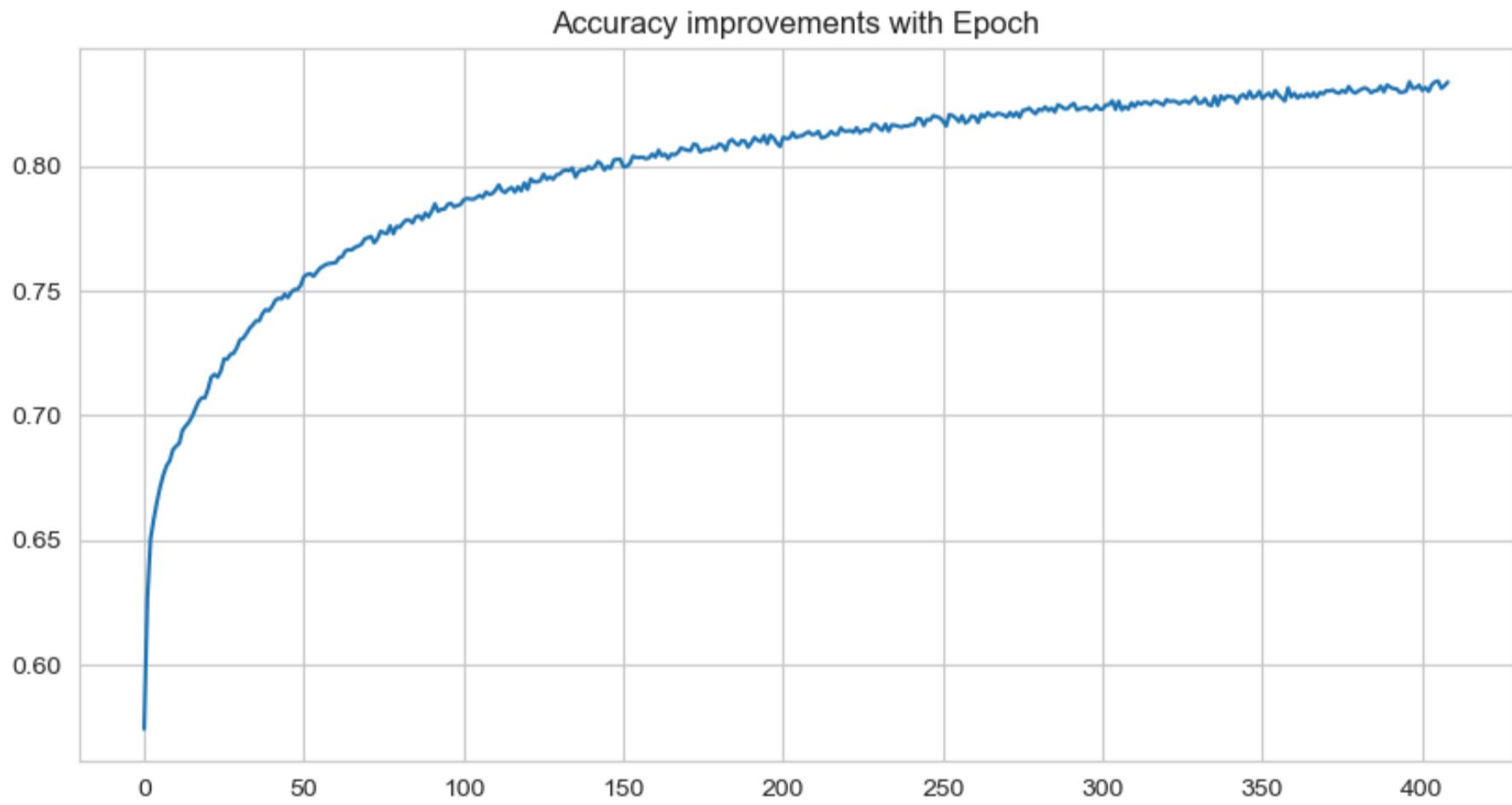
```
[[20875 1036 120]
 [ 2720 34939 2550]
 [ 23 704 12643]]
```

	precision	recall	f1-score	support
0	0.88	0.95	0.91	22031
1	0.95	0.87	0.91	40209
2	0.83	0.95	0.88	13370
accuracy			0.91	75610
macro avg	0.89	0.92	0.90	75610
weighted avg	0.91	0.91	0.91	75610

```
In [254]: pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



```
In [255...]: pd.DataFrame(history.history)[["accuracy"]].plot(figsize=(10, 5))
plt.title("Accuracy improvements with Epoch")
plt.show()
```



```
In [230...]: # Save the Model

from tensorflow.keras.models import load_model

# Save the model to 'model.h5' file
ann5.save('ann5_model.h5')

# To Load the model Later for further use
loaded_ann5 = load_model('ann5_model.h5')
```

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)` . This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')` .
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.

ANN-5 Model Summary:

- **(Dense) layers: 7 / Neurons: 512-256-256-128-128-64 / Dropout: 30-30-25-25-20-20% / Learning Rate: 0.001 / Batch Size: 512 / Epochs: 600 / Early Stop (val_accuracy): 60**

- **Accuracy:** 0.8318 / **Val_Accuracy:** 0.7749 / **Loss:** 0.3968 / **Val_Loss:** 0.5937 / **Train Recall (Class 2):** 0.95 / **Test Recall (Class 2):** 0.77

Improvements: Enhanced batch normalization throughout the network has helped stabilize training, which may contribute to the consistent recall rate for Class 2.

No Improvement: Despite improvements in training procedures, validation accuracy and loss have not improved significantly, suggesting challenges in model generalization remain.

Got Worse: Compared to previous models, the loss metrics, particularly validation loss, show less improvement, indicating ongoing issues with model fit on unseen data, possibly due to overfitting or insufficient regularization.

ANN-5 Model with SMOTE (%91)

```
In [29]: print('Credi_Score Classes before Smote: ', y_train.value_counts())
print('\nCredi_Score Classes after Smote: ', y_train_smote.value_counts())
```

```
Credi_Score Classes before Smote: Credit_Score
1    40209
0    22031
2    13370
Name: count, dtype: int64
```

```
Credi_Score Classes after Smote: Credit_Score
1    40209
2    40209
0    40209
Name: count, dtype: int64
```

```
In [263... # ANN-5 Model with SMOTE
```

```
# 1) Model Architecture:
# Fully connected (Dense) --> 7 Layers
ann5_smote = Sequential([
    Dense(512, input_dim=X_train.shape[1], activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(256, activation='relu'),
    BatchNormalization(),    # BatchNormalization() is added!
    Dropout(0.3),
    Dense(256, activation='relu'),
    BatchNormalization(),
    Dropout(0.25),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.25),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.2),
    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.2),
    Dense(3, activation='softmax')
])
```

```
# 2) Compiling the Model:
ann5_smote.compile(optimizer=Adam(learning_rate=0.001),
                    loss='sparse_categorical_crossentropy',
                    metrics=['accuracy'])
```

```
# 3) Early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=60,
                               mode="auto",
                               restore_best_weights=True)
```

```
# 4) Train the model
history = ann5_smote.fit(
            x=X_train,
            y=y_train,
```

```
validation_data=(X_val, y_val),  
batch_size=512,  
epochs=600,  
verbose=1,  
callbacks=[early_stopping])
```

Epoch 1/600
148/148 8s 13ms/step - accuracy: 0.4588 - loss: 1.1913 - val_accuracy: 0.5341 - val_loss: 0.9443
Epoch 2/600
148/148 2s 10ms/step - accuracy: 0.5859 - loss: 0.8534 - val_accuracy: 0.5827 - val_loss: 0.8524
Epoch 3/600
148/148 2s 12ms/step - accuracy: 0.6203 - loss: 0.7904 - val_accuracy: 0.6454 - val_loss: 0.7414
Epoch 4/600
148/148 2s 11ms/step - accuracy: 0.6351 - loss: 0.7613 - val_accuracy: 0.6662 - val_loss: 0.7241
Epoch 5/600
148/148 2s 11ms/step - accuracy: 0.6484 - loss: 0.7461 - val_accuracy: 0.6730 - val_loss: 0.7158
Epoch 6/600
148/148 2s 10ms/step - accuracy: 0.6530 - loss: 0.7340 - val_accuracy: 0.6766 - val_loss: 0.7109
Epoch 7/600
148/148 2s 11ms/step - accuracy: 0.6638 - loss: 0.7277 - val_accuracy: 0.6764 - val_loss: 0.7108
Epoch 8/600
148/148 2s 11ms/step - accuracy: 0.6648 - loss: 0.7210 - val_accuracy: 0.6805 - val_loss: 0.7066
Epoch 9/600
148/148 2s 10ms/step - accuracy: 0.6686 - loss: 0.7164 - val_accuracy: 0.6822 - val_loss: 0.7065
Epoch 10/600
148/148 2s 11ms/step - accuracy: 0.6770 - loss: 0.7072 - val_accuracy: 0.6818 - val_loss: 0.7018
Epoch 11/600
148/148 2s 11ms/step - accuracy: 0.6754 - loss: 0.7065 - val_accuracy: 0.6850 - val_loss: 0.6998
Epoch 12/600
148/148 2s 11ms/step - accuracy: 0.6790 - loss: 0.7035 - val_accuracy: 0.6851 - val_loss: 0.6985
Epoch 13/600
148/148 2s 11ms/step - accuracy: 0.6814 - loss: 0.6988 - val_accuracy: 0.6880 - val_loss: 0.6933
Epoch 14/600
148/148 2s 11ms/step - accuracy: 0.6863 - loss: 0.6927 - val_accuracy: 0.6879 - val_loss: 0.6926
Epoch 15/600
148/148 2s 11ms/step - accuracy: 0.6869 - loss: 0.6916 - val_accuracy: 0.6891 - val_loss: 0.6880
Epoch 16/600
148/148 2s 10ms/step - accuracy: 0.6917 - loss: 0.6837 - val_accuracy: 0.6899 - val_loss: 0.6895
Epoch 17/600
148/148 2s 10ms/step - accuracy: 0.6923 - loss: 0.6795 - val_accuracy: 0.6940 - val_loss: 0.6831
Epoch 18/600
148/148 2s 10ms/step - accuracy: 0.6944 - loss: 0.6795 - val_accuracy: 0.6935 - val_loss: 0.6832
Epoch 19/600
148/148 2s 11ms/step - accuracy: 0.6980 - loss: 0.6752 - val_accuracy: 0.6931 - val_loss: 0.6801
Epoch 20/600
148/148 2s 10ms/step - accuracy: 0.6988 - loss: 0.6703 - val_accuracy: 0.6935 - val_loss: 0.6804
Epoch 21/600
148/148 2s 11ms/step - accuracy: 0.6996 - loss: 0.6652 - val_accuracy: 0.6991 - val_loss: 0.6757
Epoch 22/600
148/148 2s 13ms/step - accuracy: 0.6982 - loss: 0.6703 - val_accuracy: 0.7000 - val_loss: 0.6773
Epoch 23/600
148/148 2s 13ms/step - accuracy: 0.7043 - loss: 0.6639 - val_accuracy: 0.6997 - val_loss: 0.6752
Epoch 24/600
148/148 2s 13ms/step - accuracy: 0.7061 - loss: 0.6573 - val_accuracy: 0.6989 - val_loss: 0.6767
Epoch 25/600
148/148 3s 12ms/step - accuracy: 0.7080 - loss: 0.6521 - val_accuracy: 0.7026 - val_loss: 0.6665
Epoch 26/600

148/148 2s 13ms/step - accuracy: 0.7095 - loss: 0.6506 - val_accuracy: 0.7045 - val_loss: 0.6679
Epoch 27/600
148/148 2s 11ms/step - accuracy: 0.7113 - loss: 0.6486 - val_accuracy: 0.7022 - val_loss: 0.6672
Epoch 28/600
148/148 2s 11ms/step - accuracy: 0.7157 - loss: 0.6422 - val_accuracy: 0.7030 - val_loss: 0.6702
Epoch 29/600
148/148 2s 11ms/step - accuracy: 0.7174 - loss: 0.6385 - val_accuracy: 0.7062 - val_loss: 0.6627
Epoch 30/600
148/148 2s 12ms/step - accuracy: 0.7181 - loss: 0.6340 - val_accuracy: 0.7104 - val_loss: 0.6600
Epoch 31/600
148/148 2s 13ms/step - accuracy: 0.7215 - loss: 0.6283 - val_accuracy: 0.7105 - val_loss: 0.6567
Epoch 32/600
148/148 2s 12ms/step - accuracy: 0.7205 - loss: 0.6296 - val_accuracy: 0.7108 - val_loss: 0.6547
Epoch 33/600
148/148 2s 12ms/step - accuracy: 0.7239 - loss: 0.6240 - val_accuracy: 0.7093 - val_loss: 0.6571
Epoch 34/600
148/148 2s 10ms/step - accuracy: 0.7264 - loss: 0.6193 - val_accuracy: 0.7177 - val_loss: 0.6462
Epoch 35/600
148/148 2s 11ms/step - accuracy: 0.7262 - loss: 0.6188 - val_accuracy: 0.7193 - val_loss: 0.6509
Epoch 36/600
148/148 2s 10ms/step - accuracy: 0.7318 - loss: 0.6121 - val_accuracy: 0.7158 - val_loss: 0.6416
Epoch 37/600
148/148 2s 11ms/step - accuracy: 0.7270 - loss: 0.6165 - val_accuracy: 0.7158 - val_loss: 0.6454
Epoch 38/600
148/148 2s 11ms/step - accuracy: 0.7335 - loss: 0.6080 - val_accuracy: 0.7204 - val_loss: 0.6420
Epoch 39/600
148/148 2s 11ms/step - accuracy: 0.7358 - loss: 0.6025 - val_accuracy: 0.7208 - val_loss: 0.6407
Epoch 40/600
148/148 2s 11ms/step - accuracy: 0.7326 - loss: 0.6015 - val_accuracy: 0.7218 - val_loss: 0.6369
Epoch 41/600
148/148 2s 12ms/step - accuracy: 0.7353 - loss: 0.5976 - val_accuracy: 0.7264 - val_loss: 0.6357
Epoch 42/600
148/148 2s 12ms/step - accuracy: 0.7359 - loss: 0.5999 - val_accuracy: 0.7302 - val_loss: 0.6327
Epoch 43/600
148/148 2s 13ms/step - accuracy: 0.7404 - loss: 0.5942 - val_accuracy: 0.7278 - val_loss: 0.6337
Epoch 44/600
148/148 2s 13ms/step - accuracy: 0.7430 - loss: 0.5902 - val_accuracy: 0.7309 - val_loss: 0.6305
Epoch 45/600
148/148 2s 11ms/step - accuracy: 0.7428 - loss: 0.5871 - val_accuracy: 0.7305 - val_loss: 0.6241
Epoch 46/600
148/148 2s 11ms/step - accuracy: 0.7427 - loss: 0.5849 - val_accuracy: 0.7341 - val_loss: 0.6236
Epoch 47/600
148/148 2s 11ms/step - accuracy: 0.7480 - loss: 0.5805 - val_accuracy: 0.7295 - val_loss: 0.6266
Epoch 48/600
148/148 2s 12ms/step - accuracy: 0.7474 - loss: 0.5782 - val_accuracy: 0.7359 - val_loss: 0.6183
Epoch 49/600
148/148 2s 12ms/step - accuracy: 0.7474 - loss: 0.5768 - val_accuracy: 0.7296 - val_loss: 0.6307
Epoch 50/600
148/148 2s 11ms/step - accuracy: 0.7501 - loss: 0.5723 - val_accuracy: 0.7355 - val_loss: 0.6180
Epoch 51/600
148/148 2s 12ms/step - accuracy: 0.7508 - loss: 0.5694 - val_accuracy: 0.7368 - val_loss:

s: 0.6142
Epoch 52/600
148/148 2s 12ms/step - accuracy: 0.7537 - loss: 0.5691 - val_accuracy: 0.7378 - val_loss: 0.5874
s: 0.6147
Epoch 53/600
148/148 2s 12ms/step - accuracy: 0.7525 - loss: 0.5640 - val_accuracy: 0.7398 - val_loss: 0.5874
s: 0.6074
Epoch 54/600
148/148 2s 11ms/step - accuracy: 0.7568 - loss: 0.5592 - val_accuracy: 0.7365 - val_loss: 0.5874
s: 0.6160
Epoch 55/600
148/148 2s 10ms/step - accuracy: 0.7551 - loss: 0.5616 - val_accuracy: 0.7388 - val_loss: 0.5874
s: 0.6132
Epoch 56/600
148/148 2s 11ms/step - accuracy: 0.7577 - loss: 0.5571 - val_accuracy: 0.7404 - val_loss: 0.5874
s: 0.6089
Epoch 57/600
148/148 2s 12ms/step - accuracy: 0.7573 - loss: 0.5580 - val_accuracy: 0.7396 - val_loss: 0.5874
s: 0.6098
Epoch 58/600
148/148 2s 12ms/step - accuracy: 0.7589 - loss: 0.5521 - val_accuracy: 0.7445 - val_loss: 0.5874
s: 0.6082
Epoch 59/600
148/148 2s 13ms/step - accuracy: 0.7599 - loss: 0.5489 - val_accuracy: 0.7458 - val_loss: 0.5874
s: 0.6128
Epoch 60/600
148/148 2s 14ms/step - accuracy: 0.7613 - loss: 0.5524 - val_accuracy: 0.7423 - val_loss: 0.5874
s: 0.6068
Epoch 61/600
148/148 2s 13ms/step - accuracy: 0.7592 - loss: 0.5484 - val_accuracy: 0.7425 - val_loss: 0.5874
s: 0.6105
Epoch 62/600
148/148 2s 10ms/step - accuracy: 0.7620 - loss: 0.5470 - val_accuracy: 0.7436 - val_loss: 0.5874
s: 0.6035
Epoch 63/600
148/148 2s 10ms/step - accuracy: 0.7644 - loss: 0.5433 - val_accuracy: 0.7467 - val_loss: 0.5874
s: 0.6048
Epoch 64/600
148/148 1s 10ms/step - accuracy: 0.7623 - loss: 0.5390 - val_accuracy: 0.7444 - val_loss: 0.5874
s: 0.6040
Epoch 65/600
148/148 2s 10ms/step - accuracy: 0.7649 - loss: 0.5378 - val_accuracy: 0.7467 - val_loss: 0.5874
s: 0.6052
Epoch 66/600
148/148 1s 10ms/step - accuracy: 0.7662 - loss: 0.5366 - val_accuracy: 0.7469 - val_loss: 0.5874
s: 0.5998
Epoch 67/600
148/148 2s 13ms/step - accuracy: 0.7670 - loss: 0.5365 - val_accuracy: 0.7481 - val_loss: 0.5874
s: 0.6010
Epoch 68/600
148/148 2s 11ms/step - accuracy: 0.7676 - loss: 0.5337 - val_accuracy: 0.7496 - val_loss: 0.5874
s: 0.5947
Epoch 69/600
148/148 2s 12ms/step - accuracy: 0.7681 - loss: 0.5347 - val_accuracy: 0.7496 - val_loss: 0.5874
s: 0.5915
Epoch 70/600
148/148 2s 11ms/step - accuracy: 0.7675 - loss: 0.5314 - val_accuracy: 0.7506 - val_loss: 0.5874
s: 0.5986
Epoch 71/600
148/148 2s 11ms/step - accuracy: 0.7703 - loss: 0.5273 - val_accuracy: 0.7533 - val_loss: 0.5874
s: 0.5928
Epoch 72/600
148/148 2s 10ms/step - accuracy: 0.7714 - loss: 0.5289 - val_accuracy: 0.7504 - val_loss: 0.5874
s: 0.5889
Epoch 73/600
148/148 2s 10ms/step - accuracy: 0.7738 - loss: 0.5235 - val_accuracy: 0.7491 - val_loss: 0.5874
s: 0.5956
Epoch 74/600
148/148 2s 11ms/step - accuracy: 0.7701 - loss: 0.5288 - val_accuracy: 0.7550 - val_loss: 0.5874
s: 0.5874
Epoch 75/600
148/148 2s 10ms/step - accuracy: 0.7733 - loss: 0.5241 - val_accuracy: 0.7521 - val_loss: 0.5874
s: 0.5925
Epoch 76/600
148/148 1s 10ms/step - accuracy: 0.7740 - loss: 0.5206 - val_accuracy: 0.7546 - val_loss: 0.5874
s: 0.5920

Epoch 77/600
148/148 2s 12ms/step - accuracy: 0.7764 - loss: 0.5196 - val_accuracy: 0.7509 - val_loss: 0.5938
Epoch 78/600
148/148 2s 12ms/step - accuracy: 0.7773 - loss: 0.5159 - val_accuracy: 0.7528 - val_loss: 0.5910
Epoch 79/600
148/148 2s 11ms/step - accuracy: 0.7761 - loss: 0.5142 - val_accuracy: 0.7534 - val_loss: 0.5975
Epoch 80/600
148/148 2s 13ms/step - accuracy: 0.7754 - loss: 0.5169 - val_accuracy: 0.7556 - val_loss: 0.5860
Epoch 81/600
148/148 2s 11ms/step - accuracy: 0.7766 - loss: 0.5148 - val_accuracy: 0.7547 - val_loss: 0.5956
Epoch 82/600
148/148 2s 11ms/step - accuracy: 0.7766 - loss: 0.5164 - val_accuracy: 0.7551 - val_loss: 0.5921
Epoch 83/600
148/148 2s 12ms/step - accuracy: 0.7794 - loss: 0.5105 - val_accuracy: 0.7537 - val_loss: 0.5886
Epoch 84/600
148/148 2s 12ms/step - accuracy: 0.7760 - loss: 0.5131 - val_accuracy: 0.7556 - val_loss: 0.5915
Epoch 85/600
148/148 2s 11ms/step - accuracy: 0.7785 - loss: 0.5127 - val_accuracy: 0.7577 - val_loss: 0.5864
Epoch 86/600
148/148 2s 12ms/step - accuracy: 0.7829 - loss: 0.5014 - val_accuracy: 0.7561 - val_loss: 0.5879
Epoch 87/600
148/148 2s 11ms/step - accuracy: 0.7817 - loss: 0.5038 - val_accuracy: 0.7577 - val_loss: 0.5867
Epoch 88/600
148/148 2s 14ms/step - accuracy: 0.7822 - loss: 0.5042 - val_accuracy: 0.7587 - val_loss: 0.5913
Epoch 89/600
148/148 2s 14ms/step - accuracy: 0.7808 - loss: 0.5048 - val_accuracy: 0.7584 - val_loss: 0.5843
Epoch 90/600
148/148 2s 12ms/step - accuracy: 0.7821 - loss: 0.5007 - val_accuracy: 0.7585 - val_loss: 0.5856
Epoch 91/600
148/148 2s 11ms/step - accuracy: 0.7822 - loss: 0.5029 - val_accuracy: 0.7562 - val_loss: 0.5869
Epoch 92/600
148/148 2s 12ms/step - accuracy: 0.7829 - loss: 0.4999 - val_accuracy: 0.7584 - val_loss: 0.5872
Epoch 93/600
148/148 2s 12ms/step - accuracy: 0.7835 - loss: 0.5008 - val_accuracy: 0.7601 - val_loss: 0.5854
Epoch 94/600
148/148 2s 11ms/step - accuracy: 0.7838 - loss: 0.4986 - val_accuracy: 0.7596 - val_loss: 0.5896
Epoch 95/600
148/148 2s 11ms/step - accuracy: 0.7815 - loss: 0.5028 - val_accuracy: 0.7587 - val_loss: 0.5933
Epoch 96/600
148/148 2s 11ms/step - accuracy: 0.7861 - loss: 0.4946 - val_accuracy: 0.7610 - val_loss: 0.5852
Epoch 97/600
148/148 2s 12ms/step - accuracy: 0.7835 - loss: 0.4968 - val_accuracy: 0.7598 - val_loss: 0.5869
Epoch 98/600
148/148 2s 12ms/step - accuracy: 0.7859 - loss: 0.4922 - val_accuracy: 0.7581 - val_loss: 0.5844
Epoch 99/600
148/148 2s 11ms/step - accuracy: 0.7874 - loss: 0.4934 - val_accuracy: 0.7609 - val_loss: 0.5833
Epoch 100/600
148/148 2s 11ms/step - accuracy: 0.7882 - loss: 0.4921 - val_accuracy: 0.7632 - val_loss: 0.5869
Epoch 101/600
148/148 2s 11ms/step - accuracy: 0.7894 - loss: 0.4895 - val_accuracy: 0.7591 - val_loss: 0.5834
Epoch 102/600

148/148 2s 10ms/step - accuracy: 0.7885 - loss: 0.4910 - val_accuracy: 0.7611 - val_loss: 0.5791
Epoch 103/600
148/148 2s 13ms/step - accuracy: 0.7887 - loss: 0.4876 - val_accuracy: 0.7638 - val_loss: 0.5806
Epoch 104/600
148/148 2s 12ms/step - accuracy: 0.7871 - loss: 0.4962 - val_accuracy: 0.7614 - val_loss: 0.5863
Epoch 105/600
148/148 2s 11ms/step - accuracy: 0.7890 - loss: 0.4915 - val_accuracy: 0.7618 - val_loss: 0.5877
Epoch 106/600
148/148 2s 12ms/step - accuracy: 0.7905 - loss: 0.4871 - val_accuracy: 0.7611 - val_loss: 0.5834
Epoch 107/600
148/148 2s 11ms/step - accuracy: 0.7922 - loss: 0.4845 - val_accuracy: 0.7618 - val_loss: 0.5857
Epoch 108/600
148/148 2s 11ms/step - accuracy: 0.7924 - loss: 0.4854 - val_accuracy: 0.7626 - val_loss: 0.5869
Epoch 109/600
148/148 2s 11ms/step - accuracy: 0.7943 - loss: 0.4831 - val_accuracy: 0.7592 - val_loss: 0.5905
Epoch 110/600
148/148 2s 11ms/step - accuracy: 0.7928 - loss: 0.4825 - val_accuracy: 0.7632 - val_loss: 0.5804
Epoch 111/600
148/148 2s 11ms/step - accuracy: 0.7917 - loss: 0.4812 - val_accuracy: 0.7620 - val_loss: 0.5780
Epoch 112/600
148/148 2s 11ms/step - accuracy: 0.7941 - loss: 0.4798 - val_accuracy: 0.7602 - val_loss: 0.5839
Epoch 113/600
148/148 2s 10ms/step - accuracy: 0.7918 - loss: 0.4825 - val_accuracy: 0.7633 - val_loss: 0.5863
Epoch 114/600
148/148 2s 12ms/step - accuracy: 0.7932 - loss: 0.4811 - val_accuracy: 0.7645 - val_loss: 0.5837
Epoch 115/600
148/148 2s 12ms/step - accuracy: 0.7935 - loss: 0.4806 - val_accuracy: 0.7646 - val_loss: 0.5867
Epoch 116/600
148/148 2s 12ms/step - accuracy: 0.7909 - loss: 0.4807 - val_accuracy: 0.7621 - val_loss: 0.5831
Epoch 117/600
148/148 2s 11ms/step - accuracy: 0.7936 - loss: 0.4810 - val_accuracy: 0.7626 - val_loss: 0.5889
Epoch 118/600
148/148 2s 11ms/step - accuracy: 0.7960 - loss: 0.4751 - val_accuracy: 0.7616 - val_loss: 0.5937
Epoch 119/600
148/148 2s 11ms/step - accuracy: 0.7973 - loss: 0.4750 - val_accuracy: 0.7618 - val_loss: 0.5808
Epoch 120/600
148/148 2s 11ms/step - accuracy: 0.7940 - loss: 0.4793 - val_accuracy: 0.7628 - val_loss: 0.5814
Epoch 121/600
148/148 2s 11ms/step - accuracy: 0.7944 - loss: 0.4776 - val_accuracy: 0.7643 - val_loss: 0.5792
Epoch 122/600
148/148 2s 11ms/step - accuracy: 0.7981 - loss: 0.4713 - val_accuracy: 0.7659 - val_loss: 0.5853
Epoch 123/600
148/148 2s 11ms/step - accuracy: 0.7982 - loss: 0.4705 - val_accuracy: 0.7672 - val_loss: 0.5771
Epoch 124/600
148/148 2s 11ms/step - accuracy: 0.7949 - loss: 0.4791 - val_accuracy: 0.7609 - val_loss: 0.5933
Epoch 125/600
148/148 2s 11ms/step - accuracy: 0.7933 - loss: 0.4774 - val_accuracy: 0.7628 - val_loss: 0.5829
Epoch 126/600
148/148 2s 12ms/step - accuracy: 0.7951 - loss: 0.4741 - val_accuracy: 0.7660 - val_loss: 0.5800
Epoch 127/600
148/148 2s 11ms/step - accuracy: 0.7965 - loss: 0.4699 - val_accuracy: 0.7635 - val_loss:

s: 0.5779
Epoch 128/600
148/148 2s 11ms/step - accuracy: 0.7957 - loss: 0.4723 - val_accuracy: 0.7632 - val_loss: 0.5889
Epoch 129/600
148/148 2s 11ms/step - accuracy: 0.7976 - loss: 0.4731 - val_accuracy: 0.7650 - val_loss: 0.5812
Epoch 130/600
148/148 2s 11ms/step - accuracy: 0.8008 - loss: 0.4647 - val_accuracy: 0.7632 - val_loss: 0.5857
s: 0.5790
Epoch 131/600
148/148 2s 11ms/step - accuracy: 0.7969 - loss: 0.4716 - val_accuracy: 0.7654 - val_loss: 0.5827
s: 0.5797
Epoch 132/600
148/148 2s 12ms/step - accuracy: 0.8008 - loss: 0.4659 - val_accuracy: 0.7659 - val_loss: 0.5799
s: 0.5828
Epoch 133/600
148/148 2s 11ms/step - accuracy: 0.8011 - loss: 0.4684 - val_accuracy: 0.7627 - val_loss: 0.5826
s: 0.5828
Epoch 135/600
148/148 2s 11ms/step - accuracy: 0.7982 - loss: 0.4700 - val_accuracy: 0.7662 - val_loss: 0.5751
s: 0.5845
Epoch 136/600
148/148 2s 11ms/step - accuracy: 0.7973 - loss: 0.4676 - val_accuracy: 0.7664 - val_loss: 0.5816
s: 0.5787
Epoch 138/600
148/148 2s 11ms/step - accuracy: 0.7982 - loss: 0.4662 - val_accuracy: 0.7650 - val_loss: 0.5808
s: 0.5822
Epoch 139/600
148/148 2s 11ms/step - accuracy: 0.7981 - loss: 0.4693 - val_accuracy: 0.7676 - val_loss: 0.5839
s: 0.5804
Epoch 141/600
148/148 2s 11ms/step - accuracy: 0.8023 - loss: 0.4624 - val_accuracy: 0.7661 - val_loss: 0.5858
s: 0.5905
Epoch 142/600
148/148 2s 11ms/step - accuracy: 0.7994 - loss: 0.4678 - val_accuracy: 0.7680 - val_loss: 0.5845
s: 0.5810
Epoch 144/600
148/148 2s 12ms/step - accuracy: 0.8060 - loss: 0.4588 - val_accuracy: 0.7678 - val_loss: 0.5858
s: 0.5835
Epoch 145/600
148/148 2s 12ms/step - accuracy: 0.7998 - loss: 0.4644 - val_accuracy: 0.7693 - val_loss: 0.5835
s: 0.5845
Epoch 146/600
148/148 2s 11ms/step - accuracy: 0.8000 - loss: 0.4604 - val_accuracy: 0.7689 - val_loss: 0.5845
s: 0.5901
Epoch 147/600
148/148 2s 11ms/step - accuracy: 0.8026 - loss: 0.4627 - val_accuracy: 0.7696 - val_loss: 0.5764
s: 0.5801
Epoch 148/600
148/148 2s 11ms/step - accuracy: 0.8032 - loss: 0.4581 - val_accuracy: 0.7660 - val_loss: 0.5764
s: 0.5796
Epoch 149/600
148/148 2s 11ms/step - accuracy: 0.8033 - loss: 0.4596 - val_accuracy: 0.7688 - val_loss: 0.5764
s: 0.5796
Epoch 150/600
148/148 2s 11ms/step - accuracy: 0.8041 - loss: 0.4590 - val_accuracy: 0.7674 - val_loss: 0.5841
s: 0.5841

Epoch 153/600
148/148 2s 11ms/step - accuracy: 0.8013 - loss: 0.4617 - val_accuracy: 0.7715 - val_loss: 0.5826
Epoch 154/600
148/148 2s 11ms/step - accuracy: 0.8080 - loss: 0.4484 - val_accuracy: 0.7684 - val_loss: 0.5766
Epoch 155/600
148/148 2s 11ms/step - accuracy: 0.8053 - loss: 0.4544 - val_accuracy: 0.7676 - val_loss: 0.5884
s: 0.5884
Epoch 156/600
148/148 2s 11ms/step - accuracy: 0.8027 - loss: 0.4587 - val_accuracy: 0.7665 - val_loss: 0.5844
s: 0.5844
Epoch 157/600
148/148 2s 11ms/step - accuracy: 0.8063 - loss: 0.4525 - val_accuracy: 0.7686 - val_loss: 0.5844
s: 0.5844
Epoch 158/600
148/148 2s 11ms/step - accuracy: 0.8018 - loss: 0.4612 - val_accuracy: 0.7681 - val_loss: 0.5890
s: 0.5890
Epoch 159/600
148/148 2s 11ms/step - accuracy: 0.8045 - loss: 0.4585 - val_accuracy: 0.7680 - val_loss: 0.5813
s: 0.5813
Epoch 160/600
148/148 2s 11ms/step - accuracy: 0.8077 - loss: 0.4539 - val_accuracy: 0.7699 - val_loss: 0.5797
s: 0.5797
Epoch 161/600
148/148 2s 11ms/step - accuracy: 0.8045 - loss: 0.4566 - val_accuracy: 0.7683 - val_loss: 0.5869
s: 0.5869
Epoch 162/600
148/148 2s 11ms/step - accuracy: 0.8060 - loss: 0.4505 - val_accuracy: 0.7693 - val_loss: 0.5792
s: 0.5792
Epoch 163/600
148/148 2s 11ms/step - accuracy: 0.8069 - loss: 0.4510 - val_accuracy: 0.7714 - val_loss: 0.5772
s: 0.5772
Epoch 164/600
148/148 2s 11ms/step - accuracy: 0.8064 - loss: 0.4507 - val_accuracy: 0.7702 - val_loss: 0.5798
s: 0.5798
Epoch 165/600
148/148 2s 11ms/step - accuracy: 0.8077 - loss: 0.4484 - val_accuracy: 0.7694 - val_loss: 0.5763
s: 0.5763
Epoch 166/600
148/148 2s 11ms/step - accuracy: 0.8085 - loss: 0.4462 - val_accuracy: 0.7691 - val_loss: 0.5815
s: 0.5815
Epoch 167/600
148/148 2s 11ms/step - accuracy: 0.8069 - loss: 0.4526 - val_accuracy: 0.7672 - val_loss: 0.5890
s: 0.5890
Epoch 168/600
148/148 2s 11ms/step - accuracy: 0.8066 - loss: 0.4519 - val_accuracy: 0.7691 - val_loss: 0.5829
s: 0.5829
Epoch 169/600
148/148 2s 11ms/step - accuracy: 0.8070 - loss: 0.4506 - val_accuracy: 0.7687 - val_loss: 0.5953
s: 0.5953
Epoch 170/600
148/148 2s 11ms/step - accuracy: 0.8082 - loss: 0.4483 - val_accuracy: 0.7694 - val_loss: 0.5908
s: 0.5908
Epoch 171/600
148/148 2s 11ms/step - accuracy: 0.8087 - loss: 0.4468 - val_accuracy: 0.7714 - val_loss: 0.5888
s: 0.5888
Epoch 172/600
148/148 2s 11ms/step - accuracy: 0.8084 - loss: 0.4477 - val_accuracy: 0.7687 - val_loss: 0.5867
s: 0.5867
Epoch 173/600
148/148 2s 11ms/step - accuracy: 0.8078 - loss: 0.4492 - val_accuracy: 0.7691 - val_loss: 0.5808
s: 0.5808
Epoch 174/600
148/148 2s 11ms/step - accuracy: 0.8123 - loss: 0.4422 - val_accuracy: 0.7697 - val_loss: 0.5787
s: 0.5787
Epoch 175/600
148/148 2s 11ms/step - accuracy: 0.8078 - loss: 0.4509 - val_accuracy: 0.7684 - val_loss: 0.5900
s: 0.5900
Epoch 176/600
148/148 2s 11ms/step - accuracy: 0.8070 - loss: 0.4509 - val_accuracy: 0.7685 - val_loss: 0.5940
s: 0.5940
Epoch 177/600
148/148 2s 11ms/step - accuracy: 0.8109 - loss: 0.4438 - val_accuracy: 0.7664 - val_loss: 0.5801
s: 0.5801
Epoch 178/600

148/148 2s 11ms/step - accuracy: 0.8104 - loss: 0.4462 - val_accuracy: 0.7688 - val_loss: 0.5822
Epoch 179/600
148/148 2s 11ms/step - accuracy: 0.8089 - loss: 0.4429 - val_accuracy: 0.7705 - val_loss: 0.5840
Epoch 180/600
148/148 2s 11ms/step - accuracy: 0.8075 - loss: 0.4464 - val_accuracy: 0.7707 - val_loss: 0.5909
Epoch 181/600
148/148 2s 11ms/step - accuracy: 0.8075 - loss: 0.4455 - val_accuracy: 0.7724 - val_loss: 0.5758
Epoch 182/600
148/148 2s 11ms/step - accuracy: 0.8120 - loss: 0.4442 - val_accuracy: 0.7704 - val_loss: 0.5882
Epoch 183/600
148/148 2s 12ms/step - accuracy: 0.8124 - loss: 0.4421 - val_accuracy: 0.7697 - val_loss: 0.5867
Epoch 184/600
148/148 2s 12ms/step - accuracy: 0.8125 - loss: 0.4377 - val_accuracy: 0.7690 - val_loss: 0.5839
Epoch 185/600
148/148 2s 11ms/step - accuracy: 0.8119 - loss: 0.4424 - val_accuracy: 0.7686 - val_loss: 0.5855
Epoch 186/600
148/148 2s 11ms/step - accuracy: 0.8107 - loss: 0.4432 - val_accuracy: 0.7692 - val_loss: 0.5829
Epoch 187/600
148/148 2s 11ms/step - accuracy: 0.8106 - loss: 0.4433 - val_accuracy: 0.7714 - val_loss: 0.5898
Epoch 188/600
148/148 2s 12ms/step - accuracy: 0.8141 - loss: 0.4409 - val_accuracy: 0.7709 - val_loss: 0.5862
Epoch 189/600
148/148 2s 11ms/step - accuracy: 0.8131 - loss: 0.4389 - val_accuracy: 0.7709 - val_loss: 0.5835
Epoch 190/600
148/148 2s 11ms/step - accuracy: 0.8138 - loss: 0.4414 - val_accuracy: 0.7699 - val_loss: 0.5914
Epoch 191/600
148/148 2s 11ms/step - accuracy: 0.8110 - loss: 0.4398 - val_accuracy: 0.7724 - val_loss: 0.5805
Epoch 192/600
148/148 2s 11ms/step - accuracy: 0.8116 - loss: 0.4417 - val_accuracy: 0.7695 - val_loss: 0.5808
Epoch 193/600
148/148 2s 11ms/step - accuracy: 0.8112 - loss: 0.4405 - val_accuracy: 0.7689 - val_loss: 0.5933
Epoch 194/600
148/148 2s 12ms/step - accuracy: 0.8117 - loss: 0.4389 - val_accuracy: 0.7696 - val_loss: 0.5969
Epoch 195/600
148/148 2s 12ms/step - accuracy: 0.8132 - loss: 0.4360 - val_accuracy: 0.7695 - val_loss: 0.5833
Epoch 196/600
148/148 2s 11ms/step - accuracy: 0.8089 - loss: 0.4463 - val_accuracy: 0.7691 - val_loss: 0.5910
Epoch 197/600
148/148 2s 12ms/step - accuracy: 0.8138 - loss: 0.4340 - val_accuracy: 0.7698 - val_loss: 0.5894
Epoch 198/600
148/148 2s 12ms/step - accuracy: 0.8152 - loss: 0.4342 - val_accuracy: 0.7710 - val_loss: 0.5852
Epoch 199/600
148/148 2s 12ms/step - accuracy: 0.8130 - loss: 0.4389 - val_accuracy: 0.7702 - val_loss: 0.5900
Epoch 200/600
148/148 2s 11ms/step - accuracy: 0.8135 - loss: 0.4373 - val_accuracy: 0.7706 - val_loss: 0.5848
Epoch 201/600
148/148 2s 14ms/step - accuracy: 0.8138 - loss: 0.4358 - val_accuracy: 0.7697 - val_loss: 0.5963
Epoch 202/600
148/148 2s 14ms/step - accuracy: 0.8121 - loss: 0.4386 - val_accuracy: 0.7720 - val_loss: 0.5791
Epoch 203/600
148/148 2s 12ms/step - accuracy: 0.8130 - loss: 0.4353 - val_accuracy: 0.7715 - val_loss:

s: 0.5854
Epoch 204/600
148/148 2s 14ms/step - accuracy: 0.8153 - loss: 0.4340 - val_accuracy: 0.7697 - val_loss: 0.5887
Epoch 205/600
148/148 2s 12ms/step - accuracy: 0.8135 - loss: 0.4346 - val_accuracy: 0.7727 - val_loss: 0.5874
Epoch 206/600
148/148 2s 12ms/step - accuracy: 0.8141 - loss: 0.4334 - val_accuracy: 0.7731 - val_loss: 0.5900
Epoch 207/600
148/148 2s 11ms/step - accuracy: 0.8152 - loss: 0.4297 - val_accuracy: 0.7729 - val_loss: 0.5857
Epoch 208/600
148/148 2s 12ms/step - accuracy: 0.8121 - loss: 0.4376 - val_accuracy: 0.7717 - val_loss: 0.5885
Epoch 209/600
148/148 2s 11ms/step - accuracy: 0.8133 - loss: 0.4375 - val_accuracy: 0.7720 - val_loss: 0.5916
Epoch 210/600
148/148 2s 12ms/step - accuracy: 0.8143 - loss: 0.4334 - val_accuracy: 0.7721 - val_loss: 0.5967
Epoch 211/600
148/148 2s 12ms/step - accuracy: 0.8143 - loss: 0.4342 - val_accuracy: 0.7726 - val_loss: 0.5773
Epoch 212/600
148/148 2s 13ms/step - accuracy: 0.8155 - loss: 0.4347 - val_accuracy: 0.7717 - val_loss: 0.5907
Epoch 213/600
148/148 2s 12ms/step - accuracy: 0.8182 - loss: 0.4296 - val_accuracy: 0.7741 - val_loss: 0.5845
Epoch 214/600
148/148 2s 13ms/step - accuracy: 0.8145 - loss: 0.4372 - val_accuracy: 0.7707 - val_loss: 0.5870
Epoch 215/600
148/148 2s 11ms/step - accuracy: 0.8166 - loss: 0.4332 - val_accuracy: 0.7700 - val_loss: 0.5919
Epoch 216/600
148/148 2s 10ms/step - accuracy: 0.8160 - loss: 0.4354 - val_accuracy: 0.7724 - val_loss: 0.5975
Epoch 217/600
148/148 2s 11ms/step - accuracy: 0.8143 - loss: 0.4351 - val_accuracy: 0.7710 - val_loss: 0.5887
Epoch 218/600
148/148 2s 11ms/step - accuracy: 0.8198 - loss: 0.4261 - val_accuracy: 0.7695 - val_loss: 0.5907
Epoch 219/600
148/148 2s 13ms/step - accuracy: 0.8178 - loss: 0.4271 - val_accuracy: 0.7715 - val_loss: 0.5872
Epoch 220/600
148/148 2s 12ms/step - accuracy: 0.8172 - loss: 0.4290 - val_accuracy: 0.7679 - val_loss: 0.5916
Epoch 221/600
148/148 2s 12ms/step - accuracy: 0.8146 - loss: 0.4335 - val_accuracy: 0.7726 - val_loss: 0.5863
Epoch 222/600
148/148 2s 11ms/step - accuracy: 0.8153 - loss: 0.4343 - val_accuracy: 0.7700 - val_loss: 0.5919
Epoch 223/600
148/148 2s 11ms/step - accuracy: 0.8189 - loss: 0.4269 - val_accuracy: 0.7686 - val_loss: 0.5892
Epoch 224/600
148/148 2s 11ms/step - accuracy: 0.8148 - loss: 0.4315 - val_accuracy: 0.7716 - val_loss: 0.5838
Epoch 225/600
148/148 2s 11ms/step - accuracy: 0.8161 - loss: 0.4283 - val_accuracy: 0.7724 - val_loss: 0.5823
Epoch 226/600
148/148 2s 11ms/step - accuracy: 0.8175 - loss: 0.4300 - val_accuracy: 0.7730 - val_loss: 0.5868
Epoch 227/600
148/148 2s 11ms/step - accuracy: 0.8162 - loss: 0.4306 - val_accuracy: 0.7715 - val_loss: 0.5907
Epoch 228/600
148/148 2s 12ms/step - accuracy: 0.8166 - loss: 0.4293 - val_accuracy: 0.7733 - val_loss: 0.5892

Epoch 229/600
148/148 2s 11ms/step - accuracy: 0.8156 - loss: 0.4293 - val_accuracy: 0.7702 - val_loss: 0.5834
Epoch 230/600
148/148 2s 12ms/step - accuracy: 0.8158 - loss: 0.4306 - val_accuracy: 0.7690 - val_loss: 0.5883
Epoch 231/600
148/148 2s 11ms/step - accuracy: 0.8156 - loss: 0.4275 - val_accuracy: 0.7706 - val_loss: 0.5928
s: 0.5928
Epoch 232/600
148/148 2s 11ms/step - accuracy: 0.8165 - loss: 0.4289 - val_accuracy: 0.7705 - val_loss: 0.5909
s: 0.5909
Epoch 233/600
148/148 2s 10ms/step - accuracy: 0.8180 - loss: 0.4295 - val_accuracy: 0.7695 - val_loss: 0.5875
s: 0.5875
Epoch 234/600
148/148 2s 11ms/step - accuracy: 0.8207 - loss: 0.4241 - val_accuracy: 0.7727 - val_loss: 0.5851
s: 0.5851
Epoch 235/600
148/148 2s 11ms/step - accuracy: 0.8207 - loss: 0.4226 - val_accuracy: 0.7707 - val_loss: 0.5880
s: 0.5880
Epoch 236/600
148/148 2s 12ms/step - accuracy: 0.8191 - loss: 0.4238 - val_accuracy: 0.7724 - val_loss: 0.5877
s: 0.5877
Epoch 237/600
148/148 2s 13ms/step - accuracy: 0.8171 - loss: 0.4326 - val_accuracy: 0.7722 - val_loss: 0.5824
s: 0.5824
Epoch 238/600
148/148 2s 12ms/step - accuracy: 0.8183 - loss: 0.4288 - val_accuracy: 0.7693 - val_loss: 0.5827
s: 0.5827
Epoch 239/600
148/148 2s 12ms/step - accuracy: 0.8181 - loss: 0.4256 - val_accuracy: 0.7723 - val_loss: 0.5903
s: 0.5903
Epoch 240/600
148/148 2s 12ms/step - accuracy: 0.8161 - loss: 0.4285 - val_accuracy: 0.7706 - val_loss: 0.5922
s: 0.5922
Epoch 241/600
148/148 2s 12ms/step - accuracy: 0.8203 - loss: 0.4233 - val_accuracy: 0.7719 - val_loss: 0.5850
s: 0.5850
Epoch 242/600
148/148 2s 11ms/step - accuracy: 0.8178 - loss: 0.4244 - val_accuracy: 0.7715 - val_loss: 0.5866
s: 0.5866
Epoch 243/600
148/148 2s 12ms/step - accuracy: 0.8196 - loss: 0.4253 - val_accuracy: 0.7735 - val_loss: 0.5921
s: 0.5921
Epoch 244/600
148/148 2s 11ms/step - accuracy: 0.8169 - loss: 0.4266 - val_accuracy: 0.7724 - val_loss: 0.5919
s: 0.5919
Epoch 245/600
148/148 2s 11ms/step - accuracy: 0.8194 - loss: 0.4236 - val_accuracy: 0.7716 - val_loss: 0.5870
s: 0.5870
Epoch 246/600
148/148 2s 11ms/step - accuracy: 0.8197 - loss: 0.4267 - val_accuracy: 0.7722 - val_loss: 0.5906
s: 0.5906
Epoch 247/600
148/148 2s 12ms/step - accuracy: 0.8215 - loss: 0.4233 - val_accuracy: 0.7700 - val_loss: 0.5846
s: 0.5846
Epoch 248/600
148/148 2s 12ms/step - accuracy: 0.8234 - loss: 0.4224 - val_accuracy: 0.7733 - val_loss: 0.5976
s: 0.5976
Epoch 249/600
148/148 2s 11ms/step - accuracy: 0.8196 - loss: 0.4235 - val_accuracy: 0.7704 - val_loss: 0.5877
s: 0.5877
Epoch 250/600
148/148 2s 11ms/step - accuracy: 0.8210 - loss: 0.4228 - val_accuracy: 0.7727 - val_loss: 0.5848
s: 0.5848
Epoch 251/600
148/148 2s 11ms/step - accuracy: 0.8212 - loss: 0.4207 - val_accuracy: 0.7712 - val_loss: 0.5842
s: 0.5842
Epoch 252/600
148/148 2s 11ms/step - accuracy: 0.8190 - loss: 0.4251 - val_accuracy: 0.7715 - val_loss: 0.5895
s: 0.5895
Epoch 253/600
148/148 2s 11ms/step - accuracy: 0.8216 - loss: 0.4194 - val_accuracy: 0.7712 - val_loss: 0.5939
s: 0.5939
Epoch 254/600

148/148 2s 11ms/step - accuracy: 0.8225 - loss: 0.4190 - val_accuracy: 0.7716 - val_loss: 0.5867
Epoch 255/600
148/148 2s 11ms/step - accuracy: 0.8233 - loss: 0.4216 - val_accuracy: 0.7748 - val_loss: 0.5895
Epoch 256/600
148/148 2s 11ms/step - accuracy: 0.8216 - loss: 0.4194 - val_accuracy: 0.7724 - val_loss: 0.5896
Epoch 257/600
148/148 2s 12ms/step - accuracy: 0.8242 - loss: 0.4153 - val_accuracy: 0.7731 - val_loss: 0.5923
Epoch 258/600
148/148 2s 12ms/step - accuracy: 0.8189 - loss: 0.4259 - val_accuracy: 0.7724 - val_loss: 0.5934
Epoch 259/600
148/148 2s 11ms/step - accuracy: 0.8220 - loss: 0.4210 - val_accuracy: 0.7736 - val_loss: 0.5830
Epoch 260/600
148/148 2s 11ms/step - accuracy: 0.8190 - loss: 0.4222 - val_accuracy: 0.7737 - val_loss: 0.5946
Epoch 261/600
148/148 2s 11ms/step - accuracy: 0.8212 - loss: 0.4217 - val_accuracy: 0.7733 - val_loss: 0.5939
Epoch 262/600
148/148 2s 11ms/step - accuracy: 0.8218 - loss: 0.4185 - val_accuracy: 0.7713 - val_loss: 0.5907
Epoch 263/600
148/148 2s 11ms/step - accuracy: 0.8216 - loss: 0.4144 - val_accuracy: 0.7714 - val_loss: 0.5951
Epoch 264/600
148/148 2s 11ms/step - accuracy: 0.8218 - loss: 0.4174 - val_accuracy: 0.7699 - val_loss: 0.5920
Epoch 265/600
148/148 2s 11ms/step - accuracy: 0.8219 - loss: 0.4187 - val_accuracy: 0.7705 - val_loss: 0.5886
Epoch 266/600
148/148 2s 12ms/step - accuracy: 0.8210 - loss: 0.4192 - val_accuracy: 0.7728 - val_loss: 0.5886
Epoch 267/600
148/148 2s 12ms/step - accuracy: 0.8223 - loss: 0.4173 - val_accuracy: 0.7731 - val_loss: 0.5906
Epoch 268/600
148/148 2s 11ms/step - accuracy: 0.8209 - loss: 0.4228 - val_accuracy: 0.7716 - val_loss: 0.5996
Epoch 269/600
148/148 2s 11ms/step - accuracy: 0.8206 - loss: 0.4234 - val_accuracy: 0.7712 - val_loss: 0.5966
Epoch 270/600
148/148 2s 11ms/step - accuracy: 0.8195 - loss: 0.4245 - val_accuracy: 0.7715 - val_loss: 0.5933
Epoch 271/600
148/148 2s 11ms/step - accuracy: 0.8266 - loss: 0.4156 - val_accuracy: 0.7720 - val_loss: 0.5939
Epoch 272/600
148/148 2s 11ms/step - accuracy: 0.8193 - loss: 0.4206 - val_accuracy: 0.7720 - val_loss: 0.5868
Epoch 273/600
148/148 2s 11ms/step - accuracy: 0.8223 - loss: 0.4171 - val_accuracy: 0.7726 - val_loss: 0.5861
Epoch 274/600
148/148 2s 11ms/step - accuracy: 0.8246 - loss: 0.4133 - val_accuracy: 0.7733 - val_loss: 0.5901
Epoch 275/600
148/148 2s 11ms/step - accuracy: 0.8254 - loss: 0.4149 - val_accuracy: 0.7716 - val_loss: 0.5935
Epoch 276/600
148/148 2s 12ms/step - accuracy: 0.8218 - loss: 0.4199 - val_accuracy: 0.7707 - val_loss: 0.5926
Epoch 277/600
148/148 2s 11ms/step - accuracy: 0.8229 - loss: 0.4165 - val_accuracy: 0.7714 - val_loss: 0.5821
Epoch 278/600
148/148 2s 11ms/step - accuracy: 0.8243 - loss: 0.4156 - val_accuracy: 0.7709 - val_loss: 0.5889
Epoch 279/600
148/148 2s 12ms/step - accuracy: 0.8218 - loss: 0.4165 - val_accuracy: 0.7711 - val_loss:

s: 0.5947
Epoch 280/600
148/148 3s 11ms/step - accuracy: 0.8228 - loss: 0.4175 - val_accuracy: 0.7720 - val_loss: 0.5928
s: 0.5928
Epoch 281/600
148/148 2s 12ms/step - accuracy: 0.8236 - loss: 0.4133 - val_accuracy: 0.7728 - val_loss: 0.5852
s: 0.5852
Epoch 282/600
148/148 2s 11ms/step - accuracy: 0.8219 - loss: 0.4184 - val_accuracy: 0.7724 - val_loss: 0.5892
s: 0.5892
Epoch 283/600
148/148 2s 11ms/step - accuracy: 0.8212 - loss: 0.4186 - val_accuracy: 0.7705 - val_loss: 0.5927
s: 0.5927
Epoch 284/600
148/148 2s 12ms/step - accuracy: 0.8220 - loss: 0.4174 - val_accuracy: 0.7701 - val_loss: 0.5907
s: 0.5907
Epoch 285/600
148/148 2s 12ms/step - accuracy: 0.8262 - loss: 0.4119 - val_accuracy: 0.7715 - val_loss: 0.5991
s: 0.5991
Epoch 286/600
148/148 2s 12ms/step - accuracy: 0.8250 - loss: 0.4154 - val_accuracy: 0.7720 - val_loss: 0.5903
s: 0.5903
Epoch 287/600
148/148 2s 11ms/step - accuracy: 0.8228 - loss: 0.4174 - val_accuracy: 0.7701 - val_loss: 0.5876
s: 0.5876
Epoch 288/600
148/148 2s 11ms/step - accuracy: 0.8235 - loss: 0.4164 - val_accuracy: 0.7734 - val_loss: 0.5932
s: 0.5932
Epoch 289/600
148/148 2s 11ms/step - accuracy: 0.8276 - loss: 0.4085 - val_accuracy: 0.7721 - val_loss: 0.5929
s: 0.5929
Epoch 290/600
148/148 2s 12ms/step - accuracy: 0.8255 - loss: 0.4153 - val_accuracy: 0.7733 - val_loss: 0.5881
s: 0.5881
Epoch 291/600
148/148 2s 11ms/step - accuracy: 0.8235 - loss: 0.4162 - val_accuracy: 0.7713 - val_loss: 0.5993
s: 0.5993
Epoch 292/600
148/148 2s 11ms/step - accuracy: 0.8244 - loss: 0.4152 - val_accuracy: 0.7711 - val_loss: 0.5983
s: 0.5983
Epoch 293/600
148/148 2s 11ms/step - accuracy: 0.8247 - loss: 0.4143 - val_accuracy: 0.7717 - val_loss: 0.5889
s: 0.5889
Epoch 294/600
148/148 2s 12ms/step - accuracy: 0.8260 - loss: 0.4104 - val_accuracy: 0.7744 - val_loss: 0.5939
s: 0.5939
Epoch 295/600
148/148 2s 11ms/step - accuracy: 0.8254 - loss: 0.4110 - val_accuracy: 0.7710 - val_loss: 0.5955
s: 0.5955
Epoch 296/600
148/148 2s 11ms/step - accuracy: 0.8216 - loss: 0.4137 - val_accuracy: 0.7729 - val_loss: 0.5945
s: 0.5945
Epoch 297/600
148/148 2s 11ms/step - accuracy: 0.8239 - loss: 0.4095 - val_accuracy: 0.7721 - val_loss: 0.5960
s: 0.5960
Epoch 298/600
148/148 2s 11ms/step - accuracy: 0.8227 - loss: 0.4177 - val_accuracy: 0.7723 - val_loss: 0.5964
s: 0.5964
Epoch 299/600
148/148 2s 11ms/step - accuracy: 0.8256 - loss: 0.4110 - val_accuracy: 0.7702 - val_loss: 0.5916
s: 0.5916
Epoch 300/600
148/148 2s 11ms/step - accuracy: 0.8243 - loss: 0.4129 - val_accuracy: 0.7738 - val_loss: 0.5890
s: 0.5890
Epoch 301/600
148/148 2s 11ms/step - accuracy: 0.8254 - loss: 0.4110 - val_accuracy: 0.7735 - val_loss: 0.5907
s: 0.5907
Epoch 302/600
148/148 2s 11ms/step - accuracy: 0.8289 - loss: 0.4077 - val_accuracy: 0.7708 - val_loss: 0.5973
s: 0.5973
Epoch 303/600
148/148 2s 11ms/step - accuracy: 0.8251 - loss: 0.4132 - val_accuracy: 0.7709 - val_loss: 0.5955
s: 0.5955
Epoch 304/600
148/148 2s 13ms/step - accuracy: 0.8252 - loss: 0.4142 - val_accuracy: 0.7734 - val_loss: 0.6006
s: 0.6006

Epoch 305/600
148/148 2s 12ms/step - accuracy: 0.8270 - loss: 0.4096 - val_accuracy: 0.7732 - val_loss: 0.5975
Epoch 306/600
148/148 2s 11ms/step - accuracy: 0.8251 - loss: 0.4149 - val_accuracy: 0.7709 - val_loss: 0.5960
Epoch 307/600
148/148 2s 11ms/step - accuracy: 0.8277 - loss: 0.4097 - val_accuracy: 0.7738 - val_loss: 0.5916
s: 0.5973
Epoch 308/600
148/148 2s 11ms/step - accuracy: 0.8272 - loss: 0.4092 - val_accuracy: 0.7734 - val_loss: 0.5991
s: 0.5973
Epoch 309/600
148/148 2s 11ms/step - accuracy: 0.8276 - loss: 0.4076 - val_accuracy: 0.7723 - val_loss: 0.6111
s: 0.5960
Epoch 310/600
148/148 2s 11ms/step - accuracy: 0.8241 - loss: 0.4114 - val_accuracy: 0.7734 - val_loss: 0.6015
s: 0.5956
Epoch 311/600
148/148 2s 11ms/step - accuracy: 0.8258 - loss: 0.4104 - val_accuracy: 0.7759 - val_loss: 0.6015
s: 0.6015
Epoch 312/600
148/148 2s 12ms/step - accuracy: 0.8271 - loss: 0.4077 - val_accuracy: 0.7745 - val_loss: 0.5956
s: 0.5960
Epoch 313/600
148/148 2s 12ms/step - accuracy: 0.8232 - loss: 0.4137 - val_accuracy: 0.7722 - val_loss: 0.5956
s: 0.5956
Epoch 314/600
148/148 3s 11ms/step - accuracy: 0.8276 - loss: 0.4077 - val_accuracy: 0.7723 - val_loss: 0.5960
s: 0.5960
Epoch 315/600
148/148 2s 11ms/step - accuracy: 0.8254 - loss: 0.4122 - val_accuracy: 0.7759 - val_loss: 0.5824
s: 0.5824
Epoch 316/600
148/148 2s 11ms/step - accuracy: 0.8254 - loss: 0.4087 - val_accuracy: 0.7748 - val_loss: 0.5879
s: 0.5879
Epoch 317/600
148/148 2s 11ms/step - accuracy: 0.8217 - loss: 0.4164 - val_accuracy: 0.7755 - val_loss: 0.5982
s: 0.5982
Epoch 318/600
148/148 2s 11ms/step - accuracy: 0.8284 - loss: 0.4062 - val_accuracy: 0.7737 - val_loss: 0.5905
s: 0.5905
Epoch 319/600
148/148 2s 11ms/step - accuracy: 0.8268 - loss: 0.4115 - val_accuracy: 0.7725 - val_loss: 0.5974
s: 0.5974
Epoch 320/600
148/148 2s 11ms/step - accuracy: 0.8282 - loss: 0.4103 - val_accuracy: 0.7727 - val_loss: 0.6001
s: 0.6001
Epoch 321/600
148/148 2s 12ms/step - accuracy: 0.8299 - loss: 0.4089 - val_accuracy: 0.7727 - val_loss: 0.5936
s: 0.5936
Epoch 322/600
148/148 2s 12ms/step - accuracy: 0.8287 - loss: 0.4063 - val_accuracy: 0.7741 - val_loss: 0.5962
s: 0.5962
Epoch 323/600
148/148 2s 12ms/step - accuracy: 0.8277 - loss: 0.4078 - val_accuracy: 0.7726 - val_loss: 0.5989
s: 0.5989
Epoch 324/600
148/148 2s 11ms/step - accuracy: 0.8277 - loss: 0.4074 - val_accuracy: 0.7737 - val_loss: 0.6019
s: 0.6019
Epoch 325/600
148/148 2s 11ms/step - accuracy: 0.8284 - loss: 0.4048 - val_accuracy: 0.7725 - val_loss: 0.6009
s: 0.6009
Epoch 326/600
148/148 2s 11ms/step - accuracy: 0.8289 - loss: 0.4007 - val_accuracy: 0.7719 - val_loss: 0.5974
s: 0.5974
Epoch 327/600
148/148 2s 11ms/step - accuracy: 0.8260 - loss: 0.4088 - val_accuracy: 0.7721 - val_loss: 0.5925
s: 0.5925
Epoch 328/600
148/148 2s 11ms/step - accuracy: 0.8262 - loss: 0.4079 - val_accuracy: 0.7722 - val_loss: 0.5987
s: 0.5987
Epoch 329/600
148/148 2s 11ms/step - accuracy: 0.8290 - loss: 0.4055 - val_accuracy: 0.7731 - val_loss: 0.5838
s: 0.5838
Epoch 330/600

148/148 2s 11ms/step - accuracy: 0.8300 - loss: 0.4028 - val_accuracy: 0.7725 - val_loss: 0.5969
Epoch 331/600
148/148 2s 11ms/step - accuracy: 0.8279 - loss: 0.4057 - val_accuracy: 0.7754 - val_loss: 0.5971
Epoch 332/600
148/148 2s 12ms/step - accuracy: 0.8291 - loss: 0.4035 - val_accuracy: 0.7713 - val_loss: 0.6045
Epoch 333/600
148/148 2s 12ms/step - accuracy: 0.8255 - loss: 0.4073 - val_accuracy: 0.7746 - val_loss: 0.5982
Epoch 334/600
148/148 2s 11ms/step - accuracy: 0.8277 - loss: 0.4062 - val_accuracy: 0.7738 - val_loss: 0.6010
Epoch 335/600
148/148 2s 11ms/step - accuracy: 0.8277 - loss: 0.4052 - val_accuracy: 0.7732 - val_loss: 0.6042
Epoch 336/600
148/148 2s 11ms/step - accuracy: 0.8295 - loss: 0.4011 - val_accuracy: 0.7741 - val_loss: 0.5983
Epoch 337/600
148/148 2s 11ms/step - accuracy: 0.8267 - loss: 0.4046 - val_accuracy: 0.7753 - val_loss: 0.5854
Epoch 338/600
148/148 2s 11ms/step - accuracy: 0.8321 - loss: 0.4032 - val_accuracy: 0.7725 - val_loss: 0.6022
Epoch 339/600
148/148 2s 11ms/step - accuracy: 0.8307 - loss: 0.4042 - val_accuracy: 0.7747 - val_loss: 0.6045
Epoch 340/600
148/148 2s 11ms/step - accuracy: 0.8274 - loss: 0.4093 - val_accuracy: 0.7737 - val_loss: 0.5895
Epoch 341/600
148/148 2s 12ms/step - accuracy: 0.8308 - loss: 0.4013 - val_accuracy: 0.7731 - val_loss: 0.5906
Epoch 342/600
148/148 2s 12ms/step - accuracy: 0.8287 - loss: 0.4044 - val_accuracy: 0.7728 - val_loss: 0.5989
Epoch 343/600
148/148 2s 12ms/step - accuracy: 0.8319 - loss: 0.4006 - val_accuracy: 0.7736 - val_loss: 0.5938
Epoch 344/600
148/148 2s 11ms/step - accuracy: 0.8299 - loss: 0.4003 - val_accuracy: 0.7726 - val_loss: 0.6003
Epoch 345/600
148/148 2s 11ms/step - accuracy: 0.8301 - loss: 0.4057 - val_accuracy: 0.7742 - val_loss: 0.5921
Epoch 346/600
148/148 2s 11ms/step - accuracy: 0.8305 - loss: 0.4060 - val_accuracy: 0.7749 - val_loss: 0.5872
Epoch 347/600
148/148 2s 11ms/step - accuracy: 0.8290 - loss: 0.4043 - val_accuracy: 0.7750 - val_loss: 0.5941
Epoch 348/600
148/148 2s 11ms/step - accuracy: 0.8296 - loss: 0.3999 - val_accuracy: 0.7722 - val_loss: 0.5955
Epoch 349/600
148/148 2s 11ms/step - accuracy: 0.8310 - loss: 0.4019 - val_accuracy: 0.7724 - val_loss: 0.5901
Epoch 350/600
148/148 2s 11ms/step - accuracy: 0.8288 - loss: 0.4035 - val_accuracy: 0.7738 - val_loss: 0.5891
Epoch 351/600
148/148 2s 12ms/step - accuracy: 0.8334 - loss: 0.4007 - val_accuracy: 0.7729 - val_loss: 0.5954
Epoch 352/600
148/148 2s 12ms/step - accuracy: 0.8289 - loss: 0.4057 - val_accuracy: 0.7744 - val_loss: 0.6052
Epoch 353/600
148/148 2s 11ms/step - accuracy: 0.8320 - loss: 0.4019 - val_accuracy: 0.7726 - val_loss: 0.5978
Epoch 354/600
148/148 2s 11ms/step - accuracy: 0.8262 - loss: 0.4081 - val_accuracy: 0.7746 - val_loss: 0.5955
Epoch 355/600
148/148 3s 11ms/step - accuracy: 0.8312 - loss: 0.3972 - val_accuracy: 0.7770 - val_loss:

s: 0.5953
Epoch 356/600
148/148 2s 11ms/step - accuracy: 0.8320 - loss: 0.3976 - val_accuracy: 0.7746 - val_loss: 0.5930
Epoch 357/600
148/148 2s 11ms/step - accuracy: 0.8315 - loss: 0.3980 - val_accuracy: 0.7759 - val_loss: 0.5877
Epoch 358/600
148/148 2s 11ms/step - accuracy: 0.8339 - loss: 0.4001 - val_accuracy: 0.7760 - val_loss: 0.5877
s: 0.5883
Epoch 359/600
148/148 2s 11ms/step - accuracy: 0.8302 - loss: 0.4030 - val_accuracy: 0.7752 - val_loss: 0.6016
s: 0.5895
Epoch 360/600
148/148 2s 12ms/step - accuracy: 0.8270 - loss: 0.4043 - val_accuracy: 0.7765 - val_loss: 0.5953
s: 0.5962
Epoch 361/600
148/148 2s 11ms/step - accuracy: 0.8329 - loss: 0.3985 - val_accuracy: 0.7762 - val_loss: 0.5937
s: 0.6016
Epoch 362/600
148/148 2s 12ms/step - accuracy: 0.8291 - loss: 0.4052 - val_accuracy: 0.7745 - val_loss: 0.5962
s: 0.6045
Epoch 363/600
148/148 2s 11ms/step - accuracy: 0.8316 - loss: 0.4024 - val_accuracy: 0.7754 - val_loss: 0.5986
s: 0.6016
Epoch 364/600
148/148 2s 11ms/step - accuracy: 0.8285 - loss: 0.4049 - val_accuracy: 0.7753 - val_loss: 0.5937
s: 0.6045
Epoch 365/600
148/148 2s 11ms/step - accuracy: 0.8271 - loss: 0.4072 - val_accuracy: 0.7743 - val_loss: 0.5962
s: 0.6016
Epoch 366/600
148/148 2s 11ms/step - accuracy: 0.8302 - loss: 0.3989 - val_accuracy: 0.7771 - val_loss: 0.5937
s: 0.6045
Epoch 367/600
148/148 2s 11ms/step - accuracy: 0.8320 - loss: 0.3954 - val_accuracy: 0.7765 - val_loss: 0.5983
s: 0.6016
Epoch 368/600
148/148 2s 11ms/step - accuracy: 0.8278 - loss: 0.4034 - val_accuracy: 0.7748 - val_loss: 0.5963
s: 0.6045
Epoch 369/600
148/148 2s 13ms/step - accuracy: 0.8300 - loss: 0.4048 - val_accuracy: 0.7729 - val_loss: 0.5919
s: 0.6016
Epoch 370/600
148/148 2s 12ms/step - accuracy: 0.8315 - loss: 0.3982 - val_accuracy: 0.7756 - val_loss: 0.5983
s: 0.6045
Epoch 371/600
148/148 2s 11ms/step - accuracy: 0.8306 - loss: 0.4036 - val_accuracy: 0.7746 - val_loss: 0.5914
s: 0.6016
Epoch 372/600
148/148 2s 11ms/step - accuracy: 0.8294 - loss: 0.3985 - val_accuracy: 0.7715 - val_loss: 0.5985
s: 0.6041
Epoch 373/600
148/148 2s 11ms/step - accuracy: 0.8343 - loss: 0.3956 - val_accuracy: 0.7727 - val_loss: 0.5970
s: 0.6016
Epoch 374/600
148/148 2s 11ms/step - accuracy: 0.8322 - loss: 0.3962 - val_accuracy: 0.7768 - val_loss: 0.5914
s: 0.6045
Epoch 375/600
148/148 2s 11ms/step - accuracy: 0.8319 - loss: 0.3975 - val_accuracy: 0.7743 - val_loss: 0.5985
s: 0.6016
Epoch 376/600
148/148 2s 11ms/step - accuracy: 0.8314 - loss: 0.3976 - val_accuracy: 0.7732 - val_loss: 0.5982
s: 0.6045
Epoch 377/600
148/148 2s 11ms/step - accuracy: 0.8353 - loss: 0.3934 - val_accuracy: 0.7744 - val_loss: 0.5996
s: 0.6016
Epoch 378/600
148/148 2s 11ms/step - accuracy: 0.8328 - loss: 0.3992 - val_accuracy: 0.7743 - val_loss: 0.5988
s: 0.6015
Epoch 379/600
148/148 2s 12ms/step - accuracy: 0.8332 - loss: 0.4007 - val_accuracy: 0.7755 - val_loss: 0.6015
s: 0.6015

Epoch 381/600
148/148 3s 11ms/step - accuracy: 0.8325 - loss: 0.3966 - val_accuracy: 0.7747 - val_loss: 0.6049
Epoch 382/600
148/148 2s 11ms/step - accuracy: 0.8296 - loss: 0.3984 - val_accuracy: 0.7779 - val_loss: 0.6057
Epoch 383/600
148/148 2s 11ms/step - accuracy: 0.8315 - loss: 0.3998 - val_accuracy: 0.7745 - val_loss: 0.6063
Epoch 384/600
148/148 2s 11ms/step - accuracy: 0.8289 - loss: 0.4034 - val_accuracy: 0.7761 - val_loss: 0.6058
Epoch 385/600
148/148 2s 11ms/step - accuracy: 0.8300 - loss: 0.4005 - val_accuracy: 0.7760 - val_loss: 0.5995
Epoch 386/600
148/148 2s 12ms/step - accuracy: 0.8338 - loss: 0.3955 - val_accuracy: 0.7750 - val_loss: 0.6007
Epoch 387/600
148/148 2s 11ms/step - accuracy: 0.8292 - loss: 0.4006 - val_accuracy: 0.7754 - val_loss: 0.5889
Epoch 388/600
148/148 2s 12ms/step - accuracy: 0.8318 - loss: 0.3971 - val_accuracy: 0.7749 - val_loss: 0.5987
Epoch 389/600
148/148 2s 12ms/step - accuracy: 0.8356 - loss: 0.3916 - val_accuracy: 0.7756 - val_loss: 0.5917
Epoch 390/600
148/148 2s 12ms/step - accuracy: 0.8326 - loss: 0.3959 - val_accuracy: 0.7766 - val_loss: 0.6043
Epoch 391/600
148/148 2s 12ms/step - accuracy: 0.8296 - loss: 0.4001 - val_accuracy: 0.7759 - val_loss: 0.6010
Epoch 392/600
148/148 2s 11ms/step - accuracy: 0.8322 - loss: 0.3972 - val_accuracy: 0.7755 - val_loss: 0.5957
Epoch 393/600
148/148 2s 11ms/step - accuracy: 0.8321 - loss: 0.4000 - val_accuracy: 0.7749 - val_loss: 0.5918
Epoch 394/600
148/148 2s 11ms/step - accuracy: 0.8327 - loss: 0.3935 - val_accuracy: 0.7731 - val_loss: 0.6021
Epoch 395/600
148/148 2s 11ms/step - accuracy: 0.8315 - loss: 0.3987 - val_accuracy: 0.7763 - val_loss: 0.5995
Epoch 396/600
148/148 2s 11ms/step - accuracy: 0.8345 - loss: 0.3913 - val_accuracy: 0.7749 - val_loss: 0.5947
Epoch 397/600
148/148 2s 11ms/step - accuracy: 0.8335 - loss: 0.3947 - val_accuracy: 0.7749 - val_loss: 0.5945
Epoch 398/600
148/148 2s 11ms/step - accuracy: 0.8327 - loss: 0.3955 - val_accuracy: 0.7726 - val_loss: 0.5939
Epoch 399/600
148/148 2s 11ms/step - accuracy: 0.8324 - loss: 0.3965 - val_accuracy: 0.7750 - val_loss: 0.5967
Epoch 400/600
148/148 2s 11ms/step - accuracy: 0.8325 - loss: 0.3965 - val_accuracy: 0.7756 - val_loss: 0.5982
Epoch 401/600
148/148 2s 11ms/step - accuracy: 0.8312 - loss: 0.3972 - val_accuracy: 0.7767 - val_loss: 0.5960
Epoch 402/600
148/148 2s 11ms/step - accuracy: 0.8325 - loss: 0.3992 - val_accuracy: 0.7771 - val_loss: 0.5984
Epoch 403/600
148/148 2s 13ms/step - accuracy: 0.8357 - loss: 0.3921 - val_accuracy: 0.7752 - val_loss: 0.6007
Epoch 404/600
148/148 2s 10ms/step - accuracy: 0.8360 - loss: 0.3887 - val_accuracy: 0.7725 - val_loss: 0.5944
Epoch 405/600
148/148 2s 11ms/step - accuracy: 0.8318 - loss: 0.3980 - val_accuracy: 0.7735 - val_loss: 0.6050
Epoch 406/600

148/148 2s 11ms/step - accuracy: 0.8321 - loss: 0.3973 - val_accuracy: 0.7724 - val_loss: 0.5989
Epoch 407/600
148/148 2s 13ms/step - accuracy: 0.8322 - loss: 0.3960 - val_accuracy: 0.7713 - val_loss: 0.6080
Epoch 408/600
148/148 2s 12ms/step - accuracy: 0.8347 - loss: 0.3910 - val_accuracy: 0.7741 - val_loss: 0.5900
Epoch 409/600
148/148 2s 11ms/step - accuracy: 0.8306 - loss: 0.4003 - val_accuracy: 0.7753 - val_loss: 0.6049
Epoch 410/600
148/148 2s 12ms/step - accuracy: 0.8354 - loss: 0.3916 - val_accuracy: 0.7756 - val_loss: 0.5954
Epoch 411/600
148/148 2s 11ms/step - accuracy: 0.8383 - loss: 0.3927 - val_accuracy: 0.7751 - val_loss: 0.6002
Epoch 412/600
148/148 2s 11ms/step - accuracy: 0.8323 - loss: 0.3972 - val_accuracy: 0.7740 - val_loss: 0.5878
Epoch 413/600
148/148 2s 10ms/step - accuracy: 0.8347 - loss: 0.3933 - val_accuracy: 0.7765 - val_loss: 0.6046
Epoch 414/600
148/148 2s 11ms/step - accuracy: 0.8342 - loss: 0.3951 - val_accuracy: 0.7743 - val_loss: 0.5992
Epoch 415/600
148/148 2s 11ms/step - accuracy: 0.8323 - loss: 0.3988 - val_accuracy: 0.7748 - val_loss: 0.6070
Epoch 416/600
148/148 2s 12ms/step - accuracy: 0.8286 - loss: 0.4001 - val_accuracy: 0.7733 - val_loss: 0.6002
Epoch 417/600
148/148 2s 13ms/step - accuracy: 0.8329 - loss: 0.3915 - val_accuracy: 0.7754 - val_loss: 0.5986
Epoch 418/600
148/148 2s 12ms/step - accuracy: 0.8354 - loss: 0.3924 - val_accuracy: 0.7750 - val_loss: 0.6030
Epoch 419/600
148/148 2s 11ms/step - accuracy: 0.8342 - loss: 0.3926 - val_accuracy: 0.7740 - val_loss: 0.6050
Epoch 420/600
148/148 2s 11ms/step - accuracy: 0.8343 - loss: 0.3910 - val_accuracy: 0.7760 - val_loss: 0.5975
Epoch 421/600
148/148 2s 11ms/step - accuracy: 0.8322 - loss: 0.3906 - val_accuracy: 0.7743 - val_loss: 0.5996
Epoch 422/600
148/148 2s 11ms/step - accuracy: 0.8325 - loss: 0.3943 - val_accuracy: 0.7752 - val_loss: 0.5932
Epoch 423/600
148/148 2s 11ms/step - accuracy: 0.8321 - loss: 0.3956 - val_accuracy: 0.7754 - val_loss: 0.5929
Epoch 424/600
148/148 2s 11ms/step - accuracy: 0.8335 - loss: 0.3955 - val_accuracy: 0.7736 - val_loss: 0.6065
Epoch 425/600
148/148 2s 11ms/step - accuracy: 0.8354 - loss: 0.3932 - val_accuracy: 0.7724 - val_loss: 0.6012
Epoch 426/600
148/148 2s 11ms/step - accuracy: 0.8314 - loss: 0.3966 - val_accuracy: 0.7746 - val_loss: 0.5987
Epoch 427/600
148/148 2s 11ms/step - accuracy: 0.8313 - loss: 0.3986 - val_accuracy: 0.7753 - val_loss: 0.6007
Epoch 428/600
148/148 2s 11ms/step - accuracy: 0.8359 - loss: 0.3864 - val_accuracy: 0.7738 - val_loss: 0.6008
Epoch 429/600
148/148 2s 11ms/step - accuracy: 0.8329 - loss: 0.3919 - val_accuracy: 0.7761 - val_loss: 0.6022
Epoch 430/600
148/148 2s 10ms/step - accuracy: 0.8355 - loss: 0.3910 - val_accuracy: 0.7744 - val_loss: 0.6044
Epoch 431/600
148/148 2s 11ms/step - accuracy: 0.8343 - loss: 0.3922 - val_accuracy: 0.7754 - val_loss:

```
s: 0.5975
Epoch 432/600
148/148 2s 10ms/step - accuracy: 0.8328 - loss: 0.3934 - val_accuracy: 0.7759 - val_loss
s: 0.6016
Epoch 433/600
148/148 2s 11ms/step - accuracy: 0.8361 - loss: 0.3917 - val_accuracy: 0.7747 - val_loss
s: 0.6089
Epoch 434/600
148/148 2s 14ms/step - accuracy: 0.8367 - loss: 0.3917 - val_accuracy: 0.7754 - val_loss
s: 0.6012
Epoch 435/600
148/148 2s 13ms/step - accuracy: 0.8331 - loss: 0.3946 - val_accuracy: 0.7745 - val_loss
s: 0.6064
Epoch 436/600
148/148 2s 12ms/step - accuracy: 0.8366 - loss: 0.3895 - val_accuracy: 0.7725 - val_loss
s: 0.5954
Epoch 437/600
148/148 2s 11ms/step - accuracy: 0.8356 - loss: 0.3890 - val_accuracy: 0.7734 - val_loss
s: 0.6056
Epoch 438/600
148/148 2s 11ms/step - accuracy: 0.8334 - loss: 0.3933 - val_accuracy: 0.7740 - val_loss
s: 0.6028
Epoch 439/600
148/148 2s 11ms/step - accuracy: 0.8343 - loss: 0.3948 - val_accuracy: 0.7736 - val_loss
s: 0.6114
Epoch 440/600
148/148 2s 12ms/step - accuracy: 0.8359 - loss: 0.3913 - val_accuracy: 0.7728 - val_loss
s: 0.6107
Epoch 441/600
148/148 2s 11ms/step - accuracy: 0.8344 - loss: 0.3958 - val_accuracy: 0.7740 - val_loss
s: 0.5942
Epoch 442/600
148/148 2s 11ms/step - accuracy: 0.8351 - loss: 0.3901 - val_accuracy: 0.7742 - val_loss
s: 0.5986
```

```
In [264... ann5_smote.evaluate(X_train_smote, y_train_smote)
```

```
3770/3770 5s 1ms/step - accuracy: 0.9067 - loss: 0.2351
```

```
Out[264... [0.23078903555870056, 0.9180200099945068]
```

```
In [265... ann5_smote.summary()
```

```
Model: "sequential_9"
```

Layer (type)	Output Shape	Param #
dense_57 (Dense)	(None, 512)	23,040
batch_normalization_6 (BatchNormalization)	(None, 512)	2,048
dropout_48 (Dropout)	(None, 512)	0
dense_58 (Dense)	(None, 256)	131,328
batch_normalization_7 (BatchNormalization)	(None, 256)	1,024
dropout_49 (Dropout)	(None, 256)	0
dense_59 (Dense)	(None, 256)	65,792
batch_normalization_8 (BatchNormalization)	(None, 256)	1,024
dropout_50 (Dropout)	(None, 256)	0
dense_60 (Dense)	(None, 128)	32,896
batch_normalization_9 (BatchNormalization)	(None, 128)	512
dropout_51 (Dropout)	(None, 128)	0
dense_61 (Dense)	(None, 128)	16,512
batch_normalization_10 (BatchNormalization)	(None, 128)	512
dropout_52 (Dropout)	(None, 128)	0
dense_62 (Dense)	(None, 64)	8,256
batch_normalization_11 (BatchNormalization)	(None, 64)	256
dropout_53 (Dropout)	(None, 64)	0
dense_63 (Dense)	(None, 3)	195

Total params: 844,811 (3.22 MB)

Trainable params: 280,707 (1.07 MB)

Non-trainable params: 2,688 (10.50 KB)

Optimizer params: 561,416 (2.14 MB)

In [247]: eval_metric(ann5_smote, X_train_smote, y_train_smote, X_val, y_val)

3770/3770 ————— 4s 1ms/step
 591/591 ————— 1s 958us/step

Test Set:

```
[[4784 512 212]
 [1954 6358 1741]
 [ 91 362 2889]]
```

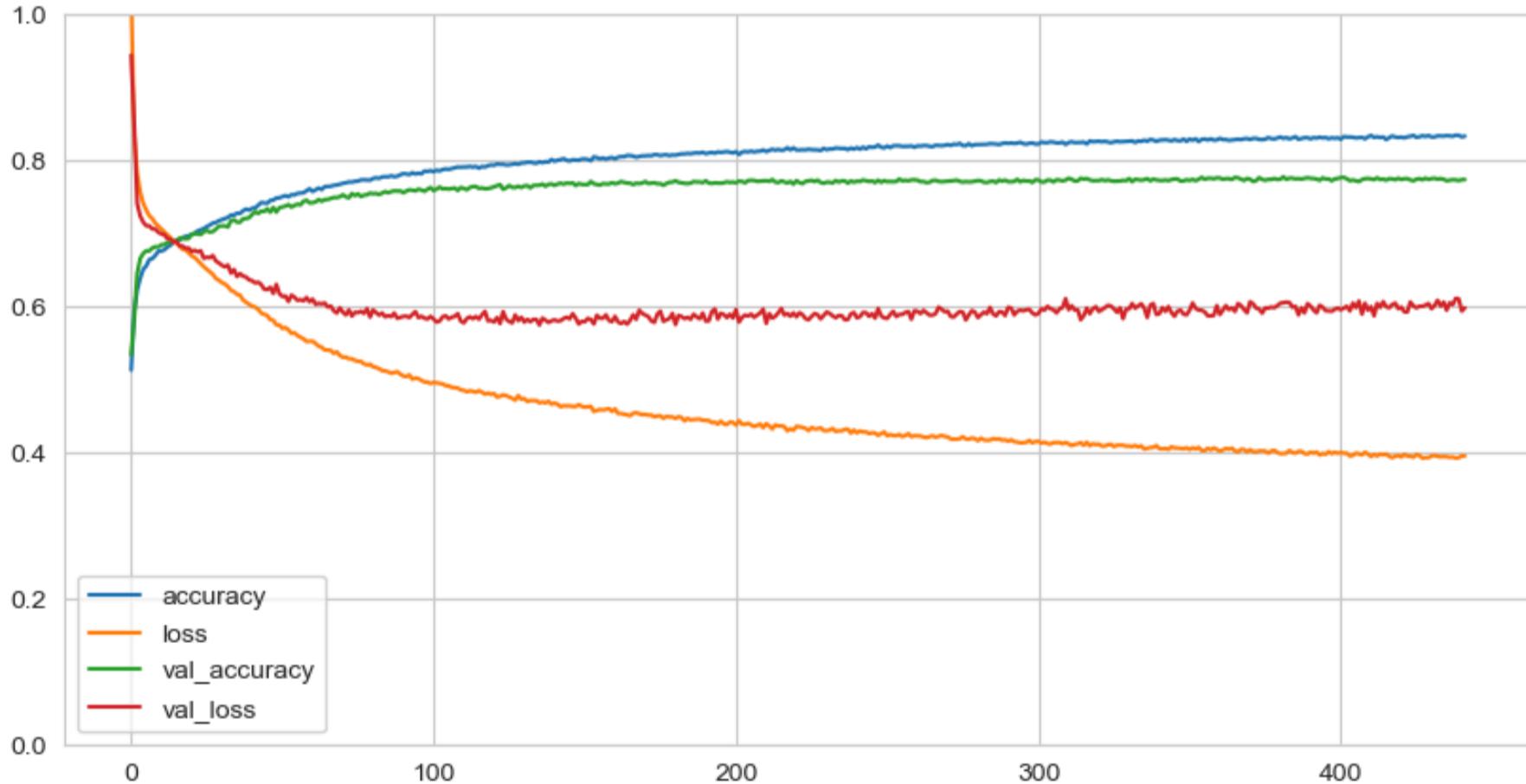
	precision	recall	f1-score	support
0	0.70	0.87	0.78	5508
1	0.88	0.63	0.74	10053
2	0.60	0.86	0.71	3342
accuracy			0.74	18903
macro avg	0.73	0.79	0.74	18903
weighted avg	0.78	0.74	0.74	18903

Train Set:

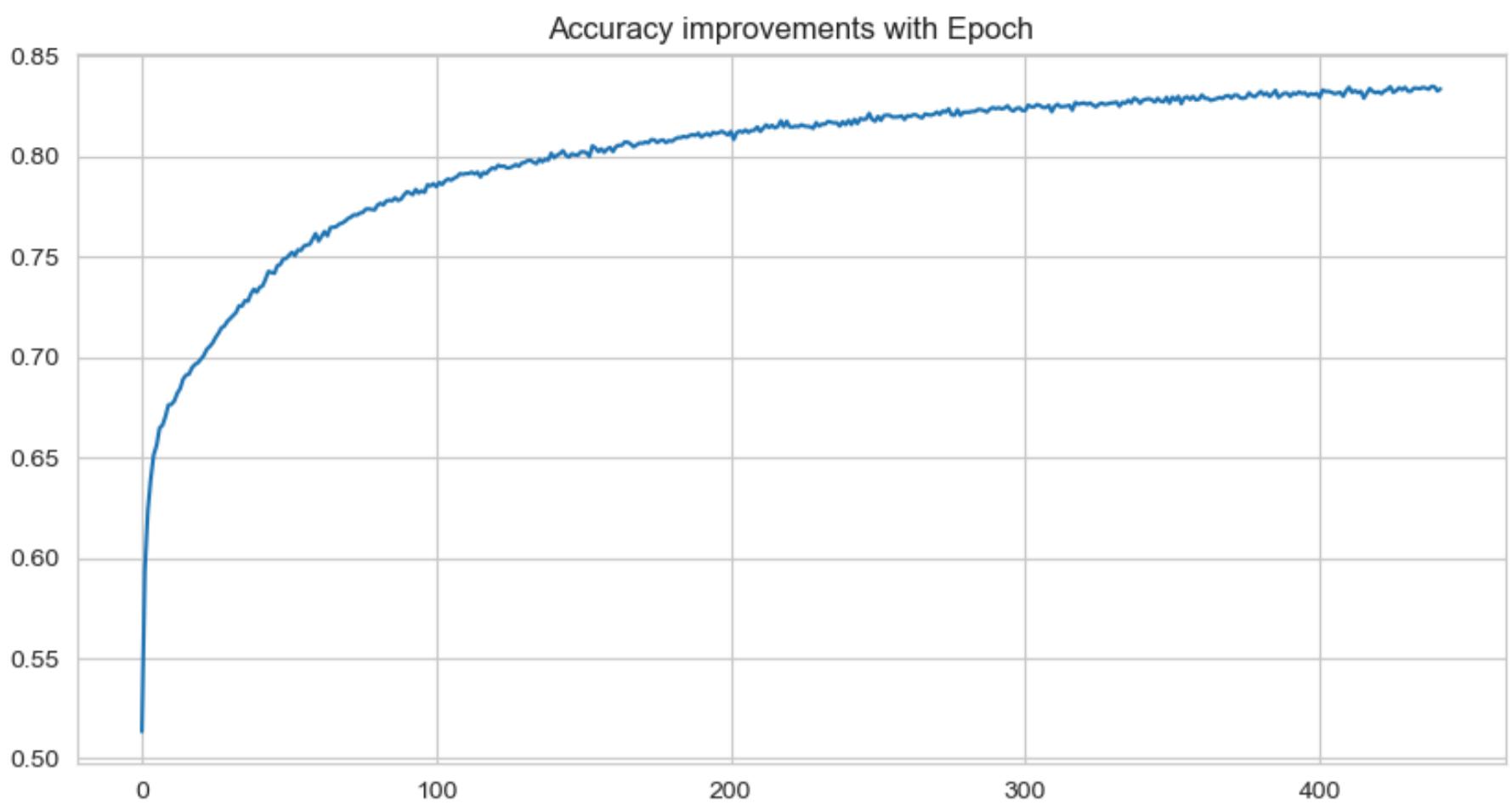
```
[[39629 384 196]
 [ 5751 29113 5345]
 [ 1 19 40189]]
```

	precision	recall	f1-score	support
0	0.87	0.99	0.93	40209
1	0.99	0.72	0.84	40209
2	0.88	1.00	0.94	40209
accuracy			0.90	120627
macro avg	0.91	0.90	0.90	120627
weighted avg	0.91	0.90	0.90	120627

```
In [266...]: pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



```
In [267...]: pd.DataFrame(history.history)[“accuracy”].plot(figsize=(10, 5))
plt.title("Accuracy improvements with Epoch")
plt.show()
```



```
In [ ]: # Save the Model

from tensorflow.keras.models import load_model

# Save the model to 'model.h5' file
ann5_smote.save('ann5_smote_model.h5')

# To Load the model later for further use
loaded_ann5_smote = load_model('ann5_smote_model.h5')
```

ANN-5 Model with SMOTE Summary:

- **(Dense) layers: 7 / Neurons:** 512-256-256-128-128-64 / **Dropout:** 30-30-25-25-20-20% / **Batch Normalization:** Added after each layer / **Learning Rate:** 0.001 / **Batch Size:** 512 / **Epochs:** 600 / **Early Stop (val_accuracy):** 60
- **Accuracy:** 0.9180 / **Val_Accuracy:** 0.7400 / **Loss:** 0.2308 / **Val_Loss:** 0.7300
- **Train Recall (Class 2):** 1.00 / **Test Recall (Class 2):** 0.86

Improvements:

- High training accuracy and exceptional recall for class 2, showing enhanced capability in recognizing the minority class due to SMOTE and batch normalization.
- Stability in validation accuracy with advanced normalization techniques helping mitigate overfitting.

No Improvement:

- While training performance is high, validation metrics did not improve significantly, indicating challenges in generalizing the learned patterns.

Got Worse:

- Validation loss remains elevated, suggesting that despite improvements, the model may still be overfitting, possibly due to the increased model complexity. Adjustments in regularization strategies might be required.

ANN-6 Model + Regularization (%79)

```
In [33]: # Fully connected (Dense) --> 7 Layers
# kernel_regularizer=l2(0.01) added

# 1) Model Architecture:
ann6 = Sequential([
    Dense(512, input_dim=X_train.shape[1], activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    Dropout(0.3),
    Dense(256, activation='relu'),
    BatchNormalization(),
```

```
Dropout(0.3),
Dense(256, activation='relu'),
BatchNormalization(),
Dropout(0.25),
Dense(128, activation='relu'),
BatchNormalization(),
Dropout(0.25),
Dense(128, activation='relu'),
BatchNormalization(),
Dropout(0.2),
Dense(64, activation='relu'),
BatchNormalization(),
Dropout(0.2),
Dense(3, activation='softmax')
])

# 2) Compiling the Model:
ann6.compile(optimizer=Adam(learning_rate=0.001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 3) Early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=60,
                               mode="auto",
                               restore_best_weights=True)

# 4) Train the model
history = ann6.fit(
    x=X_train,
    y=y_train,
    validation_data=(X_val, y_val),
    batch_size=512,
    epochs=600,
    verbose=1,
    callbacks=[early_stopping])
```

Epoch 1/600
148/148 8s 14ms/step - accuracy: 0.4534 - loss: 1.8768 - val_accuracy: 0.5374 - val_loss: 0.5362
Epoch 2/600
148/148 2s 11ms/step - accuracy: 0.5878 - loss: 1.1951 - val_accuracy: 0.5840 - val_loss: 0.5018
Epoch 3/600
148/148 1s 10ms/step - accuracy: 0.6216 - loss: 0.9503 - val_accuracy: 0.6064 - val_loss: 0.49275
Epoch 4/600
148/148 1s 10ms/step - accuracy: 0.6417 - loss: 0.8355 - val_accuracy: 0.6210 - val_loss: 0.48165
Epoch 5/600
148/148 2s 11ms/step - accuracy: 0.6528 - loss: 0.7828 - val_accuracy: 0.6666 - val_loss: 0.47597
Epoch 6/600
148/148 2s 15ms/step - accuracy: 0.6519 - loss: 0.7705 - val_accuracy: 0.6678 - val_loss: 0.47516
Epoch 7/600
148/148 2s 12ms/step - accuracy: 0.6569 - loss: 0.7549 - val_accuracy: 0.6651 - val_loss: 0.47442
Epoch 8/600
148/148 2s 10ms/step - accuracy: 0.6595 - loss: 0.7553 - val_accuracy: 0.6677 - val_loss: 0.47415
Epoch 9/600
148/148 1s 10ms/step - accuracy: 0.6578 - loss: 0.7536 - val_accuracy: 0.6677 - val_loss: 0.47436
Epoch 10/600
148/148 2s 12ms/step - accuracy: 0.6616 - loss: 0.7485 - val_accuracy: 0.6670 - val_loss: 0.47371
Epoch 11/600
148/148 2s 12ms/step - accuracy: 0.6639 - loss: 0.7430 - val_accuracy: 0.6701 - val_loss: 0.47389
Epoch 12/600
148/148 2s 13ms/step - accuracy: 0.6675 - loss: 0.7384 - val_accuracy: 0.6698 - val_loss: 0.47342
Epoch 13/600
148/148 2s 14ms/step - accuracy: 0.6661 - loss: 0.7400 - val_accuracy: 0.6686 - val_loss: 0.47357
Epoch 14/600
148/148 2s 12ms/step - accuracy: 0.6677 - loss: 0.7356 - val_accuracy: 0.6713 - val_loss: 0.47359
Epoch 15/600
148/148 2s 11ms/step - accuracy: 0.6684 - loss: 0.7379 - val_accuracy: 0.6654 - val_loss: 0.47398
Epoch 16/600
148/148 2s 11ms/step - accuracy: 0.6654 - loss: 0.7394 - val_accuracy: 0.6744 - val_loss: 0.47275
Epoch 17/600
148/148 2s 11ms/step - accuracy: 0.6695 - loss: 0.7322 - val_accuracy: 0.6738 - val_loss: 0.47289
Epoch 18/600
148/148 1s 10ms/step - accuracy: 0.6687 - loss: 0.7352 - val_accuracy: 0.6752 - val_loss: 0.47280
Epoch 19/600
148/148 2s 11ms/step - accuracy: 0.6722 - loss: 0.7298 - val_accuracy: 0.6716 - val_loss: 0.47288
Epoch 20/600
148/148 2s 12ms/step - accuracy: 0.6678 - loss: 0.7333 - val_accuracy: 0.6704 - val_loss: 0.47300
Epoch 21/600
148/148 2s 11ms/step - accuracy: 0.6667 - loss: 0.7343 - val_accuracy: 0.6711 - val_loss: 0.47290
Epoch 22/600
148/148 2s 11ms/step - accuracy: 0.6704 - loss: 0.7326 - val_accuracy: 0.6701 - val_loss: 0.47315
Epoch 23/600
148/148 2s 13ms/step - accuracy: 0.6674 - loss: 0.7293 - val_accuracy: 0.6778 - val_loss: 0.47206
Epoch 24/600
148/148 2s 11ms/step - accuracy: 0.6698 - loss: 0.7317 - val_accuracy: 0.6694 - val_loss: 0.47283
Epoch 25/600
148/148 2s 10ms/step - accuracy: 0.6719 - loss: 0.7279 - val_accuracy: 0.6674 - val_loss: 0.47326
Epoch 26/600

148/148 2s 11ms/step - accuracy: 0.6714 - loss: 0.7300 - val_accuracy: 0.6701 - val_loss: 0.7303
Epoch 27/600
148/148 2s 10ms/step - accuracy: 0.6738 - loss: 0.7245 - val_accuracy: 0.6775 - val_loss: 0.7279
Epoch 28/600
148/148 2s 10ms/step - accuracy: 0.6727 - loss: 0.7271 - val_accuracy: 0.6770 - val_loss: 0.7227
Epoch 29/600
148/148 1s 10ms/step - accuracy: 0.6694 - loss: 0.7280 - val_accuracy: 0.6761 - val_loss: 0.7222
Epoch 30/600
148/148 2s 10ms/step - accuracy: 0.6746 - loss: 0.7244 - val_accuracy: 0.6757 - val_loss: 0.7236
Epoch 31/600
148/148 2s 10ms/step - accuracy: 0.6776 - loss: 0.7225 - val_accuracy: 0.6741 - val_loss: 0.7252
Epoch 32/600
148/148 2s 10ms/step - accuracy: 0.6753 - loss: 0.7219 - val_accuracy: 0.6774 - val_loss: 0.7203
Epoch 33/600
148/148 1s 10ms/step - accuracy: 0.6737 - loss: 0.7227 - val_accuracy: 0.6770 - val_loss: 0.7226
Epoch 34/600
148/148 2s 10ms/step - accuracy: 0.6721 - loss: 0.7249 - val_accuracy: 0.6779 - val_loss: 0.7216
Epoch 35/600
148/148 2s 11ms/step - accuracy: 0.6748 - loss: 0.7228 - val_accuracy: 0.6760 - val_loss: 0.7282
Epoch 36/600
148/148 2s 11ms/step - accuracy: 0.6735 - loss: 0.7245 - val_accuracy: 0.6772 - val_loss: 0.7210
Epoch 37/600
148/148 2s 10ms/step - accuracy: 0.6729 - loss: 0.7254 - val_accuracy: 0.6792 - val_loss: 0.7202
Epoch 38/600
148/148 2s 10ms/step - accuracy: 0.6765 - loss: 0.7159 - val_accuracy: 0.6757 - val_loss: 0.7210
Epoch 39/600
148/148 1s 10ms/step - accuracy: 0.6709 - loss: 0.7235 - val_accuracy: 0.6816 - val_loss: 0.7175
Epoch 40/600
148/148 1s 10ms/step - accuracy: 0.6750 - loss: 0.7215 - val_accuracy: 0.6745 - val_loss: 0.7319
Epoch 41/600
148/148 1s 10ms/step - accuracy: 0.6762 - loss: 0.7199 - val_accuracy: 0.6784 - val_loss: 0.7189
Epoch 42/600
148/148 1s 10ms/step - accuracy: 0.6746 - loss: 0.7240 - val_accuracy: 0.6717 - val_loss: 0.7242
Epoch 43/600
148/148 2s 10ms/step - accuracy: 0.6753 - loss: 0.7239 - val_accuracy: 0.6676 - val_loss: 0.7320
Epoch 44/600
148/148 2s 11ms/step - accuracy: 0.6758 - loss: 0.7194 - val_accuracy: 0.6710 - val_loss: 0.7282
Epoch 45/600
148/148 2s 11ms/step - accuracy: 0.6761 - loss: 0.7213 - val_accuracy: 0.6692 - val_loss: 0.7237
Epoch 46/600
148/148 2s 10ms/step - accuracy: 0.6762 - loss: 0.7192 - val_accuracy: 0.6798 - val_loss: 0.7179
Epoch 47/600
148/148 2s 10ms/step - accuracy: 0.6761 - loss: 0.7195 - val_accuracy: 0.6735 - val_loss: 0.7212
Epoch 48/600
148/148 1s 10ms/step - accuracy: 0.6775 - loss: 0.7166 - val_accuracy: 0.6793 - val_loss: 0.7176
Epoch 49/600
148/148 2s 10ms/step - accuracy: 0.6783 - loss: 0.7177 - val_accuracy: 0.6830 - val_loss: 0.7159
Epoch 50/600
148/148 1s 10ms/step - accuracy: 0.6784 - loss: 0.7186 - val_accuracy: 0.6770 - val_loss: 0.7181
Epoch 51/600
148/148 1s 10ms/step - accuracy: 0.6751 - loss: 0.7183 - val_accuracy: 0.6784 - val_loss:

s: 0.7184
Epoch 52/600
148/148 1s 10ms/step - accuracy: 0.6775 - loss: 0.7200 - val_accuracy: 0.6712 - val_loss: 0.7215
Epoch 53/600
148/148 2s 10ms/step - accuracy: 0.6782 - loss: 0.7155 - val_accuracy: 0.6803 - val_loss: 0.7156
Epoch 54/600
148/148 1s 10ms/step - accuracy: 0.6779 - loss: 0.7190 - val_accuracy: 0.6494 - val_loss: 0.7449
Epoch 55/600
148/148 2s 11ms/step - accuracy: 0.6773 - loss: 0.7180 - val_accuracy: 0.6769 - val_loss: 0.7180
s: 0.7235
Epoch 56/600
148/148 2s 12ms/step - accuracy: 0.6782 - loss: 0.7187 - val_accuracy: 0.6736 - val_loss: 0.7292
s: 0.7292
Epoch 57/600
148/148 2s 12ms/step - accuracy: 0.6764 - loss: 0.7183 - val_accuracy: 0.6729 - val_loss: 0.7186
s: 0.7210
Epoch 59/600
148/148 2s 11ms/step - accuracy: 0.6815 - loss: 0.7104 - val_accuracy: 0.6811 - val_loss: 0.7210
s: 0.7177
Epoch 60/600
148/148 2s 11ms/step - accuracy: 0.6773 - loss: 0.7139 - val_accuracy: 0.6827 - val_loss: 0.7178
s: 0.7178
Epoch 62/600
148/148 2s 10ms/step - accuracy: 0.6793 - loss: 0.7168 - val_accuracy: 0.6743 - val_loss: 0.7245
s: 0.7160
Epoch 63/600
148/148 2s 10ms/step - accuracy: 0.6758 - loss: 0.7130 - val_accuracy: 0.6789 - val_loss: 0.7143
s: 0.7133
Epoch 64/600
148/148 1s 10ms/step - accuracy: 0.6800 - loss: 0.7161 - val_accuracy: 0.6798 - val_loss: 0.7133
s: 0.7158
Epoch 66/600
148/148 2s 12ms/step - accuracy: 0.6797 - loss: 0.7180 - val_accuracy: 0.6826 - val_loss: 0.7158
s: 0.7128
Epoch 67/600
148/148 2s 10ms/step - accuracy: 0.6812 - loss: 0.7164 - val_accuracy: 0.6767 - val_loss: 0.7254
s: 0.7254
Epoch 68/600
148/148 1s 10ms/step - accuracy: 0.6816 - loss: 0.7120 - val_accuracy: 0.6710 - val_loss: 0.7257
s: 0.7181
Epoch 69/600
148/148 2s 10ms/step - accuracy: 0.6810 - loss: 0.7124 - val_accuracy: 0.6762 - val_loss: 0.7181
s: 0.7128
Epoch 70/600
148/148 1s 10ms/step - accuracy: 0.6769 - loss: 0.7122 - val_accuracy: 0.6820 - val_loss: 0.7141
s: 0.7097
Epoch 71/600
148/148 2s 10ms/step - accuracy: 0.6801 - loss: 0.7105 - val_accuracy: 0.6862 - val_loss: 0.7141
s: 0.7128
Epoch 73/600
148/148 2s 10ms/step - accuracy: 0.6834 - loss: 0.7066 - val_accuracy: 0.6812 - val_loss: 0.7168
s: 0.7168
Epoch 74/600
148/148 2s 10ms/step - accuracy: 0.6847 - loss: 0.7070 - val_accuracy: 0.6789 - val_loss: 0.7140
s: 0.7140
Epoch 75/600
148/148 2s 12ms/step - accuracy: 0.6850 - loss: 0.7096 - val_accuracy: 0.6863 - val_loss: 0.7081
s: 0.7129
Epoch 76/600
148/148 2s 11ms/step - accuracy: 0.6846 - loss: 0.7081 - val_accuracy: 0.6777 - val_loss: 0.7129

Epoch 77/600
148/148 2s 11ms/step - accuracy: 0.6799 - loss: 0.7123 - val_accuracy: 0.6805 - val_loss: 0.7154
Epoch 78/600
148/148 1s 10ms/step - accuracy: 0.6827 - loss: 0.7086 - val_accuracy: 0.6849 - val_loss: 0.7078
Epoch 79/600
148/148 2s 11ms/step - accuracy: 0.6817 - loss: 0.7139 - val_accuracy: 0.6838 - val_loss: 0.7111
s: 0.7111
Epoch 80/600
148/148 2s 10ms/step - accuracy: 0.6827 - loss: 0.7101 - val_accuracy: 0.6865 - val_loss: 0.7091
s: 0.7091
Epoch 81/600
148/148 2s 11ms/step - accuracy: 0.6818 - loss: 0.7098 - val_accuracy: 0.6851 - val_loss: 0.7100
s: 0.7100
Epoch 82/600
148/148 2s 11ms/step - accuracy: 0.6826 - loss: 0.7050 - val_accuracy: 0.6855 - val_loss: 0.7112
s: 0.7112
Epoch 83/600
148/148 2s 10ms/step - accuracy: 0.6814 - loss: 0.7101 - val_accuracy: 0.6862 - val_loss: 0.7082
s: 0.7082
Epoch 84/600
148/148 2s 10ms/step - accuracy: 0.6828 - loss: 0.7120 - val_accuracy: 0.6807 - val_loss: 0.7131
s: 0.7131
Epoch 85/600
148/148 2s 12ms/step - accuracy: 0.6847 - loss: 0.7081 - val_accuracy: 0.6797 - val_loss: 0.7129
s: 0.7129
Epoch 86/600
148/148 2s 12ms/step - accuracy: 0.6865 - loss: 0.7051 - val_accuracy: 0.6847 - val_loss: 0.7088
s: 0.7088
Epoch 87/600
148/148 2s 11ms/step - accuracy: 0.6857 - loss: 0.7066 - val_accuracy: 0.6821 - val_loss: 0.7104
s: 0.7104
Epoch 88/600
148/148 1s 10ms/step - accuracy: 0.6816 - loss: 0.7062 - val_accuracy: 0.6831 - val_loss: 0.7137
s: 0.7137
Epoch 89/600
148/148 2s 10ms/step - accuracy: 0.6856 - loss: 0.7028 - val_accuracy: 0.6867 - val_loss: 0.7058
s: 0.7058
Epoch 90/600
148/148 1s 10ms/step - accuracy: 0.6827 - loss: 0.7055 - val_accuracy: 0.6822 - val_loss: 0.7094
s: 0.7094
Epoch 91/600
148/148 2s 10ms/step - accuracy: 0.6866 - loss: 0.7027 - val_accuracy: 0.6843 - val_loss: 0.7076
s: 0.7076
Epoch 92/600
148/148 2s 10ms/step - accuracy: 0.6877 - loss: 0.7029 - val_accuracy: 0.6852 - val_loss: 0.7052
s: 0.7052
Epoch 93/600
148/148 2s 11ms/step - accuracy: 0.6878 - loss: 0.7042 - val_accuracy: 0.6846 - val_loss: 0.7108
s: 0.7108
Epoch 94/600
148/148 2s 11ms/step - accuracy: 0.6883 - loss: 0.7039 - val_accuracy: 0.6844 - val_loss: 0.7101
s: 0.7101
Epoch 95/600
148/148 2s 12ms/step - accuracy: 0.6882 - loss: 0.7016 - val_accuracy: 0.6836 - val_loss: 0.7068
s: 0.7068
Epoch 96/600
148/148 2s 11ms/step - accuracy: 0.6856 - loss: 0.7039 - val_accuracy: 0.6899 - val_loss: 0.7034
s: 0.7034
Epoch 97/600
148/148 2s 10ms/step - accuracy: 0.6870 - loss: 0.7046 - val_accuracy: 0.6881 - val_loss: 0.7065
s: 0.7065
Epoch 98/600
148/148 2s 10ms/step - accuracy: 0.6869 - loss: 0.7015 - val_accuracy: 0.6904 - val_loss: 0.7007
s: 0.7007
Epoch 99/600
148/148 1s 10ms/step - accuracy: 0.6867 - loss: 0.7013 - val_accuracy: 0.6854 - val_loss: 0.7084
s: 0.7084
Epoch 100/600
148/148 1s 10ms/step - accuracy: 0.6866 - loss: 0.7027 - val_accuracy: 0.6856 - val_loss: 0.7058
s: 0.7058
Epoch 101/600
148/148 1s 10ms/step - accuracy: 0.6898 - loss: 0.6991 - val_accuracy: 0.6777 - val_loss: 0.7152
s: 0.7152
Epoch 102/600

148/148 1s 10ms/step - accuracy: 0.6901 - loss: 0.7001 - val_accuracy: 0.6903 - val_loss: 0.7023
Epoch 103/600
148/148 2s 10ms/step - accuracy: 0.6891 - loss: 0.6991 - val_accuracy: 0.6871 - val_loss: 0.7054
Epoch 104/600
148/148 1s 10ms/step - accuracy: 0.6891 - loss: 0.7011 - val_accuracy: 0.6887 - val_loss: 0.6991
Epoch 105/600
148/148 2s 11ms/step - accuracy: 0.6896 - loss: 0.6975 - val_accuracy: 0.6844 - val_loss: 0.7068
Epoch 106/600
148/148 2s 11ms/step - accuracy: 0.6889 - loss: 0.6988 - val_accuracy: 0.6899 - val_loss: 0.7013
Epoch 107/600
148/148 2s 10ms/step - accuracy: 0.6892 - loss: 0.7028 - val_accuracy: 0.6922 - val_loss: 0.7010
Epoch 108/600
148/148 1s 10ms/step - accuracy: 0.6922 - loss: 0.6967 - val_accuracy: 0.6882 - val_loss: 0.7029
Epoch 109/600
148/148 2s 10ms/step - accuracy: 0.6888 - loss: 0.6992 - val_accuracy: 0.6890 - val_loss: 0.7012
Epoch 110/600
148/148 1s 10ms/step - accuracy: 0.6888 - loss: 0.7006 - val_accuracy: 0.6909 - val_loss: 0.6992
Epoch 111/600
148/148 1s 10ms/step - accuracy: 0.6916 - loss: 0.6962 - val_accuracy: 0.6888 - val_loss: 0.7032
Epoch 112/600
148/148 1s 10ms/step - accuracy: 0.6906 - loss: 0.6978 - val_accuracy: 0.6876 - val_loss: 0.7026
Epoch 113/600
148/148 2s 11ms/step - accuracy: 0.6908 - loss: 0.6961 - val_accuracy: 0.6843 - val_loss: 0.7046
Epoch 114/600
148/148 2s 10ms/step - accuracy: 0.6914 - loss: 0.6981 - val_accuracy: 0.6848 - val_loss: 0.7081
Epoch 115/600
148/148 2s 11ms/step - accuracy: 0.6905 - loss: 0.6947 - val_accuracy: 0.6941 - val_loss: 0.6969
Epoch 116/600
148/148 2s 12ms/step - accuracy: 0.6946 - loss: 0.6912 - val_accuracy: 0.6929 - val_loss: 0.7042
Epoch 117/600
148/148 2s 14ms/step - accuracy: 0.6932 - loss: 0.6937 - val_accuracy: 0.6954 - val_loss: 0.6934
Epoch 118/600
148/148 2s 12ms/step - accuracy: 0.6910 - loss: 0.6961 - val_accuracy: 0.6927 - val_loss: 0.6949
Epoch 119/600
148/148 1s 10ms/step - accuracy: 0.6915 - loss: 0.6938 - val_accuracy: 0.6933 - val_loss: 0.6960
Epoch 120/600
148/148 2s 10ms/step - accuracy: 0.6926 - loss: 0.6919 - val_accuracy: 0.6934 - val_loss: 0.6939
Epoch 121/600
148/148 1s 10ms/step - accuracy: 0.6960 - loss: 0.6914 - val_accuracy: 0.6948 - val_loss: 0.6930
Epoch 122/600
148/148 1s 10ms/step - accuracy: 0.6971 - loss: 0.6886 - val_accuracy: 0.6846 - val_loss: 0.7061
Epoch 123/600
148/148 2s 10ms/step - accuracy: 0.6916 - loss: 0.6950 - val_accuracy: 0.6933 - val_loss: 0.6928
Epoch 124/600
148/148 2s 10ms/step - accuracy: 0.6947 - loss: 0.6924 - val_accuracy: 0.6950 - val_loss: 0.6952
Epoch 125/600
148/148 2s 11ms/step - accuracy: 0.6966 - loss: 0.6879 - val_accuracy: 0.6952 - val_loss: 0.6932
Epoch 126/600
148/148 2s 11ms/step - accuracy: 0.6916 - loss: 0.6916 - val_accuracy: 0.6911 - val_loss: 0.6933
Epoch 127/600
148/148 3s 10ms/step - accuracy: 0.6954 - loss: 0.6893 - val_accuracy: 0.6965 - val_loss:

s: 0.6943
Epoch 128/600
148/148 2s 10ms/step - accuracy: 0.6952 - loss: 0.6906 - val_accuracy: 0.6948 - val_loss: 0.6948
s: 0.6976
Epoch 129/600
148/148 1s 10ms/step - accuracy: 0.6948 - loss: 0.6883 - val_accuracy: 0.6922 - val_loss: 0.6922
s: 0.6967
Epoch 130/600
148/148 2s 10ms/step - accuracy: 0.6966 - loss: 0.6908 - val_accuracy: 0.6979 - val_loss: 0.6979
s: 0.6904
Epoch 131/600
148/148 1s 10ms/step - accuracy: 0.6952 - loss: 0.6891 - val_accuracy: 0.6953 - val_loss: 0.6953
s: 0.6911
Epoch 132/600
148/148 1s 10ms/step - accuracy: 0.6963 - loss: 0.6868 - val_accuracy: 0.6951 - val_loss: 0.6951
s: 0.6987
Epoch 133/600
148/148 1s 10ms/step - accuracy: 0.6958 - loss: 0.6897 - val_accuracy: 0.6962 - val_loss: 0.6962
s: 0.6956
Epoch 134/600
148/148 2s 11ms/step - accuracy: 0.7015 - loss: 0.6839 - val_accuracy: 0.6838 - val_loss: 0.6838
s: 0.7087
Epoch 135/600
148/148 2s 11ms/step - accuracy: 0.6974 - loss: 0.6872 - val_accuracy: 0.6945 - val_loss: 0.6945
s: 0.6959
Epoch 136/600
148/148 2s 11ms/step - accuracy: 0.6962 - loss: 0.6868 - val_accuracy: 0.6978 - val_loss: 0.6978
s: 0.6887
Epoch 137/600
148/148 2s 10ms/step - accuracy: 0.6965 - loss: 0.6891 - val_accuracy: 0.6987 - val_loss: 0.6987
s: 0.6897
Epoch 138/600
148/148 2s 10ms/step - accuracy: 0.6982 - loss: 0.6870 - val_accuracy: 0.6989 - val_loss: 0.6989
s: 0.6863
Epoch 139/600
148/148 1s 10ms/step - accuracy: 0.6957 - loss: 0.6861 - val_accuracy: 0.6998 - val_loss: 0.6998
s: 0.6873
Epoch 140/600
148/148 2s 10ms/step - accuracy: 0.7014 - loss: 0.6829 - val_accuracy: 0.7011 - val_loss: 0.7011
s: 0.6841
Epoch 141/600
148/148 1s 10ms/step - accuracy: 0.6997 - loss: 0.6826 - val_accuracy: 0.6908 - val_loss: 0.6908
s: 0.7013
Epoch 142/600
148/148 2s 10ms/step - accuracy: 0.6964 - loss: 0.6885 - val_accuracy: 0.6990 - val_loss: 0.6990
s: 0.6912
Epoch 143/600
148/148 2s 11ms/step - accuracy: 0.7028 - loss: 0.6841 - val_accuracy: 0.6991 - val_loss: 0.6991
s: 0.6844
Epoch 144/600
148/148 2s 11ms/step - accuracy: 0.6979 - loss: 0.6800 - val_accuracy: 0.7030 - val_loss: 0.7030
s: 0.6832
Epoch 145/600
148/148 2s 11ms/step - accuracy: 0.7005 - loss: 0.6835 - val_accuracy: 0.6995 - val_loss: 0.6995
s: 0.6876
Epoch 146/600
148/148 2s 11ms/step - accuracy: 0.7004 - loss: 0.6845 - val_accuracy: 0.7034 - val_loss: 0.7034
s: 0.6828
Epoch 147/600
148/148 2s 11ms/step - accuracy: 0.6994 - loss: 0.6828 - val_accuracy: 0.6951 - val_loss: 0.6951
s: 0.6958
Epoch 148/600
148/148 2s 10ms/step - accuracy: 0.7034 - loss: 0.6781 - val_accuracy: 0.7010 - val_loss: 0.7010
s: 0.6829
Epoch 149/600
148/148 2s 11ms/step - accuracy: 0.7003 - loss: 0.6820 - val_accuracy: 0.6969 - val_loss: 0.6969
s: 0.6971
Epoch 150/600
148/148 2s 12ms/step - accuracy: 0.6972 - loss: 0.6864 - val_accuracy: 0.7038 - val_loss: 0.7038
s: 0.6796
Epoch 151/600
148/148 2s 10ms/step - accuracy: 0.7002 - loss: 0.6802 - val_accuracy: 0.6977 - val_loss: 0.6977
s: 0.6966
Epoch 152/600
148/148 2s 11ms/step - accuracy: 0.6997 - loss: 0.6823 - val_accuracy: 0.7035 - val_loss: 0.7035
s: 0.6823

Epoch 153/600
148/148 2s 10ms/step - accuracy: 0.7010 - loss: 0.6789 - val_accuracy: 0.6987 - val_loss: 0.6908
Epoch 154/600
148/148 2s 10ms/step - accuracy: 0.7008 - loss: 0.6788 - val_accuracy: 0.7026 - val_loss: 0.6854
Epoch 155/600
148/148 2s 11ms/step - accuracy: 0.6966 - loss: 0.6885 - val_accuracy: 0.6998 - val_loss: 0.6863
s: 0.6863
Epoch 156/600
148/148 2s 11ms/step - accuracy: 0.7005 - loss: 0.6834 - val_accuracy: 0.7002 - val_loss: 0.6840
s: 0.6840
Epoch 157/600
148/148 2s 10ms/step - accuracy: 0.7037 - loss: 0.6767 - val_accuracy: 0.7031 - val_loss: 0.6812
s: 0.6812
Epoch 158/600
148/148 2s 11ms/step - accuracy: 0.7007 - loss: 0.6810 - val_accuracy: 0.7094 - val_loss: 0.6769
s: 0.6769
Epoch 159/600
148/148 2s 10ms/step - accuracy: 0.7027 - loss: 0.6781 - val_accuracy: 0.7025 - val_loss: 0.6832
s: 0.6832
Epoch 160/600
148/148 2s 11ms/step - accuracy: 0.7001 - loss: 0.6798 - val_accuracy: 0.6998 - val_loss: 0.6857
s: 0.6857
Epoch 161/600
148/148 2s 10ms/step - accuracy: 0.7023 - loss: 0.6792 - val_accuracy: 0.7068 - val_loss: 0.6790
s: 0.6790
Epoch 162/600
148/148 2s 11ms/step - accuracy: 0.7031 - loss: 0.6795 - val_accuracy: 0.7046 - val_loss: 0.6827
s: 0.6827
Epoch 163/600
148/148 2s 10ms/step - accuracy: 0.7004 - loss: 0.6787 - val_accuracy: 0.7082 - val_loss: 0.6775
s: 0.6775
Epoch 164/600
148/148 2s 10ms/step - accuracy: 0.7032 - loss: 0.6792 - val_accuracy: 0.7093 - val_loss: 0.6732
s: 0.6732
Epoch 165/600
148/148 2s 10ms/step - accuracy: 0.7012 - loss: 0.6816 - val_accuracy: 0.7073 - val_loss: 0.6763
s: 0.6763
Epoch 166/600
148/148 2s 11ms/step - accuracy: 0.7079 - loss: 0.6722 - val_accuracy: 0.7064 - val_loss: 0.6817
s: 0.6817
Epoch 167/600
148/148 2s 10ms/step - accuracy: 0.7009 - loss: 0.6801 - val_accuracy: 0.7071 - val_loss: 0.6751
s: 0.6751
Epoch 168/600
148/148 2s 11ms/step - accuracy: 0.7061 - loss: 0.6761 - val_accuracy: 0.7054 - val_loss: 0.6795
s: 0.6795
Epoch 169/600
148/148 2s 10ms/step - accuracy: 0.7036 - loss: 0.6796 - val_accuracy: 0.7000 - val_loss: 0.6839
s: 0.6839
Epoch 170/600
148/148 2s 10ms/step - accuracy: 0.7033 - loss: 0.6789 - val_accuracy: 0.7110 - val_loss: 0.6748
s: 0.6748
Epoch 171/600
148/148 2s 11ms/step - accuracy: 0.7055 - loss: 0.6748 - val_accuracy: 0.7039 - val_loss: 0.6781
s: 0.6781
Epoch 172/600
148/148 2s 11ms/step - accuracy: 0.7039 - loss: 0.6763 - val_accuracy: 0.7065 - val_loss: 0.6742
s: 0.6742
Epoch 173/600
148/148 2s 10ms/step - accuracy: 0.7033 - loss: 0.6779 - val_accuracy: 0.7025 - val_loss: 0.6782
s: 0.6782
Epoch 174/600
148/148 2s 11ms/step - accuracy: 0.7032 - loss: 0.6774 - val_accuracy: 0.7037 - val_loss: 0.6794
s: 0.6794
Epoch 175/600
148/148 2s 12ms/step - accuracy: 0.7050 - loss: 0.6726 - val_accuracy: 0.7110 - val_loss: 0.6714
s: 0.6714
Epoch 176/600
148/148 2s 12ms/step - accuracy: 0.7038 - loss: 0.6744 - val_accuracy: 0.7116 - val_loss: 0.6727
s: 0.6727
Epoch 177/600
148/148 2s 11ms/step - accuracy: 0.7023 - loss: 0.6780 - val_accuracy: 0.7054 - val_loss: 0.6806
s: 0.6806
Epoch 178/600

148/148 1s 10ms/step - accuracy: 0.7041 - loss: 0.6750 - val_accuracy: 0.7088 - val_loss: 0.6749
Epoch 179/600
148/148 2s 11ms/step - accuracy: 0.7015 - loss: 0.6785 - val_accuracy: 0.7076 - val_loss: 0.6708
Epoch 180/600
148/148 2s 11ms/step - accuracy: 0.7065 - loss: 0.6688 - val_accuracy: 0.7080 - val_loss: 0.6759
Epoch 181/600
148/148 1s 10ms/step - accuracy: 0.7065 - loss: 0.6695 - val_accuracy: 0.7042 - val_loss: 0.6798
Epoch 182/600
148/148 2s 11ms/step - accuracy: 0.7073 - loss: 0.6710 - val_accuracy: 0.7093 - val_loss: 0.6728
Epoch 183/600
148/148 2s 10ms/step - accuracy: 0.7050 - loss: 0.6729 - val_accuracy: 0.7148 - val_loss: 0.6672
Epoch 184/600
148/148 2s 11ms/step - accuracy: 0.7086 - loss: 0.6738 - val_accuracy: 0.7116 - val_loss: 0.6713
Epoch 185/600
148/148 2s 12ms/step - accuracy: 0.7068 - loss: 0.6724 - val_accuracy: 0.7112 - val_loss: 0.6681
Epoch 186/600
148/148 2s 11ms/step - accuracy: 0.7079 - loss: 0.6685 - val_accuracy: 0.7108 - val_loss: 0.6667
Epoch 187/600
148/148 2s 11ms/step - accuracy: 0.7076 - loss: 0.6688 - val_accuracy: 0.7129 - val_loss: 0.6705
Epoch 188/600
148/148 2s 11ms/step - accuracy: 0.7085 - loss: 0.6704 - val_accuracy: 0.7152 - val_loss: 0.6686
Epoch 189/600
148/148 2s 11ms/step - accuracy: 0.7088 - loss: 0.6701 - val_accuracy: 0.7141 - val_loss: 0.6701
Epoch 190/600
148/148 2s 11ms/step - accuracy: 0.7042 - loss: 0.6752 - val_accuracy: 0.7106 - val_loss: 0.6735
Epoch 191/600
148/148 2s 11ms/step - accuracy: 0.7107 - loss: 0.6675 - val_accuracy: 0.7116 - val_loss: 0.6684
Epoch 192/600
148/148 2s 10ms/step - accuracy: 0.7056 - loss: 0.6721 - val_accuracy: 0.7151 - val_loss: 0.6605
Epoch 193/600
148/148 2s 10ms/step - accuracy: 0.7075 - loss: 0.6686 - val_accuracy: 0.7131 - val_loss: 0.6651
Epoch 194/600
148/148 2s 11ms/step - accuracy: 0.7075 - loss: 0.6684 - val_accuracy: 0.7136 - val_loss: 0.6676
Epoch 195/600
148/148 2s 11ms/step - accuracy: 0.7061 - loss: 0.6691 - val_accuracy: 0.7181 - val_loss: 0.6667
Epoch 196/600
148/148 2s 11ms/step - accuracy: 0.7052 - loss: 0.6700 - val_accuracy: 0.7129 - val_loss: 0.6624
Epoch 197/600
148/148 2s 11ms/step - accuracy: 0.7050 - loss: 0.6752 - val_accuracy: 0.7115 - val_loss: 0.6704
Epoch 198/600
148/148 1s 10ms/step - accuracy: 0.7103 - loss: 0.6680 - val_accuracy: 0.7150 - val_loss: 0.6647
Epoch 199/600
148/148 1s 10ms/step - accuracy: 0.7078 - loss: 0.6692 - val_accuracy: 0.7179 - val_loss: 0.6635
Epoch 200/600
148/148 2s 10ms/step - accuracy: 0.7068 - loss: 0.6721 - val_accuracy: 0.7128 - val_loss: 0.6686
Epoch 201/600
148/148 1s 10ms/step - accuracy: 0.7080 - loss: 0.6688 - val_accuracy: 0.7120 - val_loss: 0.6674
Epoch 202/600
148/148 2s 10ms/step - accuracy: 0.7081 - loss: 0.6690 - val_accuracy: 0.7172 - val_loss: 0.6623
Epoch 203/600
148/148 1s 10ms/step - accuracy: 0.7045 - loss: 0.6718 - val_accuracy: 0.7152 - val_loss:

s: 0.6649
Epoch 204/600
148/148 2s 10ms/step - accuracy: 0.7066 - loss: 0.6691 - val_accuracy: 0.7157 - val_loss: 0.6617
Epoch 205/600
148/148 2s 10ms/step - accuracy: 0.7052 - loss: 0.6718 - val_accuracy: 0.7125 - val_loss: 0.6683
Epoch 206/600
148/148 2s 11ms/step - accuracy: 0.7069 - loss: 0.6692 - val_accuracy: 0.7176 - val_loss: 0.6640
Epoch 207/600
148/148 2s 10ms/step - accuracy: 0.7095 - loss: 0.6662 - val_accuracy: 0.7162 - val_loss: 0.6626
Epoch 208/600
148/148 2s 11ms/step - accuracy: 0.7077 - loss: 0.6680 - val_accuracy: 0.7157 - val_loss: 0.6655
Epoch 209/600
148/148 2s 11ms/step - accuracy: 0.7114 - loss: 0.6636 - val_accuracy: 0.7154 - val_loss: 0.6651
Epoch 210/600
148/148 2s 11ms/step - accuracy: 0.7111 - loss: 0.6639 - val_accuracy: 0.7190 - val_loss: 0.6591
Epoch 211/600
148/148 2s 11ms/step - accuracy: 0.7095 - loss: 0.6637 - val_accuracy: 0.7187 - val_loss: 0.6577
Epoch 212/600
148/148 2s 11ms/step - accuracy: 0.7104 - loss: 0.6635 - val_accuracy: 0.7131 - val_loss: 0.6659
Epoch 213/600
148/148 1s 10ms/step - accuracy: 0.7100 - loss: 0.6650 - val_accuracy: 0.7172 - val_loss: 0.6678
Epoch 214/600
148/148 2s 10ms/step - accuracy: 0.7080 - loss: 0.6705 - val_accuracy: 0.7171 - val_loss: 0.6647
Epoch 215/600
148/148 2s 11ms/step - accuracy: 0.7147 - loss: 0.6616 - val_accuracy: 0.7142 - val_loss: 0.6633
Epoch 216/600
148/148 2s 12ms/step - accuracy: 0.7095 - loss: 0.6669 - val_accuracy: 0.7146 - val_loss: 0.6604
Epoch 217/600
148/148 2s 13ms/step - accuracy: 0.7102 - loss: 0.6634 - val_accuracy: 0.7211 - val_loss: 0.6579
Epoch 218/600
148/148 2s 11ms/step - accuracy: 0.7124 - loss: 0.6611 - val_accuracy: 0.7180 - val_loss: 0.6653
Epoch 219/600
148/148 2s 11ms/step - accuracy: 0.7083 - loss: 0.6628 - val_accuracy: 0.7133 - val_loss: 0.6627
Epoch 220/600
148/148 2s 10ms/step - accuracy: 0.7128 - loss: 0.6616 - val_accuracy: 0.7204 - val_loss: 0.6598
Epoch 221/600
148/148 2s 10ms/step - accuracy: 0.7087 - loss: 0.6631 - val_accuracy: 0.7139 - val_loss: 0.6636
Epoch 222/600
148/148 2s 11ms/step - accuracy: 0.7101 - loss: 0.6692 - val_accuracy: 0.7160 - val_loss: 0.6673
Epoch 223/600
148/148 2s 11ms/step - accuracy: 0.7102 - loss: 0.6645 - val_accuracy: 0.7224 - val_loss: 0.6556
Epoch 224/600
148/148 2s 12ms/step - accuracy: 0.7071 - loss: 0.6641 - val_accuracy: 0.7186 - val_loss: 0.6606
Epoch 225/600
148/148 2s 11ms/step - accuracy: 0.7123 - loss: 0.6625 - val_accuracy: 0.7157 - val_loss: 0.6623
Epoch 226/600
148/148 2s 11ms/step - accuracy: 0.7110 - loss: 0.6646 - val_accuracy: 0.7234 - val_loss: 0.6583
Epoch 227/600
148/148 2s 11ms/step - accuracy: 0.7139 - loss: 0.6589 - val_accuracy: 0.7242 - val_loss: 0.6525
Epoch 228/600
148/148 1s 10ms/step - accuracy: 0.7119 - loss: 0.6608 - val_accuracy: 0.7232 - val_loss: 0.6607

Epoch 229/600
148/148 2s 10ms/step - accuracy: 0.7142 - loss: 0.6590 - val_accuracy: 0.7082 - val_loss: 0.6784
Epoch 230/600
148/148 2s 10ms/step - accuracy: 0.7117 - loss: 0.6631 - val_accuracy: 0.7209 - val_loss: 0.6583
Epoch 231/600
148/148 2s 11ms/step - accuracy: 0.7130 - loss: 0.6577 - val_accuracy: 0.7195 - val_loss: 0.6588
Epoch 232/600
148/148 2s 11ms/step - accuracy: 0.7172 - loss: 0.6546 - val_accuracy: 0.7176 - val_loss: 0.6594
Epoch 233/600
148/148 2s 10ms/step - accuracy: 0.7122 - loss: 0.6583 - val_accuracy: 0.7158 - val_loss: 0.6609
Epoch 234/600
148/148 2s 10ms/step - accuracy: 0.7143 - loss: 0.6570 - val_accuracy: 0.7140 - val_loss: 0.6649
Epoch 235/600
148/148 2s 11ms/step - accuracy: 0.7118 - loss: 0.6635 - val_accuracy: 0.7183 - val_loss: 0.6591
Epoch 236/600
148/148 2s 12ms/step - accuracy: 0.7116 - loss: 0.6637 - val_accuracy: 0.7265 - val_loss: 0.6540
Epoch 237/600
148/148 2s 12ms/step - accuracy: 0.7153 - loss: 0.6573 - val_accuracy: 0.7215 - val_loss: 0.6555
Epoch 238/600
148/148 2s 10ms/step - accuracy: 0.7102 - loss: 0.6612 - val_accuracy: 0.7196 - val_loss: 0.6580
Epoch 239/600
148/148 1s 10ms/step - accuracy: 0.7146 - loss: 0.6551 - val_accuracy: 0.7218 - val_loss: 0.6565
Epoch 240/600
148/148 1s 10ms/step - accuracy: 0.7134 - loss: 0.6611 - val_accuracy: 0.7254 - val_loss: 0.6510
Epoch 241/600
148/148 2s 10ms/step - accuracy: 0.7107 - loss: 0.6645 - val_accuracy: 0.7238 - val_loss: 0.6525
Epoch 242/600
148/148 1s 10ms/step - accuracy: 0.7089 - loss: 0.6670 - val_accuracy: 0.7218 - val_loss: 0.6529
Epoch 243/600
148/148 1s 10ms/step - accuracy: 0.7147 - loss: 0.6577 - val_accuracy: 0.7216 - val_loss: 0.6602
Epoch 244/600
148/148 2s 10ms/step - accuracy: 0.7154 - loss: 0.6599 - val_accuracy: 0.7198 - val_loss: 0.6565
Epoch 245/600
148/148 2s 12ms/step - accuracy: 0.7145 - loss: 0.6568 - val_accuracy: 0.7237 - val_loss: 0.6550
Epoch 246/600
148/148 2s 12ms/step - accuracy: 0.7146 - loss: 0.6590 - val_accuracy: 0.7241 - val_loss: 0.6522
Epoch 247/600
148/148 2s 12ms/step - accuracy: 0.7140 - loss: 0.6585 - val_accuracy: 0.7250 - val_loss: 0.6516
Epoch 248/600
148/148 2s 10ms/step - accuracy: 0.7176 - loss: 0.6535 - val_accuracy: 0.7213 - val_loss: 0.6504
Epoch 249/600
148/148 2s 10ms/step - accuracy: 0.7154 - loss: 0.6588 - val_accuracy: 0.7237 - val_loss: 0.6490
Epoch 250/600
148/148 2s 11ms/step - accuracy: 0.7132 - loss: 0.6571 - val_accuracy: 0.7031 - val_loss: 0.6817
Epoch 251/600
148/148 2s 10ms/step - accuracy: 0.7150 - loss: 0.6567 - val_accuracy: 0.7222 - val_loss: 0.6508
Epoch 252/600
148/148 2s 10ms/step - accuracy: 0.7132 - loss: 0.6578 - val_accuracy: 0.7248 - val_loss: 0.6492
Epoch 253/600
148/148 2s 10ms/step - accuracy: 0.7178 - loss: 0.6559 - val_accuracy: 0.7275 - val_loss: 0.6508
Epoch 254/600

148/148 2s 11ms/step - accuracy: 0.7187 - loss: 0.6548 - val_accuracy: 0.7255 - val_loss: 0.6477
Epoch 255/600
148/148 2s 11ms/step - accuracy: 0.7161 - loss: 0.6579 - val_accuracy: 0.7154 - val_loss: 0.6662
Epoch 256/600
148/148 2s 10ms/step - accuracy: 0.7154 - loss: 0.6583 - val_accuracy: 0.7235 - val_loss: 0.6487
Epoch 257/600
148/148 2s 10ms/step - accuracy: 0.7146 - loss: 0.6569 - val_accuracy: 0.7245 - val_loss: 0.6496
Epoch 258/600
148/148 2s 11ms/step - accuracy: 0.7168 - loss: 0.6539 - val_accuracy: 0.7210 - val_loss: 0.6508
Epoch 259/600
148/148 2s 10ms/step - accuracy: 0.7170 - loss: 0.6535 - val_accuracy: 0.7256 - val_loss: 0.6455
Epoch 260/600
148/148 1s 10ms/step - accuracy: 0.7155 - loss: 0.6591 - val_accuracy: 0.7267 - val_loss: 0.6452
Epoch 261/600
148/148 2s 10ms/step - accuracy: 0.7186 - loss: 0.6530 - val_accuracy: 0.7269 - val_loss: 0.6489
Epoch 262/600
148/148 2s 10ms/step - accuracy: 0.7159 - loss: 0.6565 - val_accuracy: 0.7244 - val_loss: 0.6530
Epoch 263/600
148/148 2s 11ms/step - accuracy: 0.7158 - loss: 0.6575 - val_accuracy: 0.7253 - val_loss: 0.6528
Epoch 264/600
148/148 2s 11ms/step - accuracy: 0.7167 - loss: 0.6548 - val_accuracy: 0.7269 - val_loss: 0.6484
Epoch 265/600
148/148 2s 11ms/step - accuracy: 0.7167 - loss: 0.6545 - val_accuracy: 0.7139 - val_loss: 0.6673
Epoch 266/600
148/148 2s 11ms/step - accuracy: 0.7173 - loss: 0.6572 - val_accuracy: 0.7208 - val_loss: 0.6582
Epoch 267/600
148/148 2s 10ms/step - accuracy: 0.7171 - loss: 0.6528 - val_accuracy: 0.7219 - val_loss: 0.6603
Epoch 268/600
148/148 1s 10ms/step - accuracy: 0.7156 - loss: 0.6552 - val_accuracy: 0.7243 - val_loss: 0.6531
Epoch 269/600
148/148 1s 10ms/step - accuracy: 0.7182 - loss: 0.6489 - val_accuracy: 0.7266 - val_loss: 0.6440
Epoch 270/600
148/148 1s 10ms/step - accuracy: 0.7194 - loss: 0.6507 - val_accuracy: 0.7267 - val_loss: 0.6495
Epoch 271/600
148/148 1s 10ms/step - accuracy: 0.7179 - loss: 0.6527 - val_accuracy: 0.7173 - val_loss: 0.6568
Epoch 272/600
148/148 2s 10ms/step - accuracy: 0.7202 - loss: 0.6522 - val_accuracy: 0.7241 - val_loss: 0.6502
Epoch 273/600
148/148 1s 10ms/step - accuracy: 0.7140 - loss: 0.6559 - val_accuracy: 0.7249 - val_loss: 0.6488
Epoch 274/600
148/148 1s 10ms/step - accuracy: 0.7177 - loss: 0.6525 - val_accuracy: 0.7289 - val_loss: 0.6489
Epoch 275/600
148/148 2s 11ms/step - accuracy: 0.7202 - loss: 0.6482 - val_accuracy: 0.7290 - val_loss: 0.6439
Epoch 276/600
148/148 2s 11ms/step - accuracy: 0.7179 - loss: 0.6524 - val_accuracy: 0.7285 - val_loss: 0.6458
Epoch 277/600
148/148 2s 12ms/step - accuracy: 0.7187 - loss: 0.6524 - val_accuracy: 0.7218 - val_loss: 0.6548
Epoch 278/600
148/148 2s 10ms/step - accuracy: 0.7145 - loss: 0.6541 - val_accuracy: 0.7255 - val_loss: 0.6472
Epoch 279/600
148/148 2s 11ms/step - accuracy: 0.7145 - loss: 0.6540 - val_accuracy: 0.7161 - val_loss:

s: 0.6678
Epoch 280/600
148/148 2s 11ms/step - accuracy: 0.7187 - loss: 0.6558 - val_accuracy: 0.7296 - val_loss: 0.6453
Epoch 281/600
148/148 2s 10ms/step - accuracy: 0.7175 - loss: 0.6521 - val_accuracy: 0.7283 - val_loss: 0.6487
Epoch 282/600
148/148 2s 12ms/step - accuracy: 0.7200 - loss: 0.6489 - val_accuracy: 0.7288 - val_loss: 0.6466
Epoch 283/600
148/148 2s 10ms/step - accuracy: 0.7182 - loss: 0.6547 - val_accuracy: 0.7296 - val_loss: 0.6467
Epoch 284/600
148/148 1s 10ms/step - accuracy: 0.7205 - loss: 0.6495 - val_accuracy: 0.7248 - val_loss: 0.6437
Epoch 285/600
148/148 2s 11ms/step - accuracy: 0.7210 - loss: 0.6475 - val_accuracy: 0.7289 - val_loss: 0.6395
Epoch 286/600
148/148 2s 11ms/step - accuracy: 0.7162 - loss: 0.6500 - val_accuracy: 0.7225 - val_loss: 0.6484
Epoch 287/600
148/148 2s 11ms/step - accuracy: 0.7154 - loss: 0.6576 - val_accuracy: 0.7300 - val_loss: 0.6414
Epoch 288/600
148/148 2s 12ms/step - accuracy: 0.7192 - loss: 0.6524 - val_accuracy: 0.7300 - val_loss: 0.6417
Epoch 289/600
148/148 2s 11ms/step - accuracy: 0.7194 - loss: 0.6505 - val_accuracy: 0.7310 - val_loss: 0.6417
Epoch 290/600
148/148 2s 11ms/step - accuracy: 0.7177 - loss: 0.6538 - val_accuracy: 0.7265 - val_loss: 0.6540
Epoch 291/600
148/148 2s 10ms/step - accuracy: 0.7176 - loss: 0.6508 - val_accuracy: 0.7317 - val_loss: 0.6467
Epoch 292/600
148/148 2s 11ms/step - accuracy: 0.7205 - loss: 0.6504 - val_accuracy: 0.7269 - val_loss: 0.6467
Epoch 293/600
148/148 2s 12ms/step - accuracy: 0.7199 - loss: 0.6501 - val_accuracy: 0.7303 - val_loss: 0.6476
Epoch 294/600
148/148 2s 10ms/step - accuracy: 0.7169 - loss: 0.6492 - val_accuracy: 0.7268 - val_loss: 0.6424
Epoch 295/600
148/148 2s 12ms/step - accuracy: 0.7167 - loss: 0.6526 - val_accuracy: 0.7296 - val_loss: 0.6454
Epoch 296/600
148/148 2s 11ms/step - accuracy: 0.7175 - loss: 0.6545 - val_accuracy: 0.7263 - val_loss: 0.6508
Epoch 297/600
148/148 2s 12ms/step - accuracy: 0.7155 - loss: 0.6522 - val_accuracy: 0.7296 - val_loss: 0.6428
Epoch 298/600
148/148 2s 10ms/step - accuracy: 0.7185 - loss: 0.6530 - val_accuracy: 0.7315 - val_loss: 0.6399
Epoch 299/600
148/148 2s 11ms/step - accuracy: 0.7208 - loss: 0.6502 - val_accuracy: 0.7309 - val_loss: 0.6439
Epoch 300/600
148/148 2s 12ms/step - accuracy: 0.7223 - loss: 0.6460 - val_accuracy: 0.7322 - val_loss: 0.6414
Epoch 301/600
148/148 2s 10ms/step - accuracy: 0.7201 - loss: 0.6490 - val_accuracy: 0.7246 - val_loss: 0.6491
Epoch 302/600
148/148 2s 10ms/step - accuracy: 0.7201 - loss: 0.6481 - val_accuracy: 0.7293 - val_loss: 0.6446
Epoch 303/600
148/148 2s 10ms/step - accuracy: 0.7204 - loss: 0.6478 - val_accuracy: 0.7316 - val_loss: 0.6408
Epoch 304/600
148/148 1s 10ms/step - accuracy: 0.7211 - loss: 0.6489 - val_accuracy: 0.7310 - val_loss: 0.6458

Epoch 305/600
148/148 2s 10ms/step - accuracy: 0.7214 - loss: 0.6473 - val_accuracy: 0.7306 - val_loss: 0.6413
Epoch 306/600
148/148 2s 11ms/step - accuracy: 0.7216 - loss: 0.6480 - val_accuracy: 0.7334 - val_loss: 0.6428
Epoch 307/600
148/148 2s 11ms/step - accuracy: 0.7189 - loss: 0.6503 - val_accuracy: 0.7340 - val_loss: 0.6374
Epoch 308/600
148/148 2s 11ms/step - accuracy: 0.7194 - loss: 0.6472 - val_accuracy: 0.7297 - val_loss: 0.6383
Epoch 309/600
148/148 2s 10ms/step - accuracy: 0.7180 - loss: 0.6488 - val_accuracy: 0.7312 - val_loss: 0.6429
Epoch 310/600
148/148 1s 10ms/step - accuracy: 0.7195 - loss: 0.6466 - val_accuracy: 0.7306 - val_loss: 0.6448
Epoch 311/600
148/148 2s 10ms/step - accuracy: 0.7266 - loss: 0.6425 - val_accuracy: 0.7290 - val_loss: 0.6472
Epoch 312/600
148/148 1s 10ms/step - accuracy: 0.7214 - loss: 0.6492 - val_accuracy: 0.7297 - val_loss: 0.6396
Epoch 313/600
148/148 1s 10ms/step - accuracy: 0.7218 - loss: 0.6487 - val_accuracy: 0.7310 - val_loss: 0.6390
Epoch 314/600
148/148 2s 10ms/step - accuracy: 0.7198 - loss: 0.6497 - val_accuracy: 0.7319 - val_loss: 0.6352
Epoch 315/600
148/148 2s 11ms/step - accuracy: 0.7217 - loss: 0.6498 - val_accuracy: 0.7296 - val_loss: 0.6413
Epoch 316/600
148/148 2s 11ms/step - accuracy: 0.7220 - loss: 0.6465 - val_accuracy: 0.7282 - val_loss: 0.6437
Epoch 317/600
148/148 2s 11ms/step - accuracy: 0.7234 - loss: 0.6498 - val_accuracy: 0.7297 - val_loss: 0.6422
Epoch 318/600
148/148 1s 10ms/step - accuracy: 0.7212 - loss: 0.6462 - val_accuracy: 0.7316 - val_loss: 0.6434
Epoch 319/600
148/148 2s 10ms/step - accuracy: 0.7237 - loss: 0.6441 - val_accuracy: 0.7326 - val_loss: 0.6378
Epoch 320/600
148/148 1s 10ms/step - accuracy: 0.7214 - loss: 0.6488 - val_accuracy: 0.7254 - val_loss: 0.6480
Epoch 321/600
148/148 2s 10ms/step - accuracy: 0.7233 - loss: 0.6452 - val_accuracy: 0.7354 - val_loss: 0.6357
Epoch 322/600
148/148 1s 10ms/step - accuracy: 0.7198 - loss: 0.6462 - val_accuracy: 0.7309 - val_loss: 0.6394
Epoch 323/600
148/148 2s 11ms/step - accuracy: 0.7240 - loss: 0.6424 - val_accuracy: 0.7259 - val_loss: 0.6513
Epoch 324/600
148/148 2s 11ms/step - accuracy: 0.7237 - loss: 0.6453 - val_accuracy: 0.7292 - val_loss: 0.6413
Epoch 325/600
148/148 2s 11ms/step - accuracy: 0.7213 - loss: 0.6475 - val_accuracy: 0.7350 - val_loss: 0.6366
Epoch 326/600
148/148 2s 10ms/step - accuracy: 0.7241 - loss: 0.6441 - val_accuracy: 0.7335 - val_loss: 0.6378
Epoch 327/600
148/148 2s 11ms/step - accuracy: 0.7228 - loss: 0.6483 - val_accuracy: 0.7371 - val_loss: 0.6366
Epoch 328/600
148/148 2s 11ms/step - accuracy: 0.7250 - loss: 0.6420 - val_accuracy: 0.7286 - val_loss: 0.6409
Epoch 329/600
148/148 2s 11ms/step - accuracy: 0.7205 - loss: 0.6469 - val_accuracy: 0.7326 - val_loss: 0.6350
Epoch 330/600

148/148 2s 11ms/step - accuracy: 0.7193 - loss: 0.6470 - val_accuracy: 0.7357 - val_loss: 0.6371
Epoch 331/600
148/148 2s 10ms/step - accuracy: 0.7216 - loss: 0.6458 - val_accuracy: 0.7289 - val_loss: 0.6433
Epoch 332/600
148/148 1s 10ms/step - accuracy: 0.7231 - loss: 0.6444 - val_accuracy: 0.7316 - val_loss: 0.6410
Epoch 333/600
148/148 2s 11ms/step - accuracy: 0.7232 - loss: 0.6428 - val_accuracy: 0.7390 - val_loss: 0.6332
Epoch 334/600
148/148 2s 10ms/step - accuracy: 0.7207 - loss: 0.6477 - val_accuracy: 0.7324 - val_loss: 0.6362
Epoch 335/600
148/148 2s 11ms/step - accuracy: 0.7228 - loss: 0.6435 - val_accuracy: 0.7318 - val_loss: 0.6394
Epoch 336/600
148/148 2s 11ms/step - accuracy: 0.7238 - loss: 0.6398 - val_accuracy: 0.7351 - val_loss: 0.6332
Epoch 337/600
148/148 2s 11ms/step - accuracy: 0.7229 - loss: 0.6434 - val_accuracy: 0.7340 - val_loss: 0.6346
Epoch 338/600
148/148 2s 10ms/step - accuracy: 0.7218 - loss: 0.6475 - val_accuracy: 0.7299 - val_loss: 0.6414
Epoch 339/600
148/148 2s 10ms/step - accuracy: 0.7238 - loss: 0.6463 - val_accuracy: 0.7325 - val_loss: 0.6407
Epoch 340/600
148/148 2s 10ms/step - accuracy: 0.7230 - loss: 0.6470 - val_accuracy: 0.7370 - val_loss: 0.6347
Epoch 341/600
148/148 2s 10ms/step - accuracy: 0.7247 - loss: 0.6405 - val_accuracy: 0.7350 - val_loss: 0.6325
Epoch 342/600
148/148 2s 10ms/step - accuracy: 0.7215 - loss: 0.6459 - val_accuracy: 0.7297 - val_loss: 0.6473
Epoch 343/600
148/148 2s 10ms/step - accuracy: 0.7229 - loss: 0.6473 - val_accuracy: 0.7339 - val_loss: 0.6329
Epoch 344/600
148/148 1s 10ms/step - accuracy: 0.7244 - loss: 0.6411 - val_accuracy: 0.7278 - val_loss: 0.6456
Epoch 345/600
148/148 2s 10ms/step - accuracy: 0.7212 - loss: 0.6491 - val_accuracy: 0.7343 - val_loss: 0.6369
Epoch 346/600
148/148 3s 11ms/step - accuracy: 0.7271 - loss: 0.6423 - val_accuracy: 0.7303 - val_loss: 0.6388
Epoch 347/600
148/148 2s 10ms/step - accuracy: 0.7247 - loss: 0.6433 - val_accuracy: 0.7334 - val_loss: 0.6374
Epoch 348/600
148/148 1s 10ms/step - accuracy: 0.7228 - loss: 0.6435 - val_accuracy: 0.7244 - val_loss: 0.6524
Epoch 349/600
148/148 2s 10ms/step - accuracy: 0.7242 - loss: 0.6439 - val_accuracy: 0.7316 - val_loss: 0.6404
Epoch 350/600
148/148 2s 10ms/step - accuracy: 0.7242 - loss: 0.6414 - val_accuracy: 0.7335 - val_loss: 0.6355
Epoch 351/600
148/148 2s 11ms/step - accuracy: 0.7249 - loss: 0.6442 - val_accuracy: 0.7376 - val_loss: 0.6270
Epoch 352/600
148/148 2s 11ms/step - accuracy: 0.7215 - loss: 0.6463 - val_accuracy: 0.7361 - val_loss: 0.6325
Epoch 353/600
148/148 2s 10ms/step - accuracy: 0.7222 - loss: 0.6468 - val_accuracy: 0.7356 - val_loss: 0.6344
Epoch 354/600
148/148 2s 10ms/step - accuracy: 0.7266 - loss: 0.6431 - val_accuracy: 0.7382 - val_loss: 0.6312
Epoch 355/600
148/148 2s 12ms/step - accuracy: 0.7189 - loss: 0.6463 - val_accuracy: 0.7336 - val_loss:

s: 0.6367
Epoch 356/600
148/148 2s 11ms/step - accuracy: 0.7224 - loss: 0.6444 - val_accuracy: 0.7327 - val_loss: 0.6403
s: 0.6403
Epoch 357/600
148/148 2s 10ms/step - accuracy: 0.7237 - loss: 0.6427 - val_accuracy: 0.7342 - val_loss: 0.6383
s: 0.6383
Epoch 358/600
148/148 2s 10ms/step - accuracy: 0.7243 - loss: 0.6426 - val_accuracy: 0.7357 - val_loss: 0.6315
s: 0.6315
Epoch 359/600
148/148 2s 10ms/step - accuracy: 0.7245 - loss: 0.6460 - val_accuracy: 0.7361 - val_loss: 0.6325
s: 0.6325
Epoch 360/600
148/148 2s 10ms/step - accuracy: 0.7249 - loss: 0.6421 - val_accuracy: 0.7404 - val_loss: 0.6296
s: 0.6296
Epoch 361/600
148/148 1s 10ms/step - accuracy: 0.7232 - loss: 0.6443 - val_accuracy: 0.7367 - val_loss: 0.6325
s: 0.6325
Epoch 362/600
148/148 2s 10ms/step - accuracy: 0.7223 - loss: 0.6452 - val_accuracy: 0.7302 - val_loss: 0.6393
s: 0.6393
Epoch 363/600
148/148 2s 11ms/step - accuracy: 0.7221 - loss: 0.6443 - val_accuracy: 0.7315 - val_loss: 0.6304
s: 0.6304
Epoch 364/600
148/148 2s 11ms/step - accuracy: 0.7212 - loss: 0.6441 - val_accuracy: 0.7368 - val_loss: 0.6322
s: 0.6322
Epoch 365/600
148/148 2s 11ms/step - accuracy: 0.7236 - loss: 0.6409 - val_accuracy: 0.7337 - val_loss: 0.6361
s: 0.6361
Epoch 366/600
148/148 2s 11ms/step - accuracy: 0.7233 - loss: 0.6420 - val_accuracy: 0.7350 - val_loss: 0.6348
s: 0.6348
Epoch 367/600
148/148 2s 11ms/step - accuracy: 0.7244 - loss: 0.6430 - val_accuracy: 0.7370 - val_loss: 0.6324
s: 0.6324
Epoch 368/600
148/148 2s 11ms/step - accuracy: 0.7264 - loss: 0.6383 - val_accuracy: 0.7327 - val_loss: 0.6380
s: 0.6380
Epoch 369/600
148/148 2s 11ms/step - accuracy: 0.7258 - loss: 0.6423 - val_accuracy: 0.7341 - val_loss: 0.6363
s: 0.6363
Epoch 370/600
148/148 2s 10ms/step - accuracy: 0.7266 - loss: 0.6355 - val_accuracy: 0.7324 - val_loss: 0.6361
s: 0.6361
Epoch 371/600
148/148 2s 11ms/step - accuracy: 0.7200 - loss: 0.6471 - val_accuracy: 0.7332 - val_loss: 0.6324
s: 0.6324
Epoch 372/600
148/148 2s 12ms/step - accuracy: 0.7256 - loss: 0.6380 - val_accuracy: 0.7371 - val_loss: 0.6295
s: 0.6295
Epoch 373/600
148/148 2s 12ms/step - accuracy: 0.7226 - loss: 0.6439 - val_accuracy: 0.7405 - val_loss: 0.6284
s: 0.6284
Epoch 374/600
148/148 2s 12ms/step - accuracy: 0.7296 - loss: 0.6341 - val_accuracy: 0.7390 - val_loss: 0.6343
s: 0.6343
Epoch 375/600
148/148 2s 13ms/step - accuracy: 0.7279 - loss: 0.6364 - val_accuracy: 0.7380 - val_loss: 0.6313
s: 0.6313
Epoch 376/600
148/148 2s 12ms/step - accuracy: 0.7249 - loss: 0.6421 - val_accuracy: 0.7387 - val_loss: 0.6300
s: 0.6300
Epoch 377/600
148/148 2s 11ms/step - accuracy: 0.7240 - loss: 0.6398 - val_accuracy: 0.7350 - val_loss: 0.6379
s: 0.6379
Epoch 378/600
148/148 2s 11ms/step - accuracy: 0.7250 - loss: 0.6413 - val_accuracy: 0.7383 - val_loss: 0.6278
s: 0.6278
Epoch 379/600
148/148 2s 11ms/step - accuracy: 0.7292 - loss: 0.6350 - val_accuracy: 0.7361 - val_loss: 0.6306
s: 0.6306
Epoch 380/600
148/148 2s 11ms/step - accuracy: 0.7246 - loss: 0.6404 - val_accuracy: 0.7343 - val_loss: 0.6313
s: 0.6313

Epoch 381/600
148/148 2s 11ms/step - accuracy: 0.7291 - loss: 0.6380 - val_accuracy: 0.7338 - val_loss: 0.6348
Epoch 382/600
148/148 2s 12ms/step - accuracy: 0.7289 - loss: 0.6385 - val_accuracy: 0.7313 - val_loss: 0.6368
Epoch 383/600
148/148 2s 12ms/step - accuracy: 0.7280 - loss: 0.6392 - val_accuracy: 0.7357 - val_loss: 0.6365
Epoch 384/600
148/148 2s 11ms/step - accuracy: 0.7243 - loss: 0.6402 - val_accuracy: 0.7378 - val_loss: 0.6308
Epoch 385/600
148/148 2s 11ms/step - accuracy: 0.7238 - loss: 0.6408 - val_accuracy: 0.7393 - val_loss: 0.6276
Epoch 386/600
148/148 2s 11ms/step - accuracy: 0.7254 - loss: 0.6415 - val_accuracy: 0.7417 - val_loss: 0.6266
Epoch 387/600
148/148 2s 11ms/step - accuracy: 0.7241 - loss: 0.6423 - val_accuracy: 0.7303 - val_loss: 0.6447
Epoch 388/600
148/148 2s 10ms/step - accuracy: 0.7241 - loss: 0.6434 - val_accuracy: 0.7364 - val_loss: 0.6291
Epoch 389/600
148/148 2s 10ms/step - accuracy: 0.7241 - loss: 0.6422 - val_accuracy: 0.7378 - val_loss: 0.6306
Epoch 390/600
148/148 2s 10ms/step - accuracy: 0.7287 - loss: 0.6380 - val_accuracy: 0.7307 - val_loss: 0.6371
Epoch 391/600
148/148 2s 11ms/step - accuracy: 0.7295 - loss: 0.6344 - val_accuracy: 0.7318 - val_loss: 0.6410
Epoch 392/600
148/148 2s 10ms/step - accuracy: 0.7253 - loss: 0.6393 - val_accuracy: 0.7256 - val_loss: 0.6534
Epoch 393/600
148/148 2s 11ms/step - accuracy: 0.7235 - loss: 0.6399 - val_accuracy: 0.7350 - val_loss: 0.6337
Epoch 394/600
148/148 2s 11ms/step - accuracy: 0.7258 - loss: 0.6401 - val_accuracy: 0.7398 - val_loss: 0.6286
Epoch 395/600
148/148 2s 11ms/step - accuracy: 0.7269 - loss: 0.6398 - val_accuracy: 0.7430 - val_loss: 0.6261
Epoch 396/600
148/148 2s 10ms/step - accuracy: 0.7285 - loss: 0.6339 - val_accuracy: 0.7358 - val_loss: 0.6300
Epoch 397/600
148/148 2s 10ms/step - accuracy: 0.7257 - loss: 0.6406 - val_accuracy: 0.7379 - val_loss: 0.6313
Epoch 398/600
148/148 2s 10ms/step - accuracy: 0.7252 - loss: 0.6416 - val_accuracy: 0.7359 - val_loss: 0.6292
Epoch 399/600
148/148 2s 11ms/step - accuracy: 0.7304 - loss: 0.6373 - val_accuracy: 0.7398 - val_loss: 0.6299
Epoch 400/600
148/148 1s 10ms/step - accuracy: 0.7262 - loss: 0.6411 - val_accuracy: 0.7386 - val_loss: 0.6302
Epoch 401/600
148/148 2s 10ms/step - accuracy: 0.7224 - loss: 0.6426 - val_accuracy: 0.7390 - val_loss: 0.6262
Epoch 402/600
148/148 2s 10ms/step - accuracy: 0.7287 - loss: 0.6348 - val_accuracy: 0.7409 - val_loss: 0.6276
Epoch 403/600
148/148 2s 11ms/step - accuracy: 0.7275 - loss: 0.6348 - val_accuracy: 0.7393 - val_loss: 0.6288
Epoch 404/600
148/148 3s 11ms/step - accuracy: 0.7250 - loss: 0.6415 - val_accuracy: 0.7397 - val_loss: 0.6321
Epoch 405/600
148/148 2s 10ms/step - accuracy: 0.7243 - loss: 0.6428 - val_accuracy: 0.7418 - val_loss: 0.6318
Epoch 406/600

148/148 2s 10ms/step - accuracy: 0.7298 - loss: 0.6361 - val_accuracy: 0.7424 - val_loss: 0.6245
Epoch 407/600
148/148 2s 10ms/step - accuracy: 0.7281 - loss: 0.6363 - val_accuracy: 0.7367 - val_loss: 0.6288
Epoch 408/600
148/148 2s 10ms/step - accuracy: 0.7245 - loss: 0.6388 - val_accuracy: 0.7421 - val_loss: 0.6261
Epoch 409/600
148/148 2s 12ms/step - accuracy: 0.7272 - loss: 0.6404 - val_accuracy: 0.7363 - val_loss: 0.6330
Epoch 410/600
148/148 2s 11ms/step - accuracy: 0.7279 - loss: 0.6371 - val_accuracy: 0.7362 - val_loss: 0.6325
Epoch 411/600
148/148 2s 11ms/step - accuracy: 0.7257 - loss: 0.6371 - val_accuracy: 0.7352 - val_loss: 0.6338
Epoch 412/600
148/148 2s 11ms/step - accuracy: 0.7245 - loss: 0.6379 - val_accuracy: 0.7405 - val_loss: 0.6270
Epoch 413/600
148/148 2s 11ms/step - accuracy: 0.7295 - loss: 0.6352 - val_accuracy: 0.7408 - val_loss: 0.6266
Epoch 414/600
148/148 2s 11ms/step - accuracy: 0.7220 - loss: 0.6428 - val_accuracy: 0.7404 - val_loss: 0.6267
Epoch 415/600
148/148 2s 10ms/step - accuracy: 0.7286 - loss: 0.6426 - val_accuracy: 0.7430 - val_loss: 0.6259
Epoch 416/600
148/148 2s 11ms/step - accuracy: 0.7275 - loss: 0.6360 - val_accuracy: 0.7444 - val_loss: 0.6212
Epoch 417/600
148/148 2s 10ms/step - accuracy: 0.7295 - loss: 0.6346 - val_accuracy: 0.7449 - val_loss: 0.6232
Epoch 418/600
148/148 2s 11ms/step - accuracy: 0.7277 - loss: 0.6370 - val_accuracy: 0.7415 - val_loss: 0.6264
Epoch 419/600
148/148 2s 11ms/step - accuracy: 0.7284 - loss: 0.6352 - val_accuracy: 0.7412 - val_loss: 0.6232
Epoch 420/600
148/148 2s 11ms/step - accuracy: 0.7252 - loss: 0.6379 - val_accuracy: 0.7427 - val_loss: 0.6240
Epoch 421/600
148/148 2s 11ms/step - accuracy: 0.7274 - loss: 0.6358 - val_accuracy: 0.7402 - val_loss: 0.6275
Epoch 422/600
148/148 2s 11ms/step - accuracy: 0.7299 - loss: 0.6334 - val_accuracy: 0.7408 - val_loss: 0.6251
Epoch 423/600
148/148 2s 12ms/step - accuracy: 0.7252 - loss: 0.6391 - val_accuracy: 0.7376 - val_loss: 0.6300
Epoch 424/600
148/148 2s 11ms/step - accuracy: 0.7240 - loss: 0.6409 - val_accuracy: 0.7369 - val_loss: 0.6357
Epoch 425/600
148/148 2s 10ms/step - accuracy: 0.7267 - loss: 0.6377 - val_accuracy: 0.7413 - val_loss: 0.6255
Epoch 426/600
148/148 1s 10ms/step - accuracy: 0.7273 - loss: 0.6356 - val_accuracy: 0.7410 - val_loss: 0.6268
Epoch 427/600
148/148 2s 10ms/step - accuracy: 0.7305 - loss: 0.6342 - val_accuracy: 0.7406 - val_loss: 0.6246
Epoch 428/600
148/148 2s 10ms/step - accuracy: 0.7255 - loss: 0.6383 - val_accuracy: 0.7391 - val_loss: 0.6311
Epoch 429/600
148/148 2s 11ms/step - accuracy: 0.7309 - loss: 0.6323 - val_accuracy: 0.7380 - val_loss: 0.6308
Epoch 430/600
148/148 2s 11ms/step - accuracy: 0.7289 - loss: 0.6371 - val_accuracy: 0.7443 - val_loss: 0.6235
Epoch 431/600
148/148 2s 11ms/step - accuracy: 0.7306 - loss: 0.6380 - val_accuracy: 0.7381 - val_loss:

s: 0.6273
Epoch 432/600
148/148 2s 11ms/step - accuracy: 0.7300 - loss: 0.6360 - val_accuracy: 0.7430 - val_loss: 0.6250
Epoch 433/600
148/148 2s 12ms/step - accuracy: 0.7282 - loss: 0.6400 - val_accuracy: 0.7398 - val_loss: 0.6264
Epoch 434/600
148/148 2s 11ms/step - accuracy: 0.7290 - loss: 0.6357 - val_accuracy: 0.7448 - val_loss: 0.6205
s: 0.6230
Epoch 435/600
148/148 2s 11ms/step - accuracy: 0.7318 - loss: 0.6330 - val_accuracy: 0.7458 - val_loss: 0.6240
s: 0.6267
Epoch 437/600
148/148 2s 11ms/step - accuracy: 0.7285 - loss: 0.6367 - val_accuracy: 0.7384 - val_loss: 0.6218
s: 0.6218
Epoch 439/600
148/148 2s 11ms/step - accuracy: 0.7308 - loss: 0.6333 - val_accuracy: 0.7435 - val_loss: 0.6249
s: 0.6263
Epoch 440/600
148/148 2s 12ms/step - accuracy: 0.7311 - loss: 0.6319 - val_accuracy: 0.7391 - val_loss: 0.6218
s: 0.6249
Epoch 441/600
148/148 2s 12ms/step - accuracy: 0.7307 - loss: 0.6337 - val_accuracy: 0.7414 - val_loss: 0.6330
s: 0.6330
Epoch 443/600
148/148 2s 11ms/step - accuracy: 0.7277 - loss: 0.6363 - val_accuracy: 0.7406 - val_loss: 0.6343
s: 0.6276
Epoch 445/600
148/148 2s 11ms/step - accuracy: 0.7266 - loss: 0.6351 - val_accuracy: 0.7389 - val_loss: 0.6254
s: 0.6254
Epoch 447/600
148/148 2s 13ms/step - accuracy: 0.7307 - loss: 0.6332 - val_accuracy: 0.7408 - val_loss: 0.6302
s: 0.6219
Epoch 448/600
148/148 2s 11ms/step - accuracy: 0.7271 - loss: 0.6402 - val_accuracy: 0.7441 - val_loss: 0.6219
s: 0.6171
Epoch 451/600
148/148 2s 10ms/step - accuracy: 0.7270 - loss: 0.6324 - val_accuracy: 0.7455 - val_loss: 0.6334
s: 0.6242
Epoch 452/600
148/148 2s 11ms/step - accuracy: 0.7282 - loss: 0.6370 - val_accuracy: 0.7418 - val_loss: 0.6203
s: 0.6203
Epoch 454/600
148/148 2s 10ms/step - accuracy: 0.7295 - loss: 0.6348 - val_accuracy: 0.7472 - val_loss: 0.6244
s: 0.6244
Epoch 455/600
148/148 1s 10ms/step - accuracy: 0.7261 - loss: 0.6405 - val_accuracy: 0.7432 - val_loss: 0.6206
s: 0.6304

Epoch 457/600
148/148 1s 10ms/step - accuracy: 0.7324 - loss: 0.6348 - val_accuracy: 0.7415 - val_loss: 0.6221
Epoch 458/600
148/148 2s 10ms/step - accuracy: 0.7304 - loss: 0.6374 - val_accuracy: 0.7365 - val_loss: 0.6331
Epoch 459/600
148/148 1s 10ms/step - accuracy: 0.7304 - loss: 0.6336 - val_accuracy: 0.7437 - val_loss: 0.6287
s: 0.6287
Epoch 460/600
148/148 2s 10ms/step - accuracy: 0.7279 - loss: 0.6353 - val_accuracy: 0.7324 - val_loss: 0.6404
s: 0.6404
Epoch 461/600
148/148 2s 11ms/step - accuracy: 0.7312 - loss: 0.6315 - val_accuracy: 0.7472 - val_loss: 0.6221
s: 0.6221
Epoch 462/600
148/148 2s 11ms/step - accuracy: 0.7287 - loss: 0.6351 - val_accuracy: 0.7418 - val_loss: 0.6243
s: 0.6243
Epoch 463/600
148/148 2s 11ms/step - accuracy: 0.7277 - loss: 0.6304 - val_accuracy: 0.7473 - val_loss: 0.6222
s: 0.6222
Epoch 464/600
148/148 2s 10ms/step - accuracy: 0.7317 - loss: 0.6349 - val_accuracy: 0.7439 - val_loss: 0.6220
s: 0.6220
Epoch 465/600
148/148 2s 10ms/step - accuracy: 0.7302 - loss: 0.6391 - val_accuracy: 0.7468 - val_loss: 0.6177
s: 0.6177
Epoch 466/600
148/148 2s 10ms/step - accuracy: 0.7292 - loss: 0.6348 - val_accuracy: 0.7383 - val_loss: 0.6268
s: 0.6268
Epoch 467/600
148/148 1s 10ms/step - accuracy: 0.7274 - loss: 0.6371 - val_accuracy: 0.7413 - val_loss: 0.6225
s: 0.6225
Epoch 468/600
148/148 2s 11ms/step - accuracy: 0.7263 - loss: 0.6367 - val_accuracy: 0.7446 - val_loss: 0.6209
s: 0.6209
Epoch 469/600
148/148 2s 14ms/step - accuracy: 0.7314 - loss: 0.6325 - val_accuracy: 0.7388 - val_loss: 0.6299
s: 0.6299
Epoch 470/600
148/148 2s 13ms/step - accuracy: 0.7277 - loss: 0.6335 - val_accuracy: 0.7437 - val_loss: 0.6186
s: 0.6186
Epoch 471/600
148/148 2s 12ms/step - accuracy: 0.7283 - loss: 0.6351 - val_accuracy: 0.7417 - val_loss: 0.6231
s: 0.6231
Epoch 472/600
148/148 2s 12ms/step - accuracy: 0.7294 - loss: 0.6355 - val_accuracy: 0.7419 - val_loss: 0.6223
s: 0.6223
Epoch 473/600
148/148 2s 11ms/step - accuracy: 0.7298 - loss: 0.6336 - val_accuracy: 0.7468 - val_loss: 0.6232
s: 0.6232
Epoch 474/600
148/148 2s 11ms/step - accuracy: 0.7325 - loss: 0.6315 - val_accuracy: 0.7376 - val_loss: 0.6309
s: 0.6309
Epoch 475/600
148/148 2s 11ms/step - accuracy: 0.7307 - loss: 0.6309 - val_accuracy: 0.7437 - val_loss: 0.6236
s: 0.6236
Epoch 476/600
148/148 2s 11ms/step - accuracy: 0.7289 - loss: 0.6348 - val_accuracy: 0.7367 - val_loss: 0.6339
s: 0.6339
Epoch 477/600
148/148 2s 12ms/step - accuracy: 0.7291 - loss: 0.6333 - val_accuracy: 0.7403 - val_loss: 0.6306
s: 0.6306
Epoch 478/600
148/148 2s 10ms/step - accuracy: 0.7279 - loss: 0.6374 - val_accuracy: 0.7416 - val_loss: 0.6238
s: 0.6238
Epoch 479/600
148/148 2s 10ms/step - accuracy: 0.7312 - loss: 0.6325 - val_accuracy: 0.7447 - val_loss: 0.6220
s: 0.6220
Epoch 480/600
148/148 2s 10ms/step - accuracy: 0.7285 - loss: 0.6353 - val_accuracy: 0.7433 - val_loss: 0.6246
s: 0.6246
Epoch 481/600
148/148 2s 11ms/step - accuracy: 0.7313 - loss: 0.6338 - val_accuracy: 0.7455 - val_loss: 0.6203
s: 0.6203
Epoch 482/600

148/148 2s 11ms/step - accuracy: 0.7309 - loss: 0.6322 - val_accuracy: 0.7445 - val_loss: 0.6196
Epoch 483/600
148/148 2s 10ms/step - accuracy: 0.7299 - loss: 0.6371 - val_accuracy: 0.7411 - val_loss: 0.6265
Epoch 484/600
148/148 1s 10ms/step - accuracy: 0.7296 - loss: 0.6334 - val_accuracy: 0.7434 - val_loss: 0.6226
Epoch 485/600
148/148 2s 10ms/step - accuracy: 0.7322 - loss: 0.6329 - val_accuracy: 0.7421 - val_loss: 0.6222
Epoch 486/600
148/148 2s 10ms/step - accuracy: 0.7274 - loss: 0.6353 - val_accuracy: 0.7427 - val_loss: 0.6234
Epoch 487/600
148/148 2s 11ms/step - accuracy: 0.7321 - loss: 0.6316 - val_accuracy: 0.7383 - val_loss: 0.6305
Epoch 488/600
148/148 2s 11ms/step - accuracy: 0.7291 - loss: 0.6355 - val_accuracy: 0.7437 - val_loss: 0.6235
Epoch 489/600
148/148 2s 11ms/step - accuracy: 0.7321 - loss: 0.6304 - val_accuracy: 0.7408 - val_loss: 0.6298
Epoch 490/600
148/148 2s 12ms/step - accuracy: 0.7308 - loss: 0.6344 - val_accuracy: 0.7425 - val_loss: 0.6273
Epoch 491/600
148/148 2s 11ms/step - accuracy: 0.7313 - loss: 0.6329 - val_accuracy: 0.7358 - val_loss: 0.6374
Epoch 492/600
148/148 2s 11ms/step - accuracy: 0.7318 - loss: 0.6315 - val_accuracy: 0.7388 - val_loss: 0.6275
Epoch 493/600
148/148 2s 11ms/step - accuracy: 0.7315 - loss: 0.6322 - val_accuracy: 0.7465 - val_loss: 0.6152
Epoch 494/600
148/148 2s 10ms/step - accuracy: 0.7311 - loss: 0.6306 - val_accuracy: 0.7398 - val_loss: 0.6284
Epoch 495/600
148/148 2s 11ms/step - accuracy: 0.7338 - loss: 0.6262 - val_accuracy: 0.7440 - val_loss: 0.6216
Epoch 496/600
148/148 2s 10ms/step - accuracy: 0.7345 - loss: 0.6294 - val_accuracy: 0.7436 - val_loss: 0.6199
Epoch 497/600
148/148 2s 11ms/step - accuracy: 0.7311 - loss: 0.6317 - val_accuracy: 0.7462 - val_loss: 0.6188
Epoch 498/600
148/148 2s 11ms/step - accuracy: 0.7294 - loss: 0.6331 - val_accuracy: 0.7459 - val_loss: 0.6202
Epoch 499/600
148/148 2s 11ms/step - accuracy: 0.7320 - loss: 0.6325 - val_accuracy: 0.7386 - val_loss: 0.6277
Epoch 500/600
148/148 2s 10ms/step - accuracy: 0.7289 - loss: 0.6327 - val_accuracy: 0.7435 - val_loss: 0.6205
Epoch 501/600
148/148 2s 12ms/step - accuracy: 0.7294 - loss: 0.6328 - val_accuracy: 0.7456 - val_loss: 0.6237
Epoch 502/600
148/148 2s 12ms/step - accuracy: 0.7283 - loss: 0.6343 - val_accuracy: 0.7444 - val_loss: 0.6184
Epoch 503/600
148/148 2s 10ms/step - accuracy: 0.7281 - loss: 0.6350 - val_accuracy: 0.7489 - val_loss: 0.6153
Epoch 504/600
148/148 2s 11ms/step - accuracy: 0.7275 - loss: 0.6335 - val_accuracy: 0.7461 - val_loss: 0.6211
Epoch 505/600
148/148 2s 10ms/step - accuracy: 0.7330 - loss: 0.6292 - val_accuracy: 0.7455 - val_loss: 0.6215
Epoch 506/600
148/148 2s 10ms/step - accuracy: 0.7350 - loss: 0.6248 - val_accuracy: 0.7505 - val_loss: 0.6169
Epoch 507/600
148/148 2s 10ms/step - accuracy: 0.7356 - loss: 0.6264 - val_accuracy: 0.7502 - val_loss:

s: 0.6145
Epoch 508/600
148/148 2s 10ms/step - accuracy: 0.7287 - loss: 0.6315 - val_accuracy: 0.7426 - val_loss: 0.6196
Epoch 509/600
148/148 2s 11ms/step - accuracy: 0.7317 - loss: 0.6298 - val_accuracy: 0.7453 - val_loss: 0.6213
Epoch 510/600
148/148 2s 12ms/step - accuracy: 0.7331 - loss: 0.6315 - val_accuracy: 0.7372 - val_loss: 0.6326
Epoch 511/600
148/148 2s 11ms/step - accuracy: 0.7333 - loss: 0.6304 - val_accuracy: 0.7463 - val_loss: 0.6180
Epoch 512/600
148/148 2s 11ms/step - accuracy: 0.7333 - loss: 0.6297 - val_accuracy: 0.7489 - val_loss: 0.6185
Epoch 513/600
148/148 2s 11ms/step - accuracy: 0.7305 - loss: 0.6353 - val_accuracy: 0.7433 - val_loss: 0.6213
Epoch 514/600
148/148 2s 12ms/step - accuracy: 0.7275 - loss: 0.6381 - val_accuracy: 0.7450 - val_loss: 0.6224
Epoch 515/600
148/148 2s 12ms/step - accuracy: 0.7331 - loss: 0.6280 - val_accuracy: 0.7404 - val_loss: 0.6305
Epoch 516/600
148/148 2s 10ms/step - accuracy: 0.7312 - loss: 0.6285 - val_accuracy: 0.7498 - val_loss: 0.6146
Epoch 517/600
148/148 2s 11ms/step - accuracy: 0.7323 - loss: 0.6276 - val_accuracy: 0.7412 - val_loss: 0.6210
Epoch 518/600
148/148 2s 10ms/step - accuracy: 0.7324 - loss: 0.6310 - val_accuracy: 0.7431 - val_loss: 0.6263
Epoch 519/600
148/148 2s 10ms/step - accuracy: 0.7305 - loss: 0.6330 - val_accuracy: 0.7464 - val_loss: 0.6188
Epoch 520/600
148/148 2s 12ms/step - accuracy: 0.7311 - loss: 0.6296 - val_accuracy: 0.7450 - val_loss: 0.6171
Epoch 521/600
148/148 2s 12ms/step - accuracy: 0.7311 - loss: 0.6326 - val_accuracy: 0.7434 - val_loss: 0.6205
Epoch 522/600
148/148 2s 11ms/step - accuracy: 0.7319 - loss: 0.6327 - val_accuracy: 0.7452 - val_loss: 0.6177
Epoch 523/600
148/148 2s 10ms/step - accuracy: 0.7338 - loss: 0.6264 - val_accuracy: 0.7455 - val_loss: 0.6174
Epoch 524/600
148/148 2s 12ms/step - accuracy: 0.7332 - loss: 0.6308 - val_accuracy: 0.7466 - val_loss: 0.6129
Epoch 525/600
148/148 2s 11ms/step - accuracy: 0.7322 - loss: 0.6322 - val_accuracy: 0.7437 - val_loss: 0.6227
Epoch 526/600
148/148 2s 12ms/step - accuracy: 0.7320 - loss: 0.6282 - val_accuracy: 0.7453 - val_loss: 0.6193
Epoch 527/600
148/148 2s 11ms/step - accuracy: 0.7339 - loss: 0.6290 - val_accuracy: 0.7437 - val_loss: 0.6155
Epoch 528/600
148/148 2s 11ms/step - accuracy: 0.7303 - loss: 0.6330 - val_accuracy: 0.7473 - val_loss: 0.6170
Epoch 529/600
148/148 2s 12ms/step - accuracy: 0.7351 - loss: 0.6246 - val_accuracy: 0.7505 - val_loss: 0.6162
Epoch 530/600
148/148 2s 11ms/step - accuracy: 0.7307 - loss: 0.6315 - val_accuracy: 0.7472 - val_loss: 0.6169
Epoch 531/600
148/148 2s 12ms/step - accuracy: 0.7298 - loss: 0.6329 - val_accuracy: 0.7467 - val_loss: 0.6198
Epoch 532/600
148/148 2s 11ms/step - accuracy: 0.7329 - loss: 0.6303 - val_accuracy: 0.7352 - val_loss: 0.6382

Epoch 533/600
148/148 2s 10ms/step - accuracy: 0.7321 - loss: 0.6315 - val_accuracy: 0.7470 - val_loss: 0.6164
Epoch 534/600
148/148 2s 10ms/step - accuracy: 0.7297 - loss: 0.6317 - val_accuracy: 0.7442 - val_loss: 0.6189
Epoch 535/600
148/148 2s 11ms/step - accuracy: 0.7304 - loss: 0.6326 - val_accuracy: 0.7492 - val_loss: 0.6154
Epoch 536/600
148/148 2s 10ms/step - accuracy: 0.7302 - loss: 0.6319 - val_accuracy: 0.7471 - val_loss: 0.6170
Epoch 537/600
148/148 2s 11ms/step - accuracy: 0.7312 - loss: 0.6337 - val_accuracy: 0.7442 - val_loss: 0.6228
Epoch 538/600
148/148 2s 10ms/step - accuracy: 0.7336 - loss: 0.6265 - val_accuracy: 0.7455 - val_loss: 0.6214
Epoch 539/600
148/148 2s 11ms/step - accuracy: 0.7305 - loss: 0.6322 - val_accuracy: 0.7476 - val_loss: 0.6157
Epoch 540/600
148/148 2s 11ms/step - accuracy: 0.7312 - loss: 0.6307 - val_accuracy: 0.7487 - val_loss: 0.6154
Epoch 541/600
148/148 2s 13ms/step - accuracy: 0.7318 - loss: 0.6261 - val_accuracy: 0.7429 - val_loss: 0.6239
Epoch 542/600
148/148 2s 11ms/step - accuracy: 0.7339 - loss: 0.6271 - val_accuracy: 0.7471 - val_loss: 0.6102
Epoch 543/600
148/148 2s 10ms/step - accuracy: 0.7347 - loss: 0.6280 - val_accuracy: 0.7477 - val_loss: 0.6165
Epoch 544/600
148/148 2s 11ms/step - accuracy: 0.7314 - loss: 0.6292 - val_accuracy: 0.7465 - val_loss: 0.6170
Epoch 545/600
148/148 2s 11ms/step - accuracy: 0.7311 - loss: 0.6294 - val_accuracy: 0.7468 - val_loss: 0.6166
Epoch 546/600
148/148 1s 10ms/step - accuracy: 0.7346 - loss: 0.6309 - val_accuracy: 0.7445 - val_loss: 0.6216
Epoch 547/600
148/148 2s 11ms/step - accuracy: 0.7333 - loss: 0.6256 - val_accuracy: 0.7473 - val_loss: 0.6142
Epoch 548/600
148/148 2s 11ms/step - accuracy: 0.7323 - loss: 0.6306 - val_accuracy: 0.7480 - val_loss: 0.6183
Epoch 549/600
148/148 2s 12ms/step - accuracy: 0.7330 - loss: 0.6268 - val_accuracy: 0.7501 - val_loss: 0.6129
Epoch 550/600
148/148 2s 11ms/step - accuracy: 0.7360 - loss: 0.6208 - val_accuracy: 0.7481 - val_loss: 0.6157
Epoch 551/600
148/148 2s 11ms/step - accuracy: 0.7302 - loss: 0.6321 - val_accuracy: 0.7435 - val_loss: 0.6184
Epoch 552/600
148/148 1s 10ms/step - accuracy: 0.7328 - loss: 0.6297 - val_accuracy: 0.7481 - val_loss: 0.6178
Epoch 553/600
148/148 1s 10ms/step - accuracy: 0.7332 - loss: 0.6256 - val_accuracy: 0.7487 - val_loss: 0.6173
Epoch 554/600
148/148 1s 10ms/step - accuracy: 0.7320 - loss: 0.6291 - val_accuracy: 0.7458 - val_loss: 0.6162
Epoch 555/600
148/148 2s 10ms/step - accuracy: 0.7348 - loss: 0.6291 - val_accuracy: 0.7362 - val_loss: 0.6336
Epoch 556/600
148/148 1s 10ms/step - accuracy: 0.7340 - loss: 0.6268 - val_accuracy: 0.7436 - val_loss: 0.6212
Epoch 557/600
148/148 2s 10ms/step - accuracy: 0.7339 - loss: 0.6288 - val_accuracy: 0.7487 - val_loss: 0.6163
Epoch 558/600

148/148 2s 10ms/step - accuracy: 0.7343 - loss: 0.6277 - val_accuracy: 0.7513 - val_loss: 0.6132
Epoch 559/600
148/148 2s 11ms/step - accuracy: 0.7337 - loss: 0.6294 - val_accuracy: 0.7489 - val_loss: 0.6177
Epoch 560/600
148/148 2s 11ms/step - accuracy: 0.7353 - loss: 0.6267 - val_accuracy: 0.7426 - val_loss: 0.6269
Epoch 561/600
148/148 2s 11ms/step - accuracy: 0.7361 - loss: 0.6238 - val_accuracy: 0.7477 - val_loss: 0.6142
Epoch 562/600
148/148 2s 10ms/step - accuracy: 0.7319 - loss: 0.6316 - val_accuracy: 0.7507 - val_loss: 0.6106
Epoch 563/600
148/148 2s 10ms/step - accuracy: 0.7352 - loss: 0.6263 - val_accuracy: 0.7459 - val_loss: 0.6167
Epoch 564/600
148/148 2s 10ms/step - accuracy: 0.7329 - loss: 0.6287 - val_accuracy: 0.7494 - val_loss: 0.6185
Epoch 565/600
148/148 2s 10ms/step - accuracy: 0.7337 - loss: 0.6284 - val_accuracy: 0.7483 - val_loss: 0.6147
Epoch 566/600
148/148 2s 10ms/step - accuracy: 0.7335 - loss: 0.6257 - val_accuracy: 0.7473 - val_loss: 0.6171
Epoch 567/600
148/148 1s 10ms/step - accuracy: 0.7340 - loss: 0.6265 - val_accuracy: 0.7439 - val_loss: 0.6245
Epoch 569/600
148/148 2s 10ms/step - accuracy: 0.7377 - loss: 0.6234 - val_accuracy: 0.7478 - val_loss: 0.6155
Epoch 570/600
148/148 2s 10ms/step - accuracy: 0.7330 - loss: 0.6274 - val_accuracy: 0.7485 - val_loss: 0.6171
Epoch 571/600
148/148 2s 11ms/step - accuracy: 0.7306 - loss: 0.6317 - val_accuracy: 0.7401 - val_loss: 0.6220
Epoch 572/600
148/148 2s 10ms/step - accuracy: 0.7300 - loss: 0.6317 - val_accuracy: 0.7444 - val_loss: 0.6190
Epoch 573/600
148/148 2s 10ms/step - accuracy: 0.7323 - loss: 0.6282 - val_accuracy: 0.7463 - val_loss: 0.6183
Epoch 574/600
148/148 2s 11ms/step - accuracy: 0.7298 - loss: 0.6367 - val_accuracy: 0.7384 - val_loss: 0.6336
Epoch 575/600
148/148 2s 11ms/step - accuracy: 0.7300 - loss: 0.6349 - val_accuracy: 0.7494 - val_loss: 0.6168
Epoch 576/600
148/148 1s 10ms/step - accuracy: 0.7339 - loss: 0.6298 - val_accuracy: 0.7505 - val_loss: 0.6098
Epoch 577/600
148/148 2s 10ms/step - accuracy: 0.7375 - loss: 0.6223 - val_accuracy: 0.7485 - val_loss: 0.6155
Epoch 578/600
148/148 2s 10ms/step - accuracy: 0.7340 - loss: 0.6242 - val_accuracy: 0.7483 - val_loss: 0.6175
Epoch 579/600
148/148 2s 10ms/step - accuracy: 0.7353 - loss: 0.6253 - val_accuracy: 0.7468 - val_loss: 0.6181
Epoch 580/600
148/148 2s 11ms/step - accuracy: 0.7344 - loss: 0.6248 - val_accuracy: 0.7483 - val_loss: 0.6152
Epoch 581/600
148/148 2s 12ms/step - accuracy: 0.7323 - loss: 0.6281 - val_accuracy: 0.7471 - val_loss: 0.6143
Epoch 582/600
148/148 2s 10ms/step - accuracy: 0.7354 - loss: 0.6240 - val_accuracy: 0.7485 - val_loss: 0.6163
Epoch 583/600
148/148 2s 10ms/step - accuracy: 0.7350 - loss: 0.6243 - val_accuracy: 0.7489 - val_loss:

```
s: 0.6133
Epoch 584/600
148/148 2s 10ms/step - accuracy: 0.7345 - loss: 0.6222 - val_accuracy: 0.7506 - val_loss
s: 0.6100
Epoch 585/600
148/148 2s 10ms/step - accuracy: 0.7358 - loss: 0.6212 - val_accuracy: 0.7503 - val_loss
s: 0.6112
Epoch 586/600
148/148 2s 10ms/step - accuracy: 0.7332 - loss: 0.6268 - val_accuracy: 0.7489 - val_loss
s: 0.6113
Epoch 587/600
148/148 2s 10ms/step - accuracy: 0.7347 - loss: 0.6234 - val_accuracy: 0.7449 - val_loss
s: 0.6189
Epoch 588/600
148/148 2s 10ms/step - accuracy: 0.7322 - loss: 0.6285 - val_accuracy: 0.7465 - val_loss
s: 0.6159
Epoch 589/600
148/148 2s 10ms/step - accuracy: 0.7312 - loss: 0.6296 - val_accuracy: 0.7482 - val_loss
s: 0.6103
Epoch 590/600
148/148 2s 10ms/step - accuracy: 0.7336 - loss: 0.6258 - val_accuracy: 0.7490 - val_loss
s: 0.6134
Epoch 591/600
148/148 2s 11ms/step - accuracy: 0.7298 - loss: 0.6287 - val_accuracy: 0.7484 - val_loss
s: 0.6137
Epoch 592/600
148/148 2s 10ms/step - accuracy: 0.7314 - loss: 0.6271 - val_accuracy: 0.7456 - val_loss
s: 0.6168
Epoch 593/600
148/148 2s 10ms/step - accuracy: 0.7351 - loss: 0.6284 - val_accuracy: 0.7459 - val_loss
s: 0.6178
Epoch 594/600
148/148 2s 10ms/step - accuracy: 0.7345 - loss: 0.6246 - val_accuracy: 0.7502 - val_loss
s: 0.6127
Epoch 595/600
148/148 2s 11ms/step - accuracy: 0.7329 - loss: 0.6290 - val_accuracy: 0.7446 - val_loss
s: 0.6158
Epoch 596/600
148/148 2s 11ms/step - accuracy: 0.7378 - loss: 0.6219 - val_accuracy: 0.7500 - val_loss
s: 0.6164
Epoch 597/600
148/148 2s 11ms/step - accuracy: 0.7298 - loss: 0.6321 - val_accuracy: 0.7458 - val_loss
s: 0.6187
Epoch 598/600
148/148 2s 10ms/step - accuracy: 0.7323 - loss: 0.6322 - val_accuracy: 0.7482 - val_loss
s: 0.6166
Epoch 599/600
148/148 2s 12ms/step - accuracy: 0.7335 - loss: 0.6277 - val_accuracy: 0.7452 - val_loss
s: 0.6200
Epoch 600/600
148/148 2s 12ms/step - accuracy: 0.7383 - loss: 0.6201 - val_accuracy: 0.7243 - val_loss
s: 0.6508
```

```
In [34]: ann6.evaluate(X_train, y_train)
```

```
2363/2363 3s 1ms/step - accuracy: 0.7952 - loss: 0.5090
```

```
Out[34]: [0.5079138875007629, 0.7955296635627747]
```

```
In [35]: ann6.summary()
```

```
Model: "sequential_2"
```

Layer (type)	Output Shape	Param #
dense_14 (Dense)	(None, 512)	23,040
batch_normalization_12 (BatchNormalization)	(None, 512)	2,048
dropout_12 (Dropout)	(None, 512)	0
dense_15 (Dense)	(None, 256)	131,328
batch_normalization_13 (BatchNormalization)	(None, 256)	1,024
dropout_13 (Dropout)	(None, 256)	0
dense_16 (Dense)	(None, 256)	65,792
batch_normalization_14 (BatchNormalization)	(None, 256)	1,024
dropout_14 (Dropout)	(None, 256)	0
dense_17 (Dense)	(None, 128)	32,896
batch_normalization_15 (BatchNormalization)	(None, 128)	512
dropout_15 (Dropout)	(None, 128)	0
dense_18 (Dense)	(None, 128)	16,512
batch_normalization_16 (BatchNormalization)	(None, 128)	512
dropout_16 (Dropout)	(None, 128)	0
dense_19 (Dense)	(None, 64)	8,256
batch_normalization_17 (BatchNormalization)	(None, 64)	256
dropout_17 (Dropout)	(None, 64)	0
dense_20 (Dense)	(None, 3)	195

Total params: 844,811 (3.22 MB)

Trainable params: 280,707 (1.07 MB)

Non-trainable params: 2,688 (10.50 KB)

Optimizer params: 561,416 (2.14 MB)

In [36]: eval_metric(ann6, X_train, y_train, X_val, y_val)

2363/2363 ————— 3s 1ms/step

591/591 ————— 1s 1ms/step

Test Set:

```
[[4261 1003 244]
 [1381 7504 1168]
 [ 74  832 2436]]
```

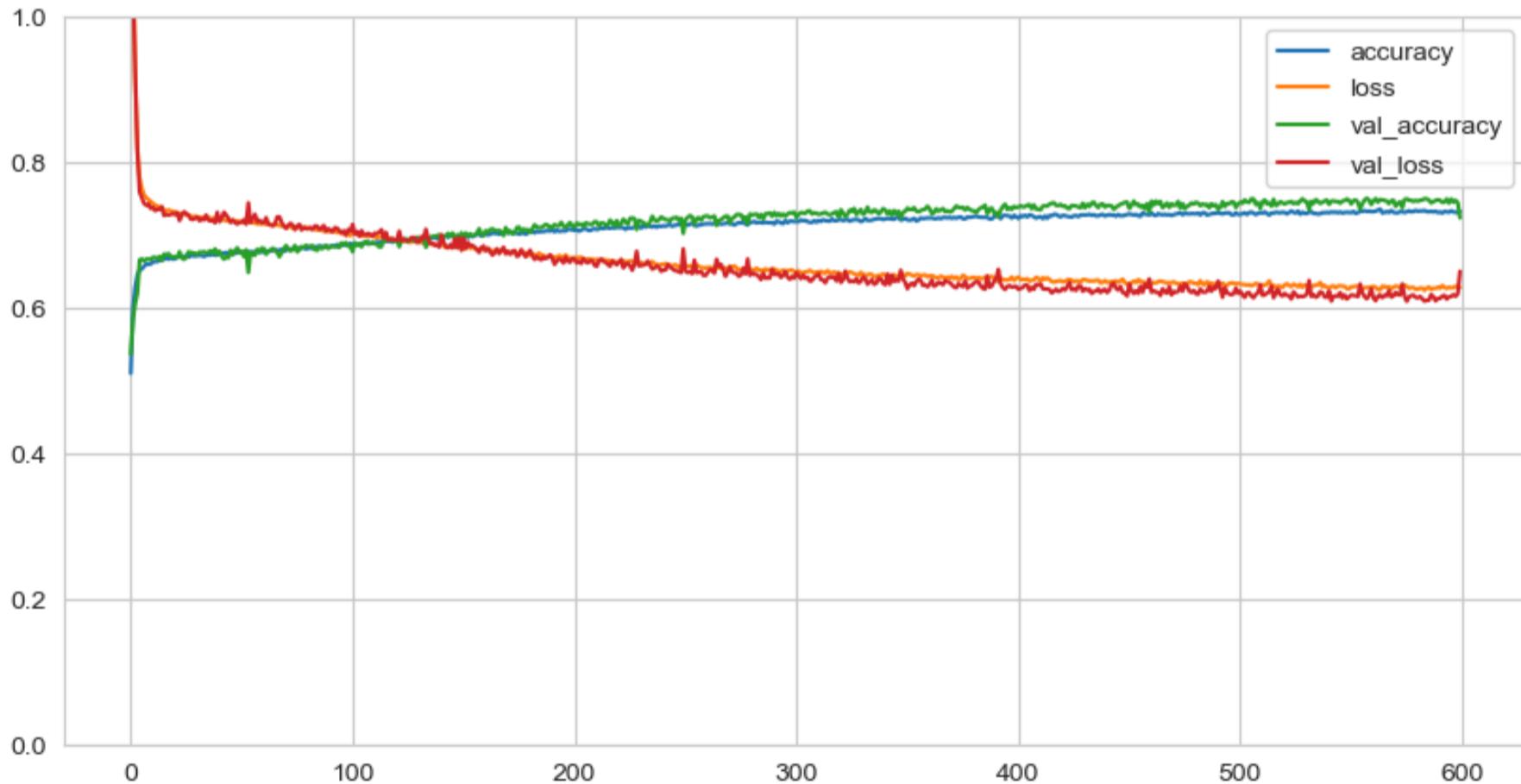
	precision	recall	f1-score	support
0	0.75	0.77	0.76	5508
1	0.80	0.75	0.77	10053
2	0.63	0.73	0.68	3342
accuracy			0.75	18903
macro avg	0.73	0.75	0.74	18903
weighted avg	0.76	0.75	0.75	18903

Train Set:

```
[[18066 3284 681]
 [ 4567 31611 4031]
 [ 121 2776 10473]]
```

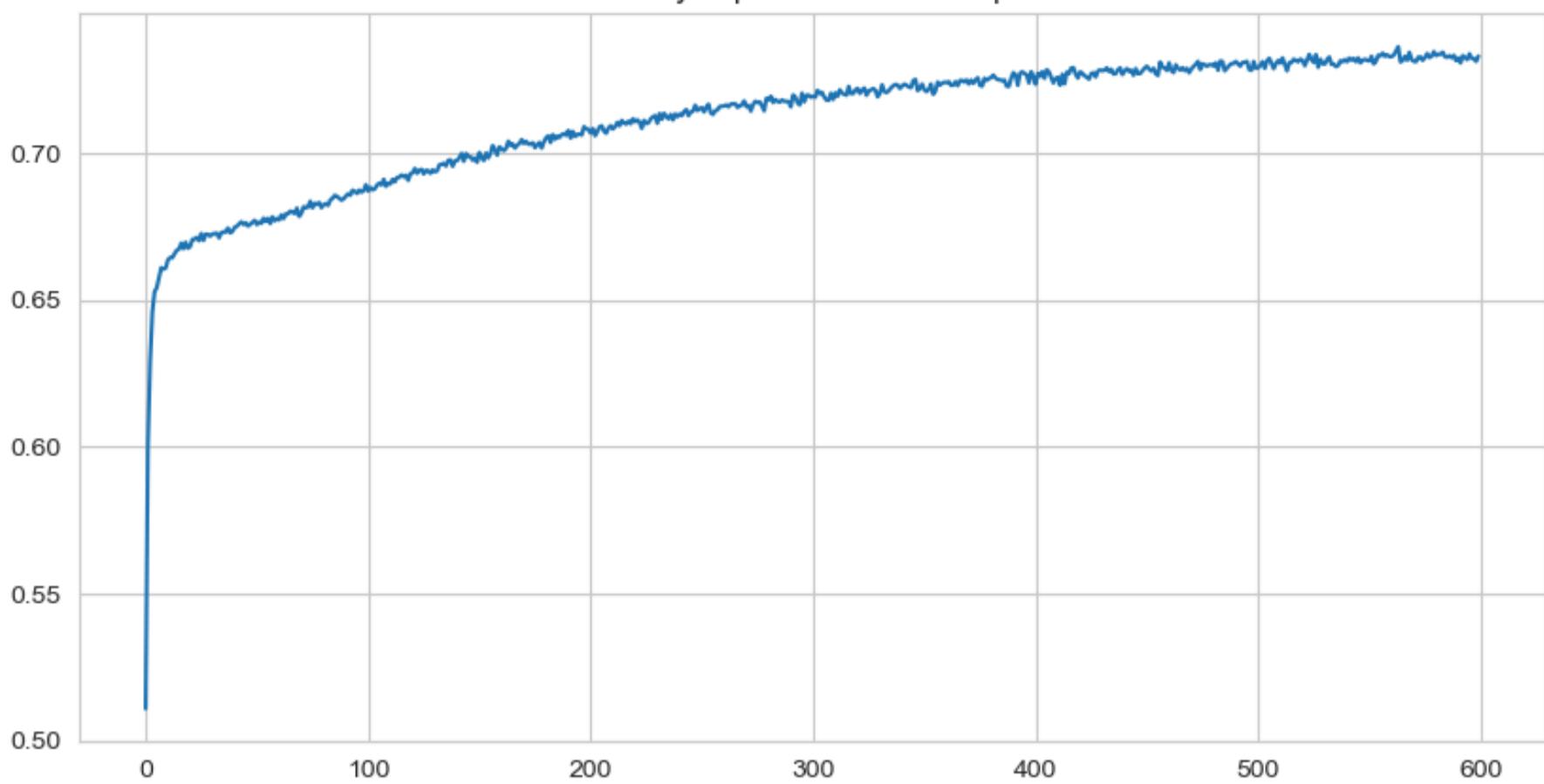
	precision	recall	f1-score	support
0	0.79	0.82	0.81	22031
1	0.84	0.79	0.81	40209
2	0.69	0.78	0.73	13370
accuracy			0.80	75610
macro avg	0.77	0.80	0.78	75610
weighted avg	0.80	0.80	0.80	75610

```
In [37]: pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



```
In [38]: pd.DataFrame(history.history)[“accuracy”].plot(figsize=(10, 5))
plt.title(“Accuracy improvements with Epoch”)
plt.show()
```

Accuracy improvements with Epoch



In [230...]

```
# Save the Model

from tensorflow.keras.models import load_model

# Save the model to 'model.h5' file
ann6.save('ann6_model.h5')

# To Load the model later for further use
loaded_ann6 = load_model('ann6_model.h5')
```

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)` . This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')` .
 WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.

ANN-6 Model with SMOTE Summary:

- **(Dense) layers: 7 / Neurons:** 512-256-256-128-128-64 / **Dropout:** 30-30-25-25-20-20% / **Learning Rate:** 0.001 / **Batch Size:** 512 / **Epochs:** 600 / **Early Stop (val_accuracy):** 60
- **Accuracy:** 0.80 / **Val_Accuracy:** 0.75 / **Loss:** 0.5090 / **Val_Loss:** 0.5079 / **Train Recall (Class 2):** 0.78 / **Test Recall (Class 2):** 0.73

Improvements: The model exhibits strong performance on the training set with an overall accuracy of 0.80 and a recall of 0.78 for Class 2. The inclusion of kernel regularizer (12) might have contributed to the model's robustness by reducing overfitting.

- Test recall for Class 2 has improved, indicating better sensitivity towards the minority class.

No Improvement: Validation accuracy is significantly lower than training accuracy, suggesting that the model might still be overfitting despite the regularization and dropout layers. This highlights a potential area for model adjustment to improve generalization.

Got Worse: Validation loss is similar to the training loss, suggesting that while the model is not heavily overfitting, the lower validation accuracy compared to the training accuracy indicates potential issues with the model's ability to generalize to new data. This might not represent severe overfitting but could point towards the need for slight adjustments in the model's complexity or its regularization approach.

ANN-6 Model with SMOTE (%83)

```
In [27]: print('Credi_Score Classes before Smote: ', y_train.value_counts())
print('\nCredi_Score Classes after Smote: ', y_train_smote.value_counts())
```

```
Credi_Score Classes before Smote: Credit_Score
1    40209
0    22031
2   13370
Name: count, dtype: int64
```

```
Credi_Score Classes after Smote: Credit_Score
1    40209
2    40209
0    40209
Name: count, dtype: int64
```

```
In [43]: # ANN-6 Model with SMOTE
```

```
# Fully connected (Dense) --> 7 Layers
# kernel_regularizer=l2(0.01) added

# 1) Model Architecture:
ann6_smote = Sequential([
    Dense(512, input_dim=X_train_smote.shape[1], activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    Dropout(0.3),
    Dense(256, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(256, activation='relu'),
    BatchNormalization(),
    Dropout(0.25),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.25),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.2),
    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.2),
    Dense(3, activation='softmax')
])

# 2) Compiling the Model:
ann6_smote.compile(optimizer=Adam(learning_rate=0.001),
                    loss='sparse_categorical_crossentropy',
                    metrics=['accuracy'])

# 3) Early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                                patience=60,
                                mode="auto",
                                restore_best_weights=True)

# 4) Train the model
history = ann6_smote.fit(
    x=X_train_smote,
    y=y_train_smote,
    validation_data=(X_val, y_val),
    batch_size=512,
    epochs=600,
    verbose=1,
    callbacks=[early_stopping])
```

Epoch 1/600
236/236 9s 13ms/step - accuracy: 0.5470 - loss: 1.6748 - val_accuracy: 0.5579 - val_loss: 1.1238
Epoch 2/600
236/236 3s 11ms/step - accuracy: 0.6791 - loss: 0.9748 - val_accuracy: 0.6085 - val_loss: 0.9028
Epoch 3/600
236/236 3s 11ms/step - accuracy: 0.7003 - loss: 0.8042 - val_accuracy: 0.6394 - val_loss: 0.8718
Epoch 4/600
236/236 2s 10ms/step - accuracy: 0.7044 - loss: 0.7713 - val_accuracy: 0.6495 - val_loss: 0.8423
Epoch 5/600
236/236 2s 10ms/step - accuracy: 0.7098 - loss: 0.7549 - val_accuracy: 0.6384 - val_loss: 0.8485
Epoch 6/600
236/236 3s 11ms/step - accuracy: 0.7093 - loss: 0.7487 - val_accuracy: 0.6355 - val_loss: 0.8518
Epoch 7/600
236/236 3s 12ms/step - accuracy: 0.7137 - loss: 0.7391 - val_accuracy: 0.6482 - val_loss: 0.8108
Epoch 8/600
236/236 3s 11ms/step - accuracy: 0.7104 - loss: 0.7444 - val_accuracy: 0.6507 - val_loss: 0.8143
Epoch 9/600
236/236 2s 10ms/step - accuracy: 0.7133 - loss: 0.7368 - val_accuracy: 0.6437 - val_loss: 0.8129
Epoch 10/600
236/236 3s 13ms/step - accuracy: 0.7140 - loss: 0.7376 - val_accuracy: 0.6471 - val_loss: 0.8068
Epoch 11/600
236/236 3s 11ms/step - accuracy: 0.7163 - loss: 0.7324 - val_accuracy: 0.6313 - val_loss: 0.8425
Epoch 12/600
236/236 3s 11ms/step - accuracy: 0.7170 - loss: 0.7293 - val_accuracy: 0.6452 - val_loss: 0.8183
Epoch 13/600
236/236 3s 12ms/step - accuracy: 0.7159 - loss: 0.7290 - val_accuracy: 0.6552 - val_loss: 0.7925
Epoch 14/600
236/236 3s 11ms/step - accuracy: 0.7170 - loss: 0.7244 - val_accuracy: 0.6575 - val_loss: 0.7925
Epoch 15/600
236/236 3s 11ms/step - accuracy: 0.7190 - loss: 0.7216 - val_accuracy: 0.6488 - val_loss: 0.8152
Epoch 16/600
236/236 3s 11ms/step - accuracy: 0.7194 - loss: 0.7202 - val_accuracy: 0.6373 - val_loss: 0.8199
Epoch 17/600
236/236 2s 10ms/step - accuracy: 0.7190 - loss: 0.7196 - val_accuracy: 0.6583 - val_loss: 0.7948
Epoch 18/600
236/236 3s 12ms/step - accuracy: 0.7190 - loss: 0.7174 - val_accuracy: 0.6473 - val_loss: 0.8123
Epoch 19/600
236/236 3s 11ms/step - accuracy: 0.7198 - loss: 0.7163 - val_accuracy: 0.6570 - val_loss: 0.8221
Epoch 20/600
236/236 2s 10ms/step - accuracy: 0.7211 - loss: 0.7125 - val_accuracy: 0.6533 - val_loss: 0.7947
Epoch 21/600
236/236 2s 10ms/step - accuracy: 0.7227 - loss: 0.7098 - val_accuracy: 0.6568 - val_loss: 0.7936
Epoch 22/600
236/236 2s 10ms/step - accuracy: 0.7203 - loss: 0.7133 - val_accuracy: 0.6586 - val_loss: 0.7809
Epoch 23/600
236/236 2s 10ms/step - accuracy: 0.7250 - loss: 0.7051 - val_accuracy: 0.6622 - val_loss: 0.7839
Epoch 24/600
236/236 3s 11ms/step - accuracy: 0.7258 - loss: 0.7024 - val_accuracy: 0.6674 - val_loss: 0.7799
Epoch 25/600
236/236 3s 11ms/step - accuracy: 0.7251 - loss: 0.7047 - val_accuracy: 0.6615 - val_loss: 0.7808
Epoch 26/600

236/236 2s 10ms/step - accuracy: 0.7248 - loss: 0.7035 - val_accuracy: 0.6610 - val_loss: 0.7939
Epoch 27/600
236/236 3s 11ms/step - accuracy: 0.7254 - loss: 0.7004 - val_accuracy: 0.6571 - val_loss: 0.7840
Epoch 28/600
236/236 3s 11ms/step - accuracy: 0.7267 - loss: 0.6976 - val_accuracy: 0.6606 - val_loss: 0.7829
Epoch 29/600
236/236 3s 11ms/step - accuracy: 0.7254 - loss: 0.7006 - val_accuracy: 0.6656 - val_loss: 0.7832
Epoch 30/600
236/236 3s 12ms/step - accuracy: 0.7262 - loss: 0.6955 - val_accuracy: 0.6682 - val_loss: 0.7729
Epoch 31/600
236/236 3s 11ms/step - accuracy: 0.7257 - loss: 0.6989 - val_accuracy: 0.6649 - val_loss: 0.8053
Epoch 32/600
236/236 3s 11ms/step - accuracy: 0.7266 - loss: 0.6950 - val_accuracy: 0.6552 - val_loss: 0.8013
Epoch 33/600
236/236 3s 11ms/step - accuracy: 0.7281 - loss: 0.6930 - val_accuracy: 0.6512 - val_loss: 0.7980
Epoch 34/600
236/236 3s 11ms/step - accuracy: 0.7291 - loss: 0.6918 - val_accuracy: 0.6672 - val_loss: 0.7743
Epoch 35/600
236/236 3s 11ms/step - accuracy: 0.7322 - loss: 0.6871 - val_accuracy: 0.6707 - val_loss: 0.7539
Epoch 36/600
236/236 3s 11ms/step - accuracy: 0.7313 - loss: 0.6901 - val_accuracy: 0.6464 - val_loss: 0.8035
Epoch 37/600
236/236 3s 11ms/step - accuracy: 0.7303 - loss: 0.6866 - val_accuracy: 0.6621 - val_loss: 0.7944
Epoch 38/600
236/236 3s 11ms/step - accuracy: 0.7319 - loss: 0.6858 - val_accuracy: 0.6641 - val_loss: 0.7771
Epoch 39/600
236/236 3s 12ms/step - accuracy: 0.7331 - loss: 0.6837 - val_accuracy: 0.6720 - val_loss: 0.7753
Epoch 40/600
236/236 3s 11ms/step - accuracy: 0.7331 - loss: 0.6791 - val_accuracy: 0.6543 - val_loss: 0.7834
Epoch 41/600
236/236 2s 10ms/step - accuracy: 0.7299 - loss: 0.6855 - val_accuracy: 0.6683 - val_loss: 0.7757
Epoch 42/600
236/236 3s 11ms/step - accuracy: 0.7341 - loss: 0.6794 - val_accuracy: 0.6646 - val_loss: 0.7782
Epoch 43/600
236/236 3s 11ms/step - accuracy: 0.7334 - loss: 0.6780 - val_accuracy: 0.6662 - val_loss: 0.7665
Epoch 44/600
236/236 3s 11ms/step - accuracy: 0.7341 - loss: 0.6783 - val_accuracy: 0.6747 - val_loss: 0.7647
Epoch 45/600
236/236 3s 12ms/step - accuracy: 0.7356 - loss: 0.6730 - val_accuracy: 0.6641 - val_loss: 0.7815
Epoch 46/600
236/236 3s 12ms/step - accuracy: 0.7361 - loss: 0.6741 - val_accuracy: 0.6601 - val_loss: 0.7854
Epoch 47/600
236/236 2s 10ms/step - accuracy: 0.7333 - loss: 0.6779 - val_accuracy: 0.6641 - val_loss: 0.7697
Epoch 48/600
236/236 2s 10ms/step - accuracy: 0.7386 - loss: 0.6714 - val_accuracy: 0.6730 - val_loss: 0.7745
Epoch 49/600
236/236 3s 11ms/step - accuracy: 0.7360 - loss: 0.6727 - val_accuracy: 0.6738 - val_loss: 0.7631
Epoch 50/600
236/236 2s 10ms/step - accuracy: 0.7370 - loss: 0.6704 - val_accuracy: 0.6739 - val_loss: 0.7638
Epoch 51/600
236/236 3s 11ms/step - accuracy: 0.7405 - loss: 0.6673 - val_accuracy: 0.6639 - val_loss:

s: 0.7743
Epoch 52/600
236/236 2s 10ms/step - accuracy: 0.7370 - loss: 0.6700 - val_accuracy: 0.6738 - val_loss: 0.7448
Epoch 53/600
236/236 2s 10ms/step - accuracy: 0.7352 - loss: 0.6724 - val_accuracy: 0.6705 - val_loss: 0.7718
Epoch 54/600
236/236 2s 10ms/step - accuracy: 0.7383 - loss: 0.6685 - val_accuracy: 0.6667 - val_loss: 0.7678
Epoch 55/600
236/236 3s 11ms/step - accuracy: 0.7395 - loss: 0.6658 - val_accuracy: 0.6652 - val_loss: 0.7712
Epoch 56/600
236/236 3s 11ms/step - accuracy: 0.7372 - loss: 0.6686 - val_accuracy: 0.6788 - val_loss: 0.7533
Epoch 57/600
236/236 3s 11ms/step - accuracy: 0.7391 - loss: 0.6656 - val_accuracy: 0.6619 - val_loss: 0.7764
Epoch 58/600
236/236 2s 10ms/step - accuracy: 0.7420 - loss: 0.6585 - val_accuracy: 0.6726 - val_loss: 0.7651
Epoch 59/600
236/236 2s 10ms/step - accuracy: 0.7406 - loss: 0.6628 - val_accuracy: 0.6856 - val_loss: 0.7528
Epoch 60/600
236/236 3s 11ms/step - accuracy: 0.7410 - loss: 0.6614 - val_accuracy: 0.6838 - val_loss: 0.7418
Epoch 61/600
236/236 3s 11ms/step - accuracy: 0.7413 - loss: 0.6607 - val_accuracy: 0.6697 - val_loss: 0.7682
Epoch 62/600
236/236 3s 11ms/step - accuracy: 0.7408 - loss: 0.6602 - val_accuracy: 0.6639 - val_loss: 0.7835
Epoch 63/600
236/236 3s 11ms/step - accuracy: 0.7450 - loss: 0.6550 - val_accuracy: 0.6839 - val_loss: 0.7555
Epoch 64/600
236/236 2s 10ms/step - accuracy: 0.7456 - loss: 0.6557 - val_accuracy: 0.6804 - val_loss: 0.7507
Epoch 65/600
236/236 2s 10ms/step - accuracy: 0.7442 - loss: 0.6570 - val_accuracy: 0.6730 - val_loss: 0.7594
Epoch 66/600
236/236 3s 11ms/step - accuracy: 0.7409 - loss: 0.6582 - val_accuracy: 0.6795 - val_loss: 0.7617
Epoch 67/600
236/236 5s 11ms/step - accuracy: 0.7455 - loss: 0.6525 - val_accuracy: 0.6726 - val_loss: 0.7684
Epoch 68/600
236/236 3s 11ms/step - accuracy: 0.7442 - loss: 0.6560 - val_accuracy: 0.6701 - val_loss: 0.7655
Epoch 69/600
236/236 2s 10ms/step - accuracy: 0.7389 - loss: 0.6609 - val_accuracy: 0.6804 - val_loss: 0.7556
Epoch 70/600
236/236 2s 10ms/step - accuracy: 0.7458 - loss: 0.6525 - val_accuracy: 0.6785 - val_loss: 0.7644
Epoch 71/600
236/236 2s 10ms/step - accuracy: 0.7455 - loss: 0.6552 - val_accuracy: 0.6682 - val_loss: 0.7671
Epoch 72/600
236/236 3s 11ms/step - accuracy: 0.7463 - loss: 0.6527 - val_accuracy: 0.6802 - val_loss: 0.7486
Epoch 73/600
236/236 3s 11ms/step - accuracy: 0.7463 - loss: 0.6493 - val_accuracy: 0.6836 - val_loss: 0.7418
Epoch 74/600
236/236 2s 10ms/step - accuracy: 0.7471 - loss: 0.6525 - val_accuracy: 0.6823 - val_loss: 0.7627
Epoch 75/600
236/236 2s 10ms/step - accuracy: 0.7458 - loss: 0.6513 - val_accuracy: 0.6844 - val_loss: 0.7571
Epoch 76/600
236/236 2s 10ms/step - accuracy: 0.7496 - loss: 0.6472 - val_accuracy: 0.6829 - val_loss: 0.7379

Epoch 77/600
236/236 2s 10ms/step - accuracy: 0.7480 - loss: 0.6473 - val_accuracy: 0.6747 - val_loss: 0.7590
Epoch 78/600
236/236 2s 10ms/step - accuracy: 0.7485 - loss: 0.6437 - val_accuracy: 0.6701 - val_loss: 0.7634
Epoch 79/600
236/236 2s 10ms/step - accuracy: 0.7448 - loss: 0.6487 - val_accuracy: 0.6731 - val_loss: 0.7526
Epoch 80/600
236/236 3s 11ms/step - accuracy: 0.7482 - loss: 0.6479 - val_accuracy: 0.6811 - val_loss: 0.7389
Epoch 81/600
236/236 3s 11ms/step - accuracy: 0.7484 - loss: 0.6447 - val_accuracy: 0.6769 - val_loss: 0.7594
Epoch 82/600
236/236 2s 10ms/step - accuracy: 0.7526 - loss: 0.6419 - val_accuracy: 0.6863 - val_loss: 0.7416
Epoch 83/600
236/236 2s 10ms/step - accuracy: 0.7507 - loss: 0.6417 - val_accuracy: 0.6872 - val_loss: 0.7396
Epoch 84/600
236/236 2s 10ms/step - accuracy: 0.7472 - loss: 0.6459 - val_accuracy: 0.6788 - val_loss: 0.7512
Epoch 85/600
236/236 2s 10ms/step - accuracy: 0.7465 - loss: 0.6437 - val_accuracy: 0.6744 - val_loss: 0.7564
Epoch 86/600
236/236 2s 10ms/step - accuracy: 0.7492 - loss: 0.6421 - val_accuracy: 0.6777 - val_loss: 0.7550
Epoch 87/600
236/236 3s 11ms/step - accuracy: 0.7483 - loss: 0.6449 - val_accuracy: 0.6733 - val_loss: 0.7633
Epoch 88/600
236/236 3s 11ms/step - accuracy: 0.7482 - loss: 0.6426 - val_accuracy: 0.6849 - val_loss: 0.7496
Epoch 89/600
236/236 3s 11ms/step - accuracy: 0.7509 - loss: 0.6403 - val_accuracy: 0.6793 - val_loss: 0.7542
Epoch 90/600
236/236 3s 11ms/step - accuracy: 0.7526 - loss: 0.6407 - val_accuracy: 0.6690 - val_loss: 0.7685
Epoch 91/600
236/236 3s 11ms/step - accuracy: 0.7530 - loss: 0.6343 - val_accuracy: 0.6803 - val_loss: 0.7472
Epoch 92/600
236/236 3s 11ms/step - accuracy: 0.7518 - loss: 0.6392 - val_accuracy: 0.6874 - val_loss: 0.7397
Epoch 93/600
236/236 3s 11ms/step - accuracy: 0.7488 - loss: 0.6432 - val_accuracy: 0.6842 - val_loss: 0.7439
Epoch 94/600
236/236 2s 10ms/step - accuracy: 0.7525 - loss: 0.6356 - val_accuracy: 0.6859 - val_loss: 0.7395
Epoch 95/600
236/236 2s 10ms/step - accuracy: 0.7520 - loss: 0.6377 - val_accuracy: 0.6809 - val_loss: 0.7449
Epoch 96/600
236/236 3s 11ms/step - accuracy: 0.7526 - loss: 0.6364 - val_accuracy: 0.6818 - val_loss: 0.7429
Epoch 97/600
236/236 3s 11ms/step - accuracy: 0.7543 - loss: 0.6330 - val_accuracy: 0.6849 - val_loss: 0.7441
Epoch 98/600
236/236 2s 10ms/step - accuracy: 0.7505 - loss: 0.6385 - val_accuracy: 0.6886 - val_loss: 0.7426
Epoch 99/600
236/236 3s 11ms/step - accuracy: 0.7536 - loss: 0.6347 - val_accuracy: 0.6825 - val_loss: 0.7436
Epoch 100/600
236/236 2s 10ms/step - accuracy: 0.7527 - loss: 0.6357 - val_accuracy: 0.6853 - val_loss: 0.7379
Epoch 101/600
236/236 2s 10ms/step - accuracy: 0.7559 - loss: 0.6314 - val_accuracy: 0.6797 - val_loss: 0.7465
Epoch 102/600

236/236 2s 10ms/step - accuracy: 0.7557 - loss: 0.6283 - val_accuracy: 0.6896 - val_loss: 0.7381
Epoch 103/600
236/236 2s 10ms/step - accuracy: 0.7542 - loss: 0.6326 - val_accuracy: 0.6881 - val_loss: 0.7304
Epoch 104/600
236/236 2s 10ms/step - accuracy: 0.7565 - loss: 0.6303 - val_accuracy: 0.6931 - val_loss: 0.7306
Epoch 105/600
236/236 2s 10ms/step - accuracy: 0.7552 - loss: 0.6314 - val_accuracy: 0.6878 - val_loss: 0.7317
Epoch 106/600
236/236 3s 11ms/step - accuracy: 0.7509 - loss: 0.6360 - val_accuracy: 0.6862 - val_loss: 0.7389
Epoch 107/600
236/236 2s 10ms/step - accuracy: 0.7530 - loss: 0.6340 - val_accuracy: 0.6916 - val_loss: 0.7327
Epoch 108/600
236/236 2s 10ms/step - accuracy: 0.7535 - loss: 0.6332 - val_accuracy: 0.6832 - val_loss: 0.7370
Epoch 109/600
236/236 2s 10ms/step - accuracy: 0.7536 - loss: 0.6324 - val_accuracy: 0.6897 - val_loss: 0.7327
Epoch 110/600
236/236 2s 10ms/step - accuracy: 0.7567 - loss: 0.6310 - val_accuracy: 0.6830 - val_loss: 0.7379
Epoch 111/600
236/236 2s 10ms/step - accuracy: 0.7556 - loss: 0.6285 - val_accuracy: 0.6899 - val_loss: 0.7394
Epoch 112/600
236/236 2s 10ms/step - accuracy: 0.7516 - loss: 0.6360 - val_accuracy: 0.6863 - val_loss: 0.7362
Epoch 113/600
236/236 3s 11ms/step - accuracy: 0.7548 - loss: 0.6299 - val_accuracy: 0.6924 - val_loss: 0.7330
Epoch 114/600
236/236 2s 10ms/step - accuracy: 0.7560 - loss: 0.6302 - val_accuracy: 0.6917 - val_loss: 0.7307
Epoch 115/600
236/236 2s 10ms/step - accuracy: 0.7565 - loss: 0.6266 - val_accuracy: 0.6795 - val_loss: 0.7592
Epoch 116/600
236/236 2s 10ms/step - accuracy: 0.7571 - loss: 0.6268 - val_accuracy: 0.6834 - val_loss: 0.7327
Epoch 117/600
236/236 2s 10ms/step - accuracy: 0.7554 - loss: 0.6293 - val_accuracy: 0.6908 - val_loss: 0.7381
Epoch 118/600
236/236 2s 10ms/step - accuracy: 0.7557 - loss: 0.6298 - val_accuracy: 0.6869 - val_loss: 0.7382
Epoch 119/600
236/236 3s 11ms/step - accuracy: 0.7594 - loss: 0.6244 - val_accuracy: 0.6898 - val_loss: 0.7352
Epoch 120/600
236/236 3s 12ms/step - accuracy: 0.7562 - loss: 0.6269 - val_accuracy: 0.6918 - val_loss: 0.7252
Epoch 121/600
236/236 3s 11ms/step - accuracy: 0.7577 - loss: 0.6240 - val_accuracy: 0.6889 - val_loss: 0.7333
Epoch 122/600
236/236 2s 10ms/step - accuracy: 0.7576 - loss: 0.6268 - val_accuracy: 0.6939 - val_loss: 0.7310
Epoch 123/600
236/236 2s 10ms/step - accuracy: 0.7593 - loss: 0.6225 - val_accuracy: 0.6956 - val_loss: 0.7246
Epoch 124/600
236/236 3s 11ms/step - accuracy: 0.7590 - loss: 0.6225 - val_accuracy: 0.6805 - val_loss: 0.7462
Epoch 125/600
236/236 3s 11ms/step - accuracy: 0.7568 - loss: 0.6263 - val_accuracy: 0.6950 - val_loss: 0.7239
Epoch 126/600
236/236 2s 10ms/step - accuracy: 0.7593 - loss: 0.6215 - val_accuracy: 0.6896 - val_loss: 0.7378
Epoch 127/600
236/236 2s 10ms/step - accuracy: 0.7565 - loss: 0.6279 - val_accuracy: 0.6933 - val_loss:

s: 0.7202
Epoch 128/600
236/236 2s 10ms/step - accuracy: 0.7573 - loss: 0.6236 - val_accuracy: 0.6930 - val_loss: 0.7367
Epoch 129/600
236/236 2s 10ms/step - accuracy: 0.7588 - loss: 0.6220 - val_accuracy: 0.6806 - val_loss: 0.7445
Epoch 130/600
236/236 3s 10ms/step - accuracy: 0.7604 - loss: 0.6231 - val_accuracy: 0.6868 - val_loss: 0.7327
Epoch 131/600
236/236 2s 10ms/step - accuracy: 0.7591 - loss: 0.6233 - val_accuracy: 0.7017 - val_loss: 0.7138
Epoch 132/600
236/236 2s 10ms/step - accuracy: 0.7579 - loss: 0.6262 - val_accuracy: 0.6904 - val_loss: 0.7284
Epoch 133/600
236/236 2s 10ms/step - accuracy: 0.7582 - loss: 0.6220 - val_accuracy: 0.7003 - val_loss: 0.7092
Epoch 134/600
236/236 2s 10ms/step - accuracy: 0.7575 - loss: 0.6249 - val_accuracy: 0.6909 - val_loss: 0.7353
Epoch 135/600
236/236 2s 10ms/step - accuracy: 0.7598 - loss: 0.6198 - val_accuracy: 0.6878 - val_loss: 0.7325
Epoch 136/600
236/236 2s 10ms/step - accuracy: 0.7582 - loss: 0.6222 - val_accuracy: 0.6944 - val_loss: 0.7232
Epoch 137/600
236/236 2s 10ms/step - accuracy: 0.7597 - loss: 0.6225 - val_accuracy: 0.6965 - val_loss: 0.7159
Epoch 138/600
236/236 3s 11ms/step - accuracy: 0.7602 - loss: 0.6200 - val_accuracy: 0.6921 - val_loss: 0.7358
Epoch 139/600
236/236 3s 11ms/step - accuracy: 0.7596 - loss: 0.6183 - val_accuracy: 0.6905 - val_loss: 0.7218
Epoch 140/600
236/236 2s 10ms/step - accuracy: 0.7594 - loss: 0.6235 - val_accuracy: 0.6935 - val_loss: 0.7247
Epoch 141/600
236/236 3s 11ms/step - accuracy: 0.7601 - loss: 0.6205 - val_accuracy: 0.6995 - val_loss: 0.7159
Epoch 142/600
236/236 2s 10ms/step - accuracy: 0.7612 - loss: 0.6195 - val_accuracy: 0.6943 - val_loss: 0.7222
Epoch 143/600
236/236 2s 10ms/step - accuracy: 0.7605 - loss: 0.6191 - val_accuracy: 0.6801 - val_loss: 0.7626
Epoch 144/600
236/236 2s 10ms/step - accuracy: 0.7610 - loss: 0.6191 - val_accuracy: 0.6969 - val_loss: 0.7159
Epoch 145/600
236/236 3s 12ms/step - accuracy: 0.7622 - loss: 0.6170 - val_accuracy: 0.6999 - val_loss: 0.7192
Epoch 146/600
236/236 3s 12ms/step - accuracy: 0.7597 - loss: 0.6203 - val_accuracy: 0.6903 - val_loss: 0.7492
Epoch 147/600
236/236 3s 11ms/step - accuracy: 0.7588 - loss: 0.6236 - val_accuracy: 0.6917 - val_loss: 0.7342
Epoch 148/600
236/236 2s 10ms/step - accuracy: 0.7618 - loss: 0.6180 - val_accuracy: 0.7003 - val_loss: 0.7205
Epoch 149/600
236/236 3s 11ms/step - accuracy: 0.7651 - loss: 0.6129 - val_accuracy: 0.6890 - val_loss: 0.7371
Epoch 150/600
236/236 2s 10ms/step - accuracy: 0.7623 - loss: 0.6177 - val_accuracy: 0.6953 - val_loss: 0.7302
Epoch 151/600
236/236 3s 11ms/step - accuracy: 0.7593 - loss: 0.6218 - val_accuracy: 0.6915 - val_loss: 0.7355
Epoch 152/600
236/236 2s 10ms/step - accuracy: 0.7611 - loss: 0.6203 - val_accuracy: 0.6963 - val_loss: 0.7202

Epoch 153/600
236/236 2s 10ms/step - accuracy: 0.7609 - loss: 0.6183 - val_accuracy: 0.6957 - val_loss: 0.7222
Epoch 154/600
236/236 3s 11ms/step - accuracy: 0.7627 - loss: 0.6135 - val_accuracy: 0.6951 - val_loss: 0.7212
Epoch 155/600
236/236 3s 12ms/step - accuracy: 0.7631 - loss: 0.6152 - val_accuracy: 0.6981 - val_loss: 0.7094
Epoch 156/600
236/236 2s 10ms/step - accuracy: 0.7608 - loss: 0.6190 - val_accuracy: 0.6927 - val_loss: 0.7259
Epoch 157/600
236/236 3s 11ms/step - accuracy: 0.7601 - loss: 0.6164 - val_accuracy: 0.6988 - val_loss: 0.7132
Epoch 158/600
236/236 3s 11ms/step - accuracy: 0.7608 - loss: 0.6181 - val_accuracy: 0.6971 - val_loss: 0.7224
Epoch 159/600
236/236 2s 10ms/step - accuracy: 0.7628 - loss: 0.6164 - val_accuracy: 0.6863 - val_loss: 0.7453
Epoch 160/600
236/236 2s 10ms/step - accuracy: 0.7640 - loss: 0.6151 - val_accuracy: 0.6953 - val_loss: 0.7269
Epoch 161/600
236/236 2s 10ms/step - accuracy: 0.7613 - loss: 0.6177 - val_accuracy: 0.6897 - val_loss: 0.7320
Epoch 162/600
236/236 2s 10ms/step - accuracy: 0.7612 - loss: 0.6167 - val_accuracy: 0.6993 - val_loss: 0.7113
Epoch 163/600
236/236 2s 10ms/step - accuracy: 0.7639 - loss: 0.6119 - val_accuracy: 0.7023 - val_loss: 0.7051
Epoch 164/600
236/236 2s 10ms/step - accuracy: 0.7634 - loss: 0.6124 - val_accuracy: 0.6929 - val_loss: 0.7292
Epoch 165/600
236/236 2s 10ms/step - accuracy: 0.7624 - loss: 0.6158 - val_accuracy: 0.6986 - val_loss: 0.7112
Epoch 166/600
236/236 3s 11ms/step - accuracy: 0.7648 - loss: 0.6087 - val_accuracy: 0.6943 - val_loss: 0.7255
Epoch 167/600
236/236 2s 10ms/step - accuracy: 0.7632 - loss: 0.6137 - val_accuracy: 0.6980 - val_loss: 0.7158
Epoch 168/600
236/236 2s 10ms/step - accuracy: 0.7621 - loss: 0.6136 - val_accuracy: 0.7010 - val_loss: 0.7145
Epoch 169/600
236/236 3s 11ms/step - accuracy: 0.7618 - loss: 0.6129 - val_accuracy: 0.6815 - val_loss: 0.7378
Epoch 170/600
236/236 3s 12ms/step - accuracy: 0.7633 - loss: 0.6156 - val_accuracy: 0.6976 - val_loss: 0.7224
Epoch 171/600
236/236 2s 10ms/step - accuracy: 0.7659 - loss: 0.6072 - val_accuracy: 0.6968 - val_loss: 0.7160
Epoch 172/600
236/236 2s 10ms/step - accuracy: 0.7643 - loss: 0.6124 - val_accuracy: 0.6913 - val_loss: 0.7240
Epoch 173/600
236/236 2s 10ms/step - accuracy: 0.7640 - loss: 0.6132 - val_accuracy: 0.6899 - val_loss: 0.7323
Epoch 174/600
236/236 2s 10ms/step - accuracy: 0.7659 - loss: 0.6093 - val_accuracy: 0.6948 - val_loss: 0.7226
Epoch 175/600
236/236 2s 10ms/step - accuracy: 0.7653 - loss: 0.6114 - val_accuracy: 0.6980 - val_loss: 0.7144
Epoch 176/600
236/236 2s 10ms/step - accuracy: 0.7672 - loss: 0.6094 - val_accuracy: 0.7043 - val_loss: 0.7035
Epoch 177/600
236/236 3s 11ms/step - accuracy: 0.7639 - loss: 0.6118 - val_accuracy: 0.6949 - val_loss: 0.7214
Epoch 178/600

236/236 3s 11ms/step - accuracy: 0.7648 - loss: 0.6104 - val_accuracy: 0.6962 - val_loss: 0.7206
Epoch 179/600
236/236 3s 11ms/step - accuracy: 0.7635 - loss: 0.6138 - val_accuracy: 0.6956 - val_loss: 0.7204
Epoch 180/600
236/236 2s 10ms/step - accuracy: 0.7646 - loss: 0.6081 - val_accuracy: 0.7034 - val_loss: 0.7073
Epoch 181/600
236/236 3s 10ms/step - accuracy: 0.7616 - loss: 0.6144 - val_accuracy: 0.6913 - val_loss: 0.7248
Epoch 182/600
236/236 2s 10ms/step - accuracy: 0.7659 - loss: 0.6095 - val_accuracy: 0.6936 - val_loss: 0.7211
Epoch 183/600
236/236 3s 11ms/step - accuracy: 0.7655 - loss: 0.6080 - val_accuracy: 0.6990 - val_loss: 0.7067
Epoch 184/600
236/236 3s 11ms/step - accuracy: 0.7673 - loss: 0.6075 - val_accuracy: 0.6932 - val_loss: 0.7251
Epoch 185/600
236/236 2s 10ms/step - accuracy: 0.7660 - loss: 0.6075 - val_accuracy: 0.7011 - val_loss: 0.7135
Epoch 186/600
236/236 2s 10ms/step - accuracy: 0.7650 - loss: 0.6115 - val_accuracy: 0.6910 - val_loss: 0.7273
Epoch 187/600
236/236 2s 10ms/step - accuracy: 0.7659 - loss: 0.6090 - val_accuracy: 0.7013 - val_loss: 0.7197
Epoch 188/600
236/236 3s 11ms/step - accuracy: 0.7639 - loss: 0.6103 - val_accuracy: 0.6966 - val_loss: 0.7183
Epoch 189/600
236/236 2s 10ms/step - accuracy: 0.7663 - loss: 0.6083 - val_accuracy: 0.6976 - val_loss: 0.7120
Epoch 190/600
236/236 3s 11ms/step - accuracy: 0.7675 - loss: 0.6048 - val_accuracy: 0.6960 - val_loss: 0.7266
Epoch 191/600
236/236 3s 11ms/step - accuracy: 0.7636 - loss: 0.6129 - val_accuracy: 0.7040 - val_loss: 0.7061
Epoch 192/600
236/236 3s 11ms/step - accuracy: 0.7642 - loss: 0.6116 - val_accuracy: 0.6992 - val_loss: 0.7120
Epoch 193/600
236/236 5s 10ms/step - accuracy: 0.7675 - loss: 0.6098 - val_accuracy: 0.7008 - val_loss: 0.7040
Epoch 194/600
236/236 3s 11ms/step - accuracy: 0.7652 - loss: 0.6087 - val_accuracy: 0.7028 - val_loss: 0.7094
Epoch 195/600
236/236 3s 11ms/step - accuracy: 0.7680 - loss: 0.6092 - val_accuracy: 0.7010 - val_loss: 0.7093
Epoch 196/600
236/236 3s 11ms/step - accuracy: 0.7655 - loss: 0.6062 - val_accuracy: 0.6976 - val_loss: 0.7178
Epoch 197/600
236/236 2s 10ms/step - accuracy: 0.7679 - loss: 0.6031 - val_accuracy: 0.6999 - val_loss: 0.7044
Epoch 198/600
236/236 2s 10ms/step - accuracy: 0.7660 - loss: 0.6095 - val_accuracy: 0.6920 - val_loss: 0.7290
Epoch 199/600
236/236 3s 11ms/step - accuracy: 0.7644 - loss: 0.6108 - val_accuracy: 0.7050 - val_loss: 0.7079
Epoch 200/600
236/236 2s 10ms/step - accuracy: 0.7646 - loss: 0.6105 - val_accuracy: 0.7016 - val_loss: 0.7119
Epoch 201/600
236/236 3s 11ms/step - accuracy: 0.7661 - loss: 0.6084 - val_accuracy: 0.7049 - val_loss: 0.7029
Epoch 202/600
236/236 3s 11ms/step - accuracy: 0.7657 - loss: 0.6092 - val_accuracy: 0.7046 - val_loss: 0.7124
Epoch 203/600
236/236 2s 10ms/step - accuracy: 0.7683 - loss: 0.6042 - val_accuracy: 0.6952 - val_loss:

s: 0.7254
Epoch 204/600
236/236 3s 11ms/step - accuracy: 0.7660 - loss: 0.6060 - val_accuracy: 0.6969 - val_loss: 0.7309
Epoch 205/600
236/236 2s 10ms/step - accuracy: 0.7644 - loss: 0.6093 - val_accuracy: 0.7024 - val_loss: 0.7153
Epoch 206/600
236/236 3s 11ms/step - accuracy: 0.7686 - loss: 0.6014 - val_accuracy: 0.6942 - val_loss: 0.7169
Epoch 207/600
236/236 2s 10ms/step - accuracy: 0.7645 - loss: 0.6114 - val_accuracy: 0.7021 - val_loss: 0.7067
Epoch 208/600
236/236 3s 11ms/step - accuracy: 0.7677 - loss: 0.6046 - val_accuracy: 0.7032 - val_loss: 0.7156
Epoch 209/600
236/236 3s 11ms/step - accuracy: 0.7702 - loss: 0.6008 - val_accuracy: 0.7005 - val_loss: 0.7114
Epoch 210/600
236/236 2s 10ms/step - accuracy: 0.7661 - loss: 0.6071 - val_accuracy: 0.6921 - val_loss: 0.7278
Epoch 211/600
236/236 2s 10ms/step - accuracy: 0.7683 - loss: 0.6077 - val_accuracy: 0.7023 - val_loss: 0.7198
Epoch 212/600
236/236 3s 11ms/step - accuracy: 0.7664 - loss: 0.6066 - val_accuracy: 0.6926 - val_loss: 0.7283
Epoch 213/600
236/236 3s 11ms/step - accuracy: 0.7663 - loss: 0.6084 - val_accuracy: 0.7004 - val_loss: 0.7201
Epoch 214/600
236/236 3s 12ms/step - accuracy: 0.7670 - loss: 0.6066 - val_accuracy: 0.7051 - val_loss: 0.7156
Epoch 215/600
236/236 3s 11ms/step - accuracy: 0.7673 - loss: 0.6039 - val_accuracy: 0.6955 - val_loss: 0.7183
Epoch 216/600
236/236 2s 10ms/step - accuracy: 0.7667 - loss: 0.6063 - val_accuracy: 0.6982 - val_loss: 0.7213
Epoch 217/600
236/236 2s 10ms/step - accuracy: 0.7669 - loss: 0.6074 - val_accuracy: 0.6858 - val_loss: 0.7425
Epoch 218/600
236/236 2s 10ms/step - accuracy: 0.7699 - loss: 0.6035 - val_accuracy: 0.7002 - val_loss: 0.7124
Epoch 219/600
236/236 2s 10ms/step - accuracy: 0.7689 - loss: 0.6003 - val_accuracy: 0.7045 - val_loss: 0.7139
Epoch 220/600
236/236 3s 11ms/step - accuracy: 0.7714 - loss: 0.5999 - val_accuracy: 0.7005 - val_loss: 0.7158
Epoch 221/600
236/236 3s 11ms/step - accuracy: 0.7655 - loss: 0.6084 - val_accuracy: 0.7029 - val_loss: 0.7178
Epoch 222/600
236/236 2s 10ms/step - accuracy: 0.7688 - loss: 0.6022 - val_accuracy: 0.7051 - val_loss: 0.7196
Epoch 223/600
236/236 2s 10ms/step - accuracy: 0.7684 - loss: 0.6046 - val_accuracy: 0.6998 - val_loss: 0.7131
Epoch 224/600
236/236 2s 10ms/step - accuracy: 0.7701 - loss: 0.6001 - val_accuracy: 0.7042 - val_loss: 0.7115
Epoch 225/600
236/236 2s 10ms/step - accuracy: 0.7671 - loss: 0.6056 - val_accuracy: 0.7047 - val_loss: 0.7146
Epoch 226/600
236/236 2s 10ms/step - accuracy: 0.7686 - loss: 0.6031 - val_accuracy: 0.7087 - val_loss: 0.7002
Epoch 227/600
236/236 3s 11ms/step - accuracy: 0.7661 - loss: 0.6067 - val_accuracy: 0.6982 - val_loss: 0.7248
Epoch 228/600
236/236 3s 11ms/step - accuracy: 0.7675 - loss: 0.6032 - val_accuracy: 0.7016 - val_loss: 0.7085

Epoch 229/600
236/236 3s 11ms/step - accuracy: 0.7693 - loss: 0.5996 - val_accuracy: 0.7042 - val_loss: 0.7086
Epoch 230/600
236/236 3s 11ms/step - accuracy: 0.7674 - loss: 0.6043 - val_accuracy: 0.7081 - val_loss: 0.7065
Epoch 231/600
236/236 3s 11ms/step - accuracy: 0.7700 - loss: 0.6010 - val_accuracy: 0.7099 - val_loss: 0.6981
Epoch 232/600
236/236 2s 10ms/step - accuracy: 0.7690 - loss: 0.6042 - val_accuracy: 0.6986 - val_loss: 0.7176
Epoch 233/600
236/236 3s 11ms/step - accuracy: 0.7677 - loss: 0.6048 - val_accuracy: 0.6972 - val_loss: 0.7158
Epoch 234/600
236/236 2s 10ms/step - accuracy: 0.7701 - loss: 0.5981 - val_accuracy: 0.7044 - val_loss: 0.7014
Epoch 235/600
236/236 2s 10ms/step - accuracy: 0.7692 - loss: 0.6022 - val_accuracy: 0.7067 - val_loss: 0.7044
Epoch 236/600
236/236 3s 11ms/step - accuracy: 0.7699 - loss: 0.5984 - val_accuracy: 0.7009 - val_loss: 0.7194
Epoch 237/600
236/236 3s 11ms/step - accuracy: 0.7704 - loss: 0.5965 - val_accuracy: 0.6926 - val_loss: 0.7315
Epoch 238/600
236/236 3s 11ms/step - accuracy: 0.7711 - loss: 0.5994 - val_accuracy: 0.7053 - val_loss: 0.6979
Epoch 239/600
236/236 3s 11ms/step - accuracy: 0.7696 - loss: 0.5991 - val_accuracy: 0.6983 - val_loss: 0.7165
Epoch 240/600
236/236 3s 11ms/step - accuracy: 0.7718 - loss: 0.5976 - val_accuracy: 0.6983 - val_loss: 0.7124
Epoch 241/600
236/236 3s 11ms/step - accuracy: 0.7694 - loss: 0.6032 - val_accuracy: 0.7018 - val_loss: 0.7126
Epoch 242/600
236/236 2s 10ms/step - accuracy: 0.7721 - loss: 0.6000 - val_accuracy: 0.6942 - val_loss: 0.7148
Epoch 243/600
236/236 3s 11ms/step - accuracy: 0.7693 - loss: 0.6011 - val_accuracy: 0.7051 - val_loss: 0.7095
Epoch 244/600
236/236 3s 11ms/step - accuracy: 0.7695 - loss: 0.6013 - val_accuracy: 0.7097 - val_loss: 0.6989
Epoch 245/600
236/236 2s 10ms/step - accuracy: 0.7698 - loss: 0.6002 - val_accuracy: 0.7022 - val_loss: 0.7177
Epoch 246/600
236/236 3s 11ms/step - accuracy: 0.7698 - loss: 0.6036 - val_accuracy: 0.7104 - val_loss: 0.7009
Epoch 247/600
236/236 2s 10ms/step - accuracy: 0.7719 - loss: 0.5992 - val_accuracy: 0.7084 - val_loss: 0.6913
Epoch 248/600
236/236 2s 10ms/step - accuracy: 0.7703 - loss: 0.5992 - val_accuracy: 0.7094 - val_loss: 0.7037
Epoch 249/600
236/236 3s 11ms/step - accuracy: 0.7691 - loss: 0.5994 - val_accuracy: 0.7026 - val_loss: 0.7176
Epoch 250/600
236/236 3s 11ms/step - accuracy: 0.7691 - loss: 0.5995 - val_accuracy: 0.7107 - val_loss: 0.6941
Epoch 251/600
236/236 3s 11ms/step - accuracy: 0.7719 - loss: 0.5995 - val_accuracy: 0.7097 - val_loss: 0.6966
Epoch 252/600
236/236 3s 11ms/step - accuracy: 0.7739 - loss: 0.5958 - val_accuracy: 0.7076 - val_loss: 0.7040
Epoch 253/600
236/236 2s 10ms/step - accuracy: 0.7728 - loss: 0.5955 - val_accuracy: 0.7026 - val_loss: 0.7071
Epoch 254/600

236/236 2s 10ms/step - accuracy: 0.7713 - loss: 0.5963 - val_accuracy: 0.6907 - val_loss: 0.7342
Epoch 255/600
236/236 2s 10ms/step - accuracy: 0.7721 - loss: 0.6009 - val_accuracy: 0.6813 - val_loss: 0.7575
Epoch 256/600
236/236 2s 10ms/step - accuracy: 0.7703 - loss: 0.6005 - val_accuracy: 0.7059 - val_loss: 0.7020
Epoch 257/600
236/236 3s 11ms/step - accuracy: 0.7713 - loss: 0.5983 - val_accuracy: 0.7028 - val_loss: 0.7180
Epoch 258/600
236/236 3s 11ms/step - accuracy: 0.7701 - loss: 0.6001 - val_accuracy: 0.7029 - val_loss: 0.7136
Epoch 259/600
236/236 3s 11ms/step - accuracy: 0.7721 - loss: 0.5977 - val_accuracy: 0.7006 - val_loss: 0.7162
Epoch 260/600
236/236 3s 11ms/step - accuracy: 0.7735 - loss: 0.5984 - val_accuracy: 0.7036 - val_loss: 0.7074
Epoch 261/600
236/236 3s 11ms/step - accuracy: 0.7728 - loss: 0.5956 - val_accuracy: 0.7112 - val_loss: 0.7050
Epoch 262/600
236/236 2s 10ms/step - accuracy: 0.7713 - loss: 0.5987 - val_accuracy: 0.7033 - val_loss: 0.7107
Epoch 263/600
236/236 3s 11ms/step - accuracy: 0.7711 - loss: 0.5961 - val_accuracy: 0.7140 - val_loss: 0.6853
Epoch 264/600
236/236 2s 10ms/step - accuracy: 0.7724 - loss: 0.5991 - val_accuracy: 0.7062 - val_loss: 0.7051
Epoch 265/600
236/236 3s 11ms/step - accuracy: 0.7720 - loss: 0.5987 - val_accuracy: 0.7075 - val_loss: 0.6978
Epoch 266/600
236/236 2s 10ms/step - accuracy: 0.7708 - loss: 0.5980 - val_accuracy: 0.7068 - val_loss: 0.7106
Epoch 267/600
236/236 2s 10ms/step - accuracy: 0.7737 - loss: 0.5939 - val_accuracy: 0.7055 - val_loss: 0.7077
Epoch 268/600
236/236 2s 10ms/step - accuracy: 0.7744 - loss: 0.5930 - val_accuracy: 0.7038 - val_loss: 0.7084
Epoch 269/600
236/236 2s 10ms/step - accuracy: 0.7725 - loss: 0.5971 - val_accuracy: 0.7128 - val_loss: 0.7018
Epoch 270/600
236/236 2s 10ms/step - accuracy: 0.7727 - loss: 0.5976 - val_accuracy: 0.7137 - val_loss: 0.7022
Epoch 271/600
236/236 2s 10ms/step - accuracy: 0.7741 - loss: 0.5932 - val_accuracy: 0.7038 - val_loss: 0.7170
Epoch 272/600
236/236 2s 10ms/step - accuracy: 0.7715 - loss: 0.5965 - val_accuracy: 0.7001 - val_loss: 0.7133
Epoch 273/600
236/236 2s 10ms/step - accuracy: 0.7742 - loss: 0.5940 - val_accuracy: 0.7105 - val_loss: 0.6964
Epoch 274/600
236/236 2s 10ms/step - accuracy: 0.7694 - loss: 0.6029 - val_accuracy: 0.7071 - val_loss: 0.7059
Epoch 275/600
236/236 2s 10ms/step - accuracy: 0.7713 - loss: 0.5990 - val_accuracy: 0.7052 - val_loss: 0.7083
Epoch 276/600
236/236 2s 10ms/step - accuracy: 0.7728 - loss: 0.5970 - val_accuracy: 0.7077 - val_loss: 0.7016
Epoch 277/600
236/236 2s 10ms/step - accuracy: 0.7723 - loss: 0.5969 - val_accuracy: 0.7142 - val_loss: 0.6916
Epoch 278/600
236/236 3s 11ms/step - accuracy: 0.7705 - loss: 0.5957 - val_accuracy: 0.7069 - val_loss: 0.7053
Epoch 279/600
236/236 3s 11ms/step - accuracy: 0.7732 - loss: 0.5969 - val_accuracy: 0.7132 - val_loss:

s: 0.6875
Epoch 280/600
236/236 2s 10ms/step - accuracy: 0.7717 - loss: 0.5980 - val_accuracy: 0.7070 - val_loss: 0.7140
Epoch 281/600
236/236 2s 10ms/step - accuracy: 0.7705 - loss: 0.5989 - val_accuracy: 0.7113 - val_loss: 0.7002
Epoch 282/600
236/236 3s 11ms/step - accuracy: 0.7739 - loss: 0.5924 - val_accuracy: 0.7069 - val_loss: 0.7028
Epoch 283/600
236/236 2s 10ms/step - accuracy: 0.7712 - loss: 0.5966 - val_accuracy: 0.7144 - val_loss: 0.6929
Epoch 284/600
236/236 3s 11ms/step - accuracy: 0.7732 - loss: 0.5940 - val_accuracy: 0.7081 - val_loss: 0.7018
Epoch 285/600
236/236 3s 11ms/step - accuracy: 0.7698 - loss: 0.6012 - val_accuracy: 0.7101 - val_loss: 0.6975
Epoch 286/600
236/236 2s 10ms/step - accuracy: 0.7738 - loss: 0.5946 - val_accuracy: 0.6997 - val_loss: 0.7156
s: 0.7108
Epoch 287/600
236/236 2s 10ms/step - accuracy: 0.7722 - loss: 0.5951 - val_accuracy: 0.7024 - val_loss: 0.6968
s: 0.7015
Epoch 288/600
236/236 2s 10ms/step - accuracy: 0.7732 - loss: 0.5950 - val_accuracy: 0.7102 - val_loss: 0.6938
s: 0.7041
Epoch 289/600
236/236 3s 12ms/step - accuracy: 0.7730 - loss: 0.5953 - val_accuracy: 0.7081 - val_loss: 0.7019
s: 0.7133
Epoch 290/600
236/236 3s 11ms/step - accuracy: 0.7716 - loss: 0.5967 - val_accuracy: 0.7049 - val_loss: 0.7072
s: 0.6938
Epoch 291/600
236/236 3s 11ms/step - accuracy: 0.7740 - loss: 0.5926 - val_accuracy: 0.7133 - val_loss: 0.7099
s: 0.7021
Epoch 292/600
236/236 2s 10ms/step - accuracy: 0.7699 - loss: 0.5983 - val_accuracy: 0.7099 - val_loss: 0.7019
s: 0.7019
Epoch 293/600
236/236 2s 10ms/step - accuracy: 0.7738 - loss: 0.5951 - val_accuracy: 0.7099 - val_loss: 0.6984
s: 0.7045
Epoch 294/600
236/236 2s 10ms/step - accuracy: 0.7745 - loss: 0.5906 - val_accuracy: 0.7108 - val_loss: 0.7024
s: 0.7092
Epoch 295/600
236/236 2s 10ms/step - accuracy: 0.7731 - loss: 0.5948 - val_accuracy: 0.7064 - val_loss: 0.7092
s: 0.7092
Epoch 296/600
236/236 3s 11ms/step - accuracy: 0.7736 - loss: 0.5916 - val_accuracy: 0.7098 - val_loss: 0.7076
s: 0.7099
Epoch 297/600
236/236 3s 11ms/step - accuracy: 0.7725 - loss: 0.5945 - val_accuracy: 0.7099 - val_loss: 0.7076
s: 0.7099
Epoch 298/600
236/236 3s 11ms/step - accuracy: 0.7741 - loss: 0.5900 - val_accuracy: 0.7039 - val_loss: 0.6951
s: 0.7076
Epoch 299/600
236/236 3s 11ms/step - accuracy: 0.7722 - loss: 0.5964 - val_accuracy: 0.7117 - val_loss: 0.6939
s: 0.7117
Epoch 300/600
236/236 3s 12ms/step - accuracy: 0.7743 - loss: 0.5910 - val_accuracy: 0.7145 - val_loss: 0.6993
s: 0.7087
Epoch 301/600
236/236 3s 11ms/step - accuracy: 0.7719 - loss: 0.5948 - val_accuracy: 0.7109 - val_loss: 0.7049
s: 0.7087
Epoch 302/600
236/236 3s 11ms/step - accuracy: 0.7743 - loss: 0.5935 - val_accuracy: 0.7087 - val_loss: 0.7049
s: 0.7049
Epoch 303/600
236/236 3s 11ms/step - accuracy: 0.7720 - loss: 0.5961 - val_accuracy: 0.7065 - val_loss: 0.7049
s: 0.7049
Epoch 304/600
236/236 2s 10ms/step - accuracy: 0.7736 - loss: 0.5910 - val_accuracy: 0.7008 - val_loss: 0.7097
s: 0.7097

Epoch 305/600
236/236 2s 10ms/step - accuracy: 0.7741 - loss: 0.5899 - val_accuracy: 0.7108 - val_loss: 0.6963
Epoch 306/600
236/236 3s 11ms/step - accuracy: 0.7712 - loss: 0.5953 - val_accuracy: 0.6997 - val_loss: 0.7179
Epoch 307/600
236/236 2s 10ms/step - accuracy: 0.7745 - loss: 0.5934 - val_accuracy: 0.7002 - val_loss: 0.7072
Epoch 308/600
236/236 2s 10ms/step - accuracy: 0.7750 - loss: 0.5885 - val_accuracy: 0.7149 - val_loss: 0.6855
Epoch 309/600
236/236 3s 11ms/step - accuracy: 0.7759 - loss: 0.5905 - val_accuracy: 0.7039 - val_loss: 0.7052
Epoch 310/600
236/236 3s 11ms/step - accuracy: 0.7767 - loss: 0.5905 - val_accuracy: 0.7066 - val_loss: 0.7090
Epoch 311/600
236/236 2s 10ms/step - accuracy: 0.7713 - loss: 0.5955 - val_accuracy: 0.7115 - val_loss: 0.7039
Epoch 312/600
236/236 2s 10ms/step - accuracy: 0.7737 - loss: 0.5925 - val_accuracy: 0.7042 - val_loss: 0.7115
Epoch 313/600
236/236 3s 11ms/step - accuracy: 0.7754 - loss: 0.5901 - val_accuracy: 0.7112 - val_loss: 0.6955
Epoch 314/600
236/236 2s 10ms/step - accuracy: 0.7706 - loss: 0.5970 - val_accuracy: 0.7032 - val_loss: 0.7077
Epoch 315/600
236/236 3s 11ms/step - accuracy: 0.7745 - loss: 0.5941 - val_accuracy: 0.7097 - val_loss: 0.7046
Epoch 316/600
236/236 3s 11ms/step - accuracy: 0.7736 - loss: 0.5929 - val_accuracy: 0.7102 - val_loss: 0.7016
Epoch 317/600
236/236 3s 11ms/step - accuracy: 0.7729 - loss: 0.5932 - val_accuracy: 0.7145 - val_loss: 0.6959
Epoch 318/600
236/236 2s 10ms/step - accuracy: 0.7744 - loss: 0.5924 - val_accuracy: 0.7054 - val_loss: 0.7071
Epoch 319/600
236/236 3s 11ms/step - accuracy: 0.7724 - loss: 0.5924 - val_accuracy: 0.7094 - val_loss: 0.6946
Epoch 320/600
236/236 3s 11ms/step - accuracy: 0.7743 - loss: 0.5890 - val_accuracy: 0.7116 - val_loss: 0.6985
Epoch 321/600
236/236 2s 10ms/step - accuracy: 0.7742 - loss: 0.5900 - val_accuracy: 0.7070 - val_loss: 0.6981
Epoch 322/600
236/236 3s 11ms/step - accuracy: 0.7717 - loss: 0.5951 - val_accuracy: 0.7105 - val_loss: 0.7012
Epoch 323/600
236/236 3s 11ms/step - accuracy: 0.7745 - loss: 0.5942 - val_accuracy: 0.7078 - val_loss: 0.7028
Epoch 324/600
236/236 2s 10ms/step - accuracy: 0.7750 - loss: 0.5910 - val_accuracy: 0.7111 - val_loss: 0.7011
Epoch 325/600
236/236 2s 10ms/step - accuracy: 0.7725 - loss: 0.5950 - val_accuracy: 0.7117 - val_loss: 0.7004
Epoch 326/600
236/236 2s 10ms/step - accuracy: 0.7758 - loss: 0.5875 - val_accuracy: 0.7050 - val_loss: 0.7036
Epoch 327/600
236/236 2s 10ms/step - accuracy: 0.7729 - loss: 0.5940 - val_accuracy: 0.7116 - val_loss: 0.6948
Epoch 328/600
236/236 3s 11ms/step - accuracy: 0.7754 - loss: 0.5911 - val_accuracy: 0.7137 - val_loss: 0.6917
Epoch 329/600
236/236 3s 11ms/step - accuracy: 0.7737 - loss: 0.5945 - val_accuracy: 0.7105 - val_loss: 0.7037
Epoch 330/600

236/236 3s 11ms/step - accuracy: 0.7728 - loss: 0.5922 - val_accuracy: 0.7152 - val_loss: 0.6847
Epoch 331/600
236/236 3s 11ms/step - accuracy: 0.7731 - loss: 0.5944 - val_accuracy: 0.7098 - val_loss: 0.6984
Epoch 332/600
236/236 2s 10ms/step - accuracy: 0.7739 - loss: 0.5921 - val_accuracy: 0.7082 - val_loss: 0.7145
Epoch 333/600
236/236 2s 10ms/step - accuracy: 0.7763 - loss: 0.5900 - val_accuracy: 0.7108 - val_loss: 0.6957
Epoch 334/600
236/236 2s 10ms/step - accuracy: 0.7736 - loss: 0.5921 - val_accuracy: 0.7095 - val_loss: 0.7008
Epoch 335/600
236/236 3s 11ms/step - accuracy: 0.7730 - loss: 0.5971 - val_accuracy: 0.7083 - val_loss: 0.7005
Epoch 336/600
236/236 3s 11ms/step - accuracy: 0.7744 - loss: 0.5905 - val_accuracy: 0.7096 - val_loss: 0.6967
Epoch 337/600
236/236 3s 11ms/step - accuracy: 0.7727 - loss: 0.5925 - val_accuracy: 0.7020 - val_loss: 0.7125
Epoch 338/600
236/236 3s 11ms/step - accuracy: 0.7775 - loss: 0.5891 - val_accuracy: 0.7082 - val_loss: 0.6975
Epoch 339/600
236/236 2s 10ms/step - accuracy: 0.7769 - loss: 0.5885 - val_accuracy: 0.7080 - val_loss: 0.7013
Epoch 340/600
236/236 2s 10ms/step - accuracy: 0.7745 - loss: 0.5904 - val_accuracy: 0.7042 - val_loss: 0.7076
Epoch 341/600
236/236 3s 11ms/step - accuracy: 0.7728 - loss: 0.5940 - val_accuracy: 0.7131 - val_loss: 0.6922
Epoch 342/600
236/236 3s 11ms/step - accuracy: 0.7750 - loss: 0.5921 - val_accuracy: 0.7045 - val_loss: 0.6999
Epoch 343/600
236/236 3s 11ms/step - accuracy: 0.7759 - loss: 0.5898 - val_accuracy: 0.7154 - val_loss: 0.6879
Epoch 344/600
236/236 3s 11ms/step - accuracy: 0.7733 - loss: 0.5932 - val_accuracy: 0.7159 - val_loss: 0.6928
Epoch 345/600
236/236 3s 12ms/step - accuracy: 0.7745 - loss: 0.5949 - val_accuracy: 0.7056 - val_loss: 0.7066
Epoch 346/600
236/236 3s 11ms/step - accuracy: 0.7775 - loss: 0.5888 - val_accuracy: 0.7096 - val_loss: 0.6972
Epoch 347/600
236/236 3s 11ms/step - accuracy: 0.7747 - loss: 0.5902 - val_accuracy: 0.7142 - val_loss: 0.6922
Epoch 348/600
236/236 2s 10ms/step - accuracy: 0.7747 - loss: 0.5933 - val_accuracy: 0.7104 - val_loss: 0.6964
Epoch 349/600
236/236 3s 11ms/step - accuracy: 0.7752 - loss: 0.5896 - val_accuracy: 0.7100 - val_loss: 0.6971
Epoch 350/600
236/236 3s 11ms/step - accuracy: 0.7759 - loss: 0.5884 - val_accuracy: 0.7042 - val_loss: 0.7144
Epoch 351/600
236/236 2s 10ms/step - accuracy: 0.7741 - loss: 0.5915 - val_accuracy: 0.7068 - val_loss: 0.7008
Epoch 352/600
236/236 3s 11ms/step - accuracy: 0.7742 - loss: 0.5931 - val_accuracy: 0.7126 - val_loss: 0.6967
Epoch 353/600
236/236 3s 12ms/step - accuracy: 0.7772 - loss: 0.5892 - val_accuracy: 0.7016 - val_loss: 0.7134
Epoch 354/600
236/236 3s 12ms/step - accuracy: 0.7761 - loss: 0.5894 - val_accuracy: 0.7061 - val_loss: 0.7122
Epoch 355/600
236/236 3s 12ms/step - accuracy: 0.7742 - loss: 0.5923 - val_accuracy: 0.7097 - val_loss:

s: 0.7008
Epoch 356/600
236/236 3s 11ms/step - accuracy: 0.7736 - loss: 0.5925 - val_accuracy: 0.7119 - val_loss: 0.6893
Epoch 357/600
236/236 3s 11ms/step - accuracy: 0.7752 - loss: 0.5924 - val_accuracy: 0.7121 - val_loss: 0.6951
Epoch 358/600
236/236 3s 11ms/step - accuracy: 0.7743 - loss: 0.5898 - val_accuracy: 0.7115 - val_loss: 0.6997
Epoch 359/600
236/236 3s 12ms/step - accuracy: 0.7752 - loss: 0.5913 - val_accuracy: 0.7039 - val_loss: 0.7090
s: 0.7033
Epoch 360/600
236/236 3s 11ms/step - accuracy: 0.7751 - loss: 0.5896 - val_accuracy: 0.7106 - val_loss: 0.6875
s: 0.7020
Epoch 361/600
236/236 3s 11ms/step - accuracy: 0.7749 - loss: 0.5890 - val_accuracy: 0.7162 - val_loss: 0.7020
s: 0.6875
Epoch 362/600
236/236 3s 11ms/step - accuracy: 0.7744 - loss: 0.5928 - val_accuracy: 0.7093 - val_loss: 0.7063
s: 0.6932
Epoch 363/600
236/236 2s 10ms/step - accuracy: 0.7764 - loss: 0.5878 - val_accuracy: 0.7173 - val_loss: 0.7063
s: 0.7016
Epoch 364/600
236/236 3s 11ms/step - accuracy: 0.7759 - loss: 0.5878 - val_accuracy: 0.7039 - val_loss: 0.6932
s: 0.7016
Epoch 365/600
236/236 3s 11ms/step - accuracy: 0.7778 - loss: 0.5866 - val_accuracy: 0.7141 - val_loss: 0.7029
s: 0.7061
Epoch 366/600
236/236 3s 11ms/step - accuracy: 0.7754 - loss: 0.5890 - val_accuracy: 0.7139 - val_loss: 0.7029
s: 0.7061
Epoch 367/600
236/236 3s 11ms/step - accuracy: 0.7747 - loss: 0.5905 - val_accuracy: 0.7096 - val_loss: 0.7054
s: 0.7029
Epoch 368/600
236/236 2s 10ms/step - accuracy: 0.7747 - loss: 0.5899 - val_accuracy: 0.7042 - val_loss: 0.7054
s: 0.7029
Epoch 369/600
236/236 3s 11ms/step - accuracy: 0.7771 - loss: 0.5885 - val_accuracy: 0.7047 - val_loss: 0.7046
s: 0.7068
Epoch 370/600
236/236 2s 10ms/step - accuracy: 0.7757 - loss: 0.5902 - val_accuracy: 0.7033 - val_loss: 0.7046
s: 0.7068
Epoch 371/600
236/236 2s 10ms/step - accuracy: 0.7747 - loss: 0.5909 - val_accuracy: 0.7200 - val_loss: 0.6978
s: 0.6877
Epoch 372/600
236/236 3s 11ms/step - accuracy: 0.7783 - loss: 0.5818 - val_accuracy: 0.7051 - val_loss: 0.6978
s: 0.7046
Epoch 373/600
236/236 3s 11ms/step - accuracy: 0.7754 - loss: 0.5919 - val_accuracy: 0.7164 - val_loss: 0.6944
s: 0.6944
Epoch 374/600
236/236 3s 12ms/step - accuracy: 0.7772 - loss: 0.5844 - val_accuracy: 0.7110 - val_loss: 0.6922
s: 0.6944
Epoch 375/600
236/236 3s 11ms/step - accuracy: 0.7732 - loss: 0.5905 - val_accuracy: 0.7109 - val_loss: 0.7238
s: 0.6922
Epoch 376/600
236/236 3s 11ms/step - accuracy: 0.7767 - loss: 0.5897 - val_accuracy: 0.7008 - val_loss: 0.6966
s: 0.7238
Epoch 377/600
236/236 3s 11ms/step - accuracy: 0.7764 - loss: 0.5875 - val_accuracy: 0.7110 - val_loss: 0.6966
s: 0.6966
Epoch 378/600
236/236 3s 11ms/step - accuracy: 0.7758 - loss: 0.5874 - val_accuracy: 0.7135 - val_loss: 0.7041
s: 0.6966
Epoch 379/600
236/236 3s 11ms/step - accuracy: 0.7763 - loss: 0.5876 - val_accuracy: 0.7047 - val_loss: 0.7024
s: 0.7041
Epoch 380/600
236/236 2s 10ms/step - accuracy: 0.7768 - loss: 0.5870 - val_accuracy: 0.7107 - val_loss: 0.7024
s: 0.7024

Epoch 381/600
236/236 2s 10ms/step - accuracy: 0.7736 - loss: 0.5917 - val_accuracy: 0.7142 - val_loss: 0.6924
Epoch 382/600
236/236 2s 10ms/step - accuracy: 0.7762 - loss: 0.5890 - val_accuracy: 0.7142 - val_loss: 0.6923
Epoch 383/600
236/236 2s 10ms/step - accuracy: 0.7761 - loss: 0.5900 - val_accuracy: 0.7135 - val_loss: 0.6919
Epoch 384/600
236/236 3s 11ms/step - accuracy: 0.7758 - loss: 0.5891 - val_accuracy: 0.7105 - val_loss: 0.6996
Epoch 385/600
236/236 3s 11ms/step - accuracy: 0.7763 - loss: 0.5881 - val_accuracy: 0.7086 - val_loss: 0.6998
Epoch 386/600
236/236 3s 11ms/step - accuracy: 0.7793 - loss: 0.5833 - val_accuracy: 0.7122 - val_loss: 0.7003
Epoch 387/600
236/236 3s 11ms/step - accuracy: 0.7761 - loss: 0.5860 - val_accuracy: 0.7003 - val_loss: 0.7136
Epoch 388/600
236/236 3s 11ms/step - accuracy: 0.7785 - loss: 0.5857 - val_accuracy: 0.7211 - val_loss: 0.6790
Epoch 389/600
236/236 3s 11ms/step - accuracy: 0.7775 - loss: 0.5838 - val_accuracy: 0.7094 - val_loss: 0.7024
Epoch 390/600
236/236 3s 11ms/step - accuracy: 0.7759 - loss: 0.5847 - val_accuracy: 0.7166 - val_loss: 0.6843
Epoch 391/600
236/236 3s 11ms/step - accuracy: 0.7782 - loss: 0.5850 - val_accuracy: 0.7031 - val_loss: 0.7127
Epoch 392/600
236/236 3s 11ms/step - accuracy: 0.7774 - loss: 0.5896 - val_accuracy: 0.7228 - val_loss: 0.6750
Epoch 393/600
236/236 2s 10ms/step - accuracy: 0.7771 - loss: 0.5849 - val_accuracy: 0.7060 - val_loss: 0.7063
Epoch 394/600
236/236 2s 10ms/step - accuracy: 0.7750 - loss: 0.5884 - val_accuracy: 0.7102 - val_loss: 0.7045
Epoch 395/600
236/236 3s 11ms/step - accuracy: 0.7764 - loss: 0.5869 - val_accuracy: 0.7144 - val_loss: 0.6924
Epoch 396/600
236/236 3s 11ms/step - accuracy: 0.7734 - loss: 0.5939 - val_accuracy: 0.7133 - val_loss: 0.7003
Epoch 397/600
236/236 3s 12ms/step - accuracy: 0.7797 - loss: 0.5822 - val_accuracy: 0.7154 - val_loss: 0.6904
Epoch 398/600
236/236 3s 11ms/step - accuracy: 0.7776 - loss: 0.5884 - val_accuracy: 0.7128 - val_loss: 0.6987
Epoch 399/600
236/236 3s 11ms/step - accuracy: 0.7791 - loss: 0.5826 - val_accuracy: 0.7150 - val_loss: 0.6923
Epoch 400/600
236/236 2s 10ms/step - accuracy: 0.7786 - loss: 0.5846 - val_accuracy: 0.7162 - val_loss: 0.6910
Epoch 401/600
236/236 2s 10ms/step - accuracy: 0.7786 - loss: 0.5850 - val_accuracy: 0.7023 - val_loss: 0.7203
Epoch 402/600
236/236 3s 11ms/step - accuracy: 0.7769 - loss: 0.5858 - val_accuracy: 0.7085 - val_loss: 0.7012
Epoch 403/600
236/236 3s 11ms/step - accuracy: 0.7750 - loss: 0.5891 - val_accuracy: 0.7192 - val_loss: 0.6883
Epoch 404/600
236/236 2s 10ms/step - accuracy: 0.7781 - loss: 0.5861 - val_accuracy: 0.7144 - val_loss: 0.6919
Epoch 405/600
236/236 2s 10ms/step - accuracy: 0.7756 - loss: 0.5891 - val_accuracy: 0.7091 - val_loss: 0.6988
Epoch 406/600

236/236 2s 10ms/step - accuracy: 0.7758 - loss: 0.5863 - val_accuracy: 0.7092 - val_loss: 0.6982
Epoch 407/600
236/236 2s 10ms/step - accuracy: 0.7780 - loss: 0.5868 - val_accuracy: 0.7120 - val_loss: 0.6964
Epoch 408/600
236/236 2s 10ms/step - accuracy: 0.7768 - loss: 0.5885 - val_accuracy: 0.7182 - val_loss: 0.6806
Epoch 409/600
236/236 2s 10ms/step - accuracy: 0.7770 - loss: 0.5863 - val_accuracy: 0.7041 - val_loss: 0.7076
Epoch 410/600
236/236 3s 12ms/step - accuracy: 0.7767 - loss: 0.5853 - val_accuracy: 0.7149 - val_loss: 0.6901
Epoch 411/600
236/236 3s 11ms/step - accuracy: 0.7804 - loss: 0.5800 - val_accuracy: 0.7194 - val_loss: 0.6840
Epoch 412/600
236/236 3s 11ms/step - accuracy: 0.7782 - loss: 0.5861 - val_accuracy: 0.7165 - val_loss: 0.6877
Epoch 413/600
236/236 3s 11ms/step - accuracy: 0.7766 - loss: 0.5883 - val_accuracy: 0.7185 - val_loss: 0.6848
Epoch 414/600
236/236 2s 10ms/step - accuracy: 0.7809 - loss: 0.5807 - val_accuracy: 0.7152 - val_loss: 0.6888
Epoch 415/600
236/236 2s 10ms/step - accuracy: 0.7797 - loss: 0.5842 - val_accuracy: 0.7125 - val_loss: 0.7009
Epoch 416/600
236/236 3s 11ms/step - accuracy: 0.7779 - loss: 0.5845 - val_accuracy: 0.7059 - val_loss: 0.7082
Epoch 417/600
236/236 2s 10ms/step - accuracy: 0.7765 - loss: 0.5868 - val_accuracy: 0.7202 - val_loss: 0.6828
Epoch 418/600
236/236 2s 10ms/step - accuracy: 0.7764 - loss: 0.5884 - val_accuracy: 0.7182 - val_loss: 0.6804
Epoch 419/600
236/236 2s 10ms/step - accuracy: 0.7769 - loss: 0.5870 - val_accuracy: 0.7173 - val_loss: 0.6893
Epoch 420/600
236/236 3s 11ms/step - accuracy: 0.7752 - loss: 0.5887 - val_accuracy: 0.7112 - val_loss: 0.6963
Epoch 421/600
236/236 3s 11ms/step - accuracy: 0.7767 - loss: 0.5898 - val_accuracy: 0.7133 - val_loss: 0.6953
Epoch 422/600
236/236 3s 11ms/step - accuracy: 0.7794 - loss: 0.5852 - val_accuracy: 0.7150 - val_loss: 0.6877
Epoch 423/600
236/236 3s 11ms/step - accuracy: 0.7781 - loss: 0.5849 - val_accuracy: 0.7198 - val_loss: 0.6871
Epoch 424/600
236/236 2s 10ms/step - accuracy: 0.7785 - loss: 0.5821 - val_accuracy: 0.6994 - val_loss: 0.7195
Epoch 425/600
236/236 3s 11ms/step - accuracy: 0.7786 - loss: 0.5841 - val_accuracy: 0.7151 - val_loss: 0.6895
Epoch 426/600
236/236 2s 10ms/step - accuracy: 0.7779 - loss: 0.5846 - val_accuracy: 0.7073 - val_loss: 0.7036
Epoch 427/600
236/236 2s 10ms/step - accuracy: 0.7768 - loss: 0.5862 - val_accuracy: 0.7122 - val_loss: 0.6972
Epoch 428/600
236/236 3s 11ms/step - accuracy: 0.7782 - loss: 0.5882 - val_accuracy: 0.7133 - val_loss: 0.7006
Epoch 429/600
236/236 3s 11ms/step - accuracy: 0.7767 - loss: 0.5861 - val_accuracy: 0.7184 - val_loss: 0.6831
Epoch 430/600
236/236 3s 11ms/step - accuracy: 0.7784 - loss: 0.5855 - val_accuracy: 0.7212 - val_loss: 0.6864
Epoch 431/600
236/236 3s 11ms/step - accuracy: 0.7773 - loss: 0.5840 - val_accuracy: 0.7157 - val_loss:

```
s: 0.6945
Epoch 432/600
236/236 3s 11ms/step - accuracy: 0.7771 - loss: 0.5849 - val_accuracy: 0.7137 - val_loss
s: 0.6892
Epoch 433/600
236/236 3s 11ms/step - accuracy: 0.7767 - loss: 0.5891 - val_accuracy: 0.7205 - val_loss
s: 0.6870
Epoch 434/600
236/236 3s 11ms/step - accuracy: 0.7801 - loss: 0.5832 - val_accuracy: 0.7099 - val_loss
s: 0.6959
Epoch 435/600
236/236 3s 11ms/step - accuracy: 0.7771 - loss: 0.5852 - val_accuracy: 0.7222 - val_loss
s: 0.6845
Epoch 436/600
236/236 3s 11ms/step - accuracy: 0.7793 - loss: 0.5819 - val_accuracy: 0.7197 - val_loss
s: 0.6808
Epoch 437/600
236/236 3s 11ms/step - accuracy: 0.7760 - loss: 0.5874 - val_accuracy: 0.7200 - val_loss
s: 0.6824
Epoch 438/600
236/236 2s 10ms/step - accuracy: 0.7796 - loss: 0.5834 - val_accuracy: 0.7161 - val_loss
s: 0.6942
Epoch 439/600
236/236 2s 10ms/step - accuracy: 0.7753 - loss: 0.5906 - val_accuracy: 0.7178 - val_loss
s: 0.6871
Epoch 440/600
236/236 3s 11ms/step - accuracy: 0.7794 - loss: 0.5846 - val_accuracy: 0.7176 - val_loss
s: 0.6862
Epoch 441/600
236/236 3s 11ms/step - accuracy: 0.7795 - loss: 0.5859 - val_accuracy: 0.7115 - val_loss
s: 0.6936
Epoch 442/600
236/236 2s 10ms/step - accuracy: 0.7777 - loss: 0.5848 - val_accuracy: 0.7184 - val_loss
s: 0.6813
Epoch 443/600
236/236 2s 10ms/step - accuracy: 0.7805 - loss: 0.5833 - val_accuracy: 0.7105 - val_loss
s: 0.7006
Epoch 444/600
236/236 2s 10ms/step - accuracy: 0.7780 - loss: 0.5837 - val_accuracy: 0.7153 - val_loss
s: 0.6954
Epoch 445/600
236/236 2s 10ms/step - accuracy: 0.7795 - loss: 0.5827 - val_accuracy: 0.7186 - val_loss
s: 0.6817
Epoch 446/600
236/236 2s 10ms/step - accuracy: 0.7785 - loss: 0.5828 - val_accuracy: 0.7166 - val_loss
s: 0.6938
Epoch 447/600
236/236 2s 10ms/step - accuracy: 0.7792 - loss: 0.5843 - val_accuracy: 0.7195 - val_loss
s: 0.6866
Epoch 448/600
236/236 3s 11ms/step - accuracy: 0.7794 - loss: 0.5857 - val_accuracy: 0.7106 - val_loss
s: 0.6994
Epoch 449/600
236/236 3s 11ms/step - accuracy: 0.7828 - loss: 0.5788 - val_accuracy: 0.7191 - val_loss
s: 0.6823
Epoch 450/600
236/236 3s 11ms/step - accuracy: 0.7775 - loss: 0.5853 - val_accuracy: 0.7188 - val_loss
s: 0.6816
Epoch 451/600
236/236 2s 10ms/step - accuracy: 0.7780 - loss: 0.5843 - val_accuracy: 0.7214 - val_loss
s: 0.6740
Epoch 452/600
236/236 3s 11ms/step - accuracy: 0.7764 - loss: 0.5853 - val_accuracy: 0.7199 - val_loss
s: 0.6813
```

```
In [44]: ann6_smote.evaluate(X_train_smote, y_train_smote)
```

```
3770/3770 4s 1ms/step - accuracy: 0.7852 - loss: 0.5382
```

```
Out[44]: [0.4545753598213196, 0.8376980423927307]
```

```
In [265... ann6_smote.summary()
```

```
Model: "sequential_9"
```

Layer (type)	Output Shape	Param #
dense_57 (Dense)	(None, 512)	23,040
batch_normalization_6 (BatchNormalization)	(None, 512)	2,048
dropout_48 (Dropout)	(None, 512)	0
dense_58 (Dense)	(None, 256)	131,328
batch_normalization_7 (BatchNormalization)	(None, 256)	1,024
dropout_49 (Dropout)	(None, 256)	0
dense_59 (Dense)	(None, 256)	65,792
batch_normalization_8 (BatchNormalization)	(None, 256)	1,024
dropout_50 (Dropout)	(None, 256)	0
dense_60 (Dense)	(None, 128)	32,896
batch_normalization_9 (BatchNormalization)	(None, 128)	512
dropout_51 (Dropout)	(None, 128)	0
dense_61 (Dense)	(None, 128)	16,512
batch_normalization_10 (BatchNormalization)	(None, 128)	512
dropout_52 (Dropout)	(None, 128)	0
dense_62 (Dense)	(None, 64)	8,256
batch_normalization_11 (BatchNormalization)	(None, 64)	256
dropout_53 (Dropout)	(None, 64)	0
dense_63 (Dense)	(None, 3)	195

Total params: 844,811 (3.22 MB)

Trainable params: 280,707 (1.07 MB)

Non-trainable params: 2,688 (10.50 KB)

Optimizer params: 561,416 (2.14 MB)

In [247...]: eval_metric(ann6_smote, X_train_smote, y_train_smote, X_val, y_val)

3770/3770 ————— 4s 1ms/step
 591/591 ————— 1s 958us/step

Test Set:

```
[[4784 512 212]
 [1954 6358 1741]
 [ 91 362 2889]]
```

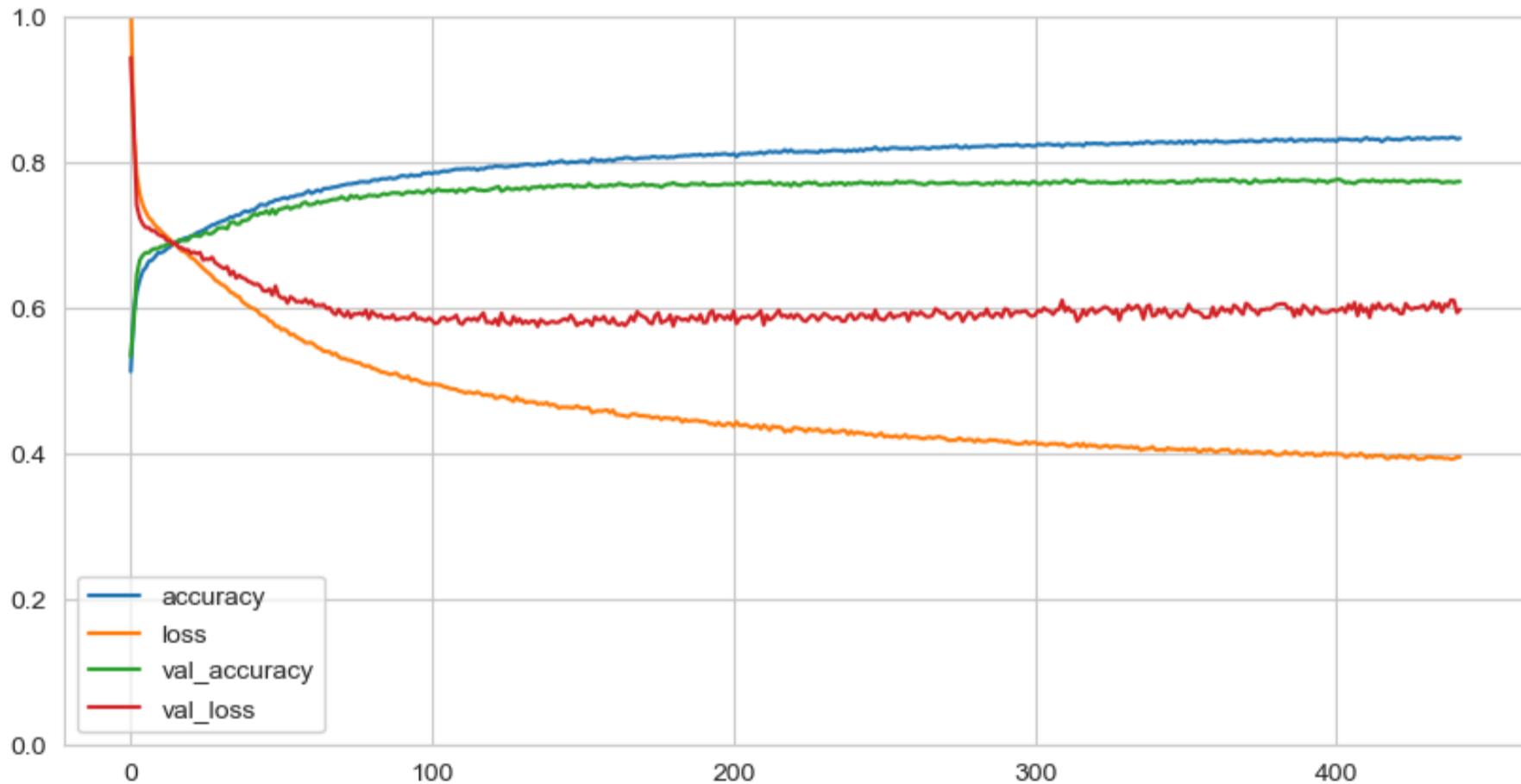
	precision	recall	f1-score	support
0	0.70	0.87	0.78	5508
1	0.88	0.63	0.74	10053
2	0.60	0.86	0.71	3342
accuracy			0.74	18903
macro avg	0.73	0.79	0.74	18903
weighted avg	0.78	0.74	0.74	18903

Train Set:

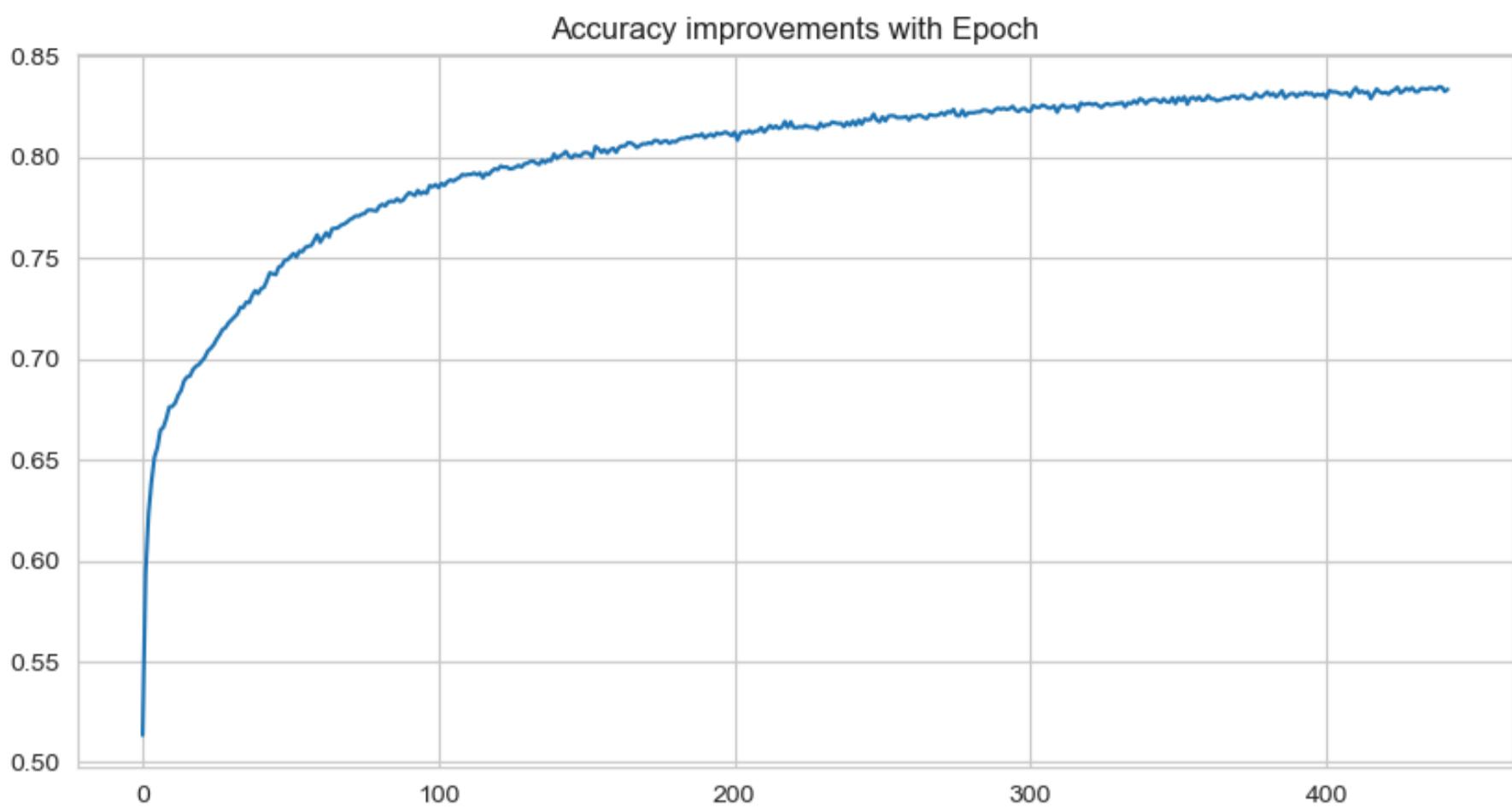
```
[[39629 384 196]
 [ 5751 29113 5345]
 [ 1 19 40189]]
```

	precision	recall	f1-score	support
0	0.87	0.99	0.93	40209
1	0.99	0.72	0.84	40209
2	0.88	1.00	0.94	40209
accuracy			0.90	120627
macro avg	0.91	0.90	0.90	120627
weighted avg	0.91	0.90	0.90	120627

```
In [266...]: pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



```
In [267...]: pd.DataFrame(history.history)[“accuracy”].plot(figsize=(10, 5))
plt.title("Accuracy improvements with Epoch")
plt.show()
```



```
In [ ]: # Save the Model

from tensorflow.keras.models import load_model

# Save the model to 'model.h5' file
ann6_smote.save('ann6_smote_model.h5')

# To Load the model later for further use
loaded_ann6_smote = load_model('ann6_smote_model.h5')
```

ANN-6 Model with SMOTE Summary:

- **(Dense) layers:** 7 / **Neurons:** 512-256-256-128-128-64 / **Dropout:** 30-30-25-25-20-20% / **Learning Rate:** 0.001 / **Batch Size:** 512 / **Epochs:** 600 / **Early Stop (val_accuracy):** 60
- **Accuracy:** 0.90 / **Val_Accuracy:** 0.74 / **Loss:** 0.5382 / **Val_Loss:** 0.6823 / **Train Recall (Class 2):** 1.00 / **Test Recall (Class 2):** 0.86

Improvements: The training performance is strong, particularly in recall for Class 2, where the model achieves a perfect score. This indicates a high sensitivity towards the minority class, possibly due to the use of SMOTE to balance class distribution.

No Improvement: Validation accuracy is lower compared to training accuracy, which might indicate the model is not generalizing well to unseen data. This could be an area for improvement to ensure the model performs consistently across different datasets.

Got Worse: The perfect recall of 1.00 in the training set may indicate overfitting, as the model is overly adapted to the training data and may not perform as well on new, unseen data. This aspect could be improved by introducing more regularization or adjusting the model's complexity to better generalize across different data scenarios.

ANN-7 Model -Dense + ReduceLROnPlateau (%88)

```
In [45]: from tensorflow.keras.callbacks import ReduceLROnPlateau

# ANN-7 Model:

# Added ReduceLROnPlateau to callbacks
# 1) Optimized Model Architecture:
ann7 = Sequential([
    Dense(512, input_dim=X_train.shape[1], activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(256, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
```

```
Dense(128, activation='relu'),
BatchNormalization(),
Dropout(0.2),
Dense(64, activation='relu'),
BatchNormalization(),
Dropout(0.2),
Dense(3, activation='softmax')
])

# 2) Compiling the Model:
ann7.compile(optimizer=Adam(learning_rate=0.001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 3) Early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                                patience=60,
                                mode="auto",
                                restore_best_weights=True)

# 4) ReduceLROnPlateau callback
reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                              factor=0.5, # Learning rate reduced by half if no improvement
                              patience=10, # Wait 10 epochs before reducing Learning rate
                              min_lr=1e-6, # Minimum learning rate set
                              verbose=1)

# 5) Train the model
history = ann7.fit(
    x=X_train,
    y=y_train,
    validation_data=(X_val, y_val),
    batch_size=512,
    epochs=600,
    verbose=1,
    callbacks=[early_stopping, reduce_lr]) # Added ReduceLROnPlateau to callbacks
```

Epoch 1/600
148/148 5s 9ms/step - accuracy: 0.4899 - loss: 1.1369 - val_accuracy: 0.5438 - val_loss: 0.8812 - learning_rate: 0.0010
Epoch 2/600
148/148 1s 7ms/step - accuracy: 0.6096 - loss: 0.8204 - val_accuracy: 0.5921 - val_loss: 0.7883 - learning_rate: 0.0010
Epoch 3/600
148/148 1s 7ms/step - accuracy: 0.6358 - loss: 0.7674 - val_accuracy: 0.6504 - val_loss: 0.7332 - learning_rate: 0.0010
Epoch 4/600
148/148 1s 7ms/step - accuracy: 0.6529 - loss: 0.7409 - val_accuracy: 0.6687 - val_loss: 0.7176 - learning_rate: 0.0010
Epoch 5/600
148/148 1s 7ms/step - accuracy: 0.6597 - loss: 0.7301 - val_accuracy: 0.6764 - val_loss: 0.7094 - learning_rate: 0.0010
Epoch 6/600
148/148 1s 8ms/step - accuracy: 0.6685 - loss: 0.7199 - val_accuracy: 0.6795 - val_loss: 0.7062 - learning_rate: 0.0010
Epoch 7/600
148/148 1s 7ms/step - accuracy: 0.6732 - loss: 0.7130 - val_accuracy: 0.6765 - val_loss: 0.7057 - learning_rate: 0.0010
Epoch 8/600
148/148 1s 7ms/step - accuracy: 0.6791 - loss: 0.7042 - val_accuracy: 0.6820 - val_loss: 0.7013 - learning_rate: 0.0010
Epoch 9/600
148/148 1s 7ms/step - accuracy: 0.6808 - loss: 0.6988 - val_accuracy: 0.6835 - val_loss: 0.6984 - learning_rate: 0.0010
Epoch 10/600
148/148 1s 8ms/step - accuracy: 0.6870 - loss: 0.6923 - val_accuracy: 0.6891 - val_loss: 0.6935 - learning_rate: 0.0010
Epoch 11/600
148/148 1s 8ms/step - accuracy: 0.6874 - loss: 0.6893 - val_accuracy: 0.6922 - val_loss: 0.6889 - learning_rate: 0.0010
Epoch 12/600
148/148 1s 8ms/step - accuracy: 0.6913 - loss: 0.6833 - val_accuracy: 0.6878 - val_loss: 0.6920 - learning_rate: 0.0010
Epoch 13/600
148/148 1s 8ms/step - accuracy: 0.6944 - loss: 0.6788 - val_accuracy: 0.6929 - val_loss: 0.6867 - learning_rate: 0.0010
Epoch 14/600
148/148 1s 8ms/step - accuracy: 0.6965 - loss: 0.6734 - val_accuracy: 0.6956 - val_loss: 0.6826 - learning_rate: 0.0010
Epoch 15/600
148/148 1s 9ms/step - accuracy: 0.7029 - loss: 0.6669 - val_accuracy: 0.6910 - val_loss: 0.6861 - learning_rate: 0.0010
Epoch 16/600
148/148 1s 7ms/step - accuracy: 0.7035 - loss: 0.6605 - val_accuracy: 0.6984 - val_loss: 0.6768 - learning_rate: 0.0010
Epoch 17/600
148/148 1s 8ms/step - accuracy: 0.7040 - loss: 0.6578 - val_accuracy: 0.6998 - val_loss: 0.6739 - learning_rate: 0.0010
Epoch 18/600
148/148 1s 8ms/step - accuracy: 0.7027 - loss: 0.6601 - val_accuracy: 0.7018 - val_loss: 0.6723 - learning_rate: 0.0010
Epoch 19/600
148/148 1s 7ms/step - accuracy: 0.7102 - loss: 0.6500 - val_accuracy: 0.6997 - val_loss: 0.6703 - learning_rate: 0.0010
Epoch 20/600
148/148 1s 8ms/step - accuracy: 0.7122 - loss: 0.6461 - val_accuracy: 0.7042 - val_loss: 0.6669 - learning_rate: 0.0010
Epoch 21/600
148/148 1s 7ms/step - accuracy: 0.7133 - loss: 0.6427 - val_accuracy: 0.7076 - val_loss: 0.6618 - learning_rate: 0.0010
Epoch 22/600
148/148 1s 7ms/step - accuracy: 0.7142 - loss: 0.6361 - val_accuracy: 0.7075 - val_loss: 0.6642 - learning_rate: 0.0010
Epoch 23/600
148/148 1s 8ms/step - accuracy: 0.7195 - loss: 0.6333 - val_accuracy: 0.7086 - val_loss: 0.6631 - learning_rate: 0.0010
Epoch 24/600
148/148 1s 7ms/step - accuracy: 0.7219 - loss: 0.6292 - val_accuracy: 0.7117 - val_loss: 0.6543 - learning_rate: 0.0010
Epoch 25/600
148/148 1s 7ms/step - accuracy: 0.7207 - loss: 0.6260 - val_accuracy: 0.7121 - val_loss: 0.6560 - learning_rate: 0.0010
Epoch 26/600

148/148 1s 8ms/step - accuracy: 0.7246 - loss: 0.6231 - val_accuracy: 0.7132 - val_loss: 0.6532 - learning_rate: 0.0010
Epoch 27/600
148/148 1s 9ms/step - accuracy: 0.7277 - loss: 0.6166 - val_accuracy: 0.7127 - val_loss: 0.6496 - learning_rate: 0.0010
Epoch 28/600
148/148 1s 8ms/step - accuracy: 0.7250 - loss: 0.6156 - val_accuracy: 0.7144 - val_loss: 0.6498 - learning_rate: 0.0010
Epoch 29/600
148/148 1s 8ms/step - accuracy: 0.7322 - loss: 0.6069 - val_accuracy: 0.7195 - val_loss: 0.6402 - learning_rate: 0.0010
Epoch 30/600
148/148 1s 8ms/step - accuracy: 0.7295 - loss: 0.6065 - val_accuracy: 0.7203 - val_loss: 0.6403 - learning_rate: 0.0010
Epoch 31/600
148/148 1s 8ms/step - accuracy: 0.7322 - loss: 0.6014 - val_accuracy: 0.7187 - val_loss: 0.6431 - learning_rate: 0.0010
Epoch 32/600
148/148 1s 7ms/step - accuracy: 0.7392 - loss: 0.5926 - val_accuracy: 0.7194 - val_loss: 0.6357 - learning_rate: 0.0010
Epoch 33/600
148/148 1s 8ms/step - accuracy: 0.7338 - loss: 0.5991 - val_accuracy: 0.7213 - val_loss: 0.6372 - learning_rate: 0.0010
Epoch 34/600
148/148 1s 8ms/step - accuracy: 0.7374 - loss: 0.5929 - val_accuracy: 0.7239 - val_loss: 0.6342 - learning_rate: 0.0010
Epoch 35/600
148/148 1s 7ms/step - accuracy: 0.7403 - loss: 0.5858 - val_accuracy: 0.7230 - val_loss: 0.6356 - learning_rate: 0.0010
Epoch 36/600
148/148 1s 8ms/step - accuracy: 0.7438 - loss: 0.5821 - val_accuracy: 0.7274 - val_loss: 0.6298 - learning_rate: 0.0010
Epoch 37/600
148/148 1s 7ms/step - accuracy: 0.7414 - loss: 0.5838 - val_accuracy: 0.7321 - val_loss: 0.6242 - learning_rate: 0.0010
Epoch 38/600
148/148 1s 7ms/step - accuracy: 0.7437 - loss: 0.5810 - val_accuracy: 0.7329 - val_loss: 0.6330 - learning_rate: 0.0010
Epoch 39/600
148/148 1s 7ms/step - accuracy: 0.7483 - loss: 0.5743 - val_accuracy: 0.7323 - val_loss: 0.6252 - learning_rate: 0.0010
Epoch 40/600
148/148 1s 8ms/step - accuracy: 0.7466 - loss: 0.5732 - val_accuracy: 0.7300 - val_loss: 0.6213 - learning_rate: 0.0010
Epoch 41/600
148/148 1s 8ms/step - accuracy: 0.7477 - loss: 0.5705 - val_accuracy: 0.7342 - val_loss: 0.6202 - learning_rate: 0.0010
Epoch 42/600
148/148 1s 8ms/step - accuracy: 0.7520 - loss: 0.5666 - val_accuracy: 0.7341 - val_loss: 0.6183 - learning_rate: 0.0010
Epoch 43/600
148/148 1s 8ms/step - accuracy: 0.7519 - loss: 0.5645 - val_accuracy: 0.7373 - val_loss: 0.6172 - learning_rate: 0.0010
Epoch 44/600
148/148 1s 7ms/step - accuracy: 0.7538 - loss: 0.5579 - val_accuracy: 0.7382 - val_loss: 0.6167 - learning_rate: 0.0010
Epoch 45/600
148/148 1s 8ms/step - accuracy: 0.7564 - loss: 0.5558 - val_accuracy: 0.7360 - val_loss: 0.6165 - learning_rate: 0.0010
Epoch 46/600
148/148 1s 8ms/step - accuracy: 0.7550 - loss: 0.5587 - val_accuracy: 0.7373 - val_loss: 0.6118 - learning_rate: 0.0010
Epoch 47/600
148/148 1s 8ms/step - accuracy: 0.7558 - loss: 0.5532 - val_accuracy: 0.7373 - val_loss: 0.6131 - learning_rate: 0.0010
Epoch 48/600
148/148 1s 8ms/step - accuracy: 0.7588 - loss: 0.5515 - val_accuracy: 0.7392 - val_loss: 0.6109 - learning_rate: 0.0010
Epoch 49/600
148/148 1s 8ms/step - accuracy: 0.7649 - loss: 0.5375 - val_accuracy: 0.7418 - val_loss: 0.6141 - learning_rate: 0.0010
Epoch 50/600
148/148 1s 7ms/step - accuracy: 0.7588 - loss: 0.5500 - val_accuracy: 0.7392 - val_loss: 0.6144 - learning_rate: 0.0010
Epoch 51/600
148/148 1s 7ms/step - accuracy: 0.7624 - loss: 0.5449 - val_accuracy: 0.7409 - val_loss:

```
s: 0.6119 - learning_rate: 0.0010
Epoch 52/600
148/148 1s 7ms/step - accuracy: 0.7624 - loss: 0.5416 - val_accuracy: 0.7413 - val_loss
s: 0.6146 - learning_rate: 0.0010
Epoch 53/600
148/148 1s 8ms/step - accuracy: 0.7651 - loss: 0.5393 - val_accuracy: 0.7418 - val_loss
s: 0.6034 - learning_rate: 0.0010
Epoch 54/600
148/148 1s 8ms/step - accuracy: 0.7666 - loss: 0.5367 - val_accuracy: 0.7409 - val_loss
s: 0.6080 - learning_rate: 0.0010
Epoch 55/600
148/148 1s 8ms/step - accuracy: 0.7665 - loss: 0.5344 - val_accuracy: 0.7449 - val_loss
s: 0.6015 - learning_rate: 0.0010
Epoch 56/600
148/148 1s 8ms/step - accuracy: 0.7693 - loss: 0.5286 - val_accuracy: 0.7446 - val_loss
s: 0.6046 - learning_rate: 0.0010
Epoch 57/600
148/148 1s 7ms/step - accuracy: 0.7651 - loss: 0.5316 - val_accuracy: 0.7455 - val_loss
s: 0.6043 - learning_rate: 0.0010
Epoch 58/600
148/148 1s 7ms/step - accuracy: 0.7713 - loss: 0.5247 - val_accuracy: 0.7457 - val_loss
s: 0.6019 - learning_rate: 0.0010
Epoch 59/600
148/148 1s 7ms/step - accuracy: 0.7699 - loss: 0.5287 - val_accuracy: 0.7426 - val_loss
s: 0.6063 - learning_rate: 0.0010
Epoch 60/600
148/148 1s 7ms/step - accuracy: 0.7707 - loss: 0.5276 - val_accuracy: 0.7447 - val_loss
s: 0.6019 - learning_rate: 0.0010
Epoch 61/600
148/148 1s 8ms/step - accuracy: 0.7715 - loss: 0.5248 - val_accuracy: 0.7481 - val_loss
s: 0.5987 - learning_rate: 0.0010
Epoch 62/600
148/148 1s 8ms/step - accuracy: 0.7722 - loss: 0.5213 - val_accuracy: 0.7476 - val_loss
s: 0.5986 - learning_rate: 0.0010
Epoch 63/600
148/148 1s 8ms/step - accuracy: 0.7723 - loss: 0.5202 - val_accuracy: 0.7486 - val_loss
s: 0.6005 - learning_rate: 0.0010
Epoch 64/600
148/148 1s 7ms/step - accuracy: 0.7741 - loss: 0.5181 - val_accuracy: 0.7478 - val_loss
s: 0.5978 - learning_rate: 0.0010
Epoch 65/600
148/148 1s 7ms/step - accuracy: 0.7752 - loss: 0.5152 - val_accuracy: 0.7524 - val_loss
s: 0.5930 - learning_rate: 0.0010
Epoch 66/600
148/148 1s 7ms/step - accuracy: 0.7773 - loss: 0.5109 - val_accuracy: 0.7490 - val_loss
s: 0.6015 - learning_rate: 0.0010
Epoch 67/600
148/148 1s 8ms/step - accuracy: 0.7773 - loss: 0.5094 - val_accuracy: 0.7512 - val_loss
s: 0.6030 - learning_rate: 0.0010
Epoch 68/600
148/148 1s 8ms/step - accuracy: 0.7776 - loss: 0.5111 - val_accuracy: 0.7516 - val_loss
s: 0.5944 - learning_rate: 0.0010
Epoch 69/600
148/148 1s 9ms/step - accuracy: 0.7808 - loss: 0.5046 - val_accuracy: 0.7511 - val_loss
s: 0.5925 - learning_rate: 0.0010
Epoch 70/600
148/148 1s 8ms/step - accuracy: 0.7764 - loss: 0.5107 - val_accuracy: 0.7524 - val_loss
s: 0.5995 - learning_rate: 0.0010
Epoch 71/600
148/148 1s 8ms/step - accuracy: 0.7809 - loss: 0.5066 - val_accuracy: 0.7546 - val_loss
s: 0.5957 - learning_rate: 0.0010
Epoch 72/600
148/148 1s 8ms/step - accuracy: 0.7793 - loss: 0.5078 - val_accuracy: 0.7513 - val_loss
s: 0.5981 - learning_rate: 0.0010
Epoch 73/600
148/148 1s 8ms/step - accuracy: 0.7806 - loss: 0.5048 - val_accuracy: 0.7519 - val_loss
s: 0.6013 - learning_rate: 0.0010
Epoch 74/600
148/148 1s 8ms/step - accuracy: 0.7767 - loss: 0.5052 - val_accuracy: 0.7524 - val_loss
s: 0.5927 - learning_rate: 0.0010
Epoch 75/600
148/148 1s 8ms/step - accuracy: 0.7827 - loss: 0.4993 - val_accuracy: 0.7547 - val_loss
s: 0.5928 - learning_rate: 0.0010
Epoch 76/600
148/148 1s 8ms/step - accuracy: 0.7846 - loss: 0.4998 - val_accuracy: 0.7528 - val_loss
s: 0.5943 - learning_rate: 0.0010
```

Epoch 77/600
148/148 1s 8ms/step - accuracy: 0.7835 - loss: 0.4958 - val_accuracy: 0.7512 - val_loss: 0.5952 - learning_rate: 0.0010
Epoch 78/600
148/148 1s 7ms/step - accuracy: 0.7843 - loss: 0.4997 - val_accuracy: 0.7575 - val_loss: 0.5876 - learning_rate: 0.0010
Epoch 79/600
148/148 1s 8ms/step - accuracy: 0.7819 - loss: 0.5012 - val_accuracy: 0.7555 - val_loss: 0.5921 - learning_rate: 0.0010
Epoch 80/600
148/148 1s 9ms/step - accuracy: 0.7836 - loss: 0.4973 - val_accuracy: 0.7585 - val_loss: 0.5891 - learning_rate: 0.0010
Epoch 81/600
148/148 1s 9ms/step - accuracy: 0.7843 - loss: 0.4957 - val_accuracy: 0.7554 - val_loss: 0.5932 - learning_rate: 0.0010
Epoch 82/600
148/148 1s 8ms/step - accuracy: 0.7862 - loss: 0.4936 - val_accuracy: 0.7567 - val_loss: 0.5976 - learning_rate: 0.0010
Epoch 83/600
148/148 1s 9ms/step - accuracy: 0.7840 - loss: 0.4990 - val_accuracy: 0.7550 - val_loss: 0.5918 - learning_rate: 0.0010
Epoch 84/600
148/148 1s 7ms/step - accuracy: 0.7867 - loss: 0.4899 - val_accuracy: 0.7568 - val_loss: 0.5880 - learning_rate: 0.0010
Epoch 85/600
148/148 1s 10ms/step - accuracy: 0.7880 - loss: 0.4879 - val_accuracy: 0.7562 - val_loss: 0.5915 - learning_rate: 0.0010
Epoch 86/600
148/148 1s 8ms/step - accuracy: 0.7893 - loss: 0.4895 - val_accuracy: 0.7570 - val_loss: 0.5910 - learning_rate: 0.0010
Epoch 87/600
148/148 1s 8ms/step - accuracy: 0.7883 - loss: 0.4898 - val_accuracy: 0.7592 - val_loss: 0.5920 - learning_rate: 0.0010
Epoch 88/600
142/148 0s 7ms/step - accuracy: 0.7851 - loss: 0.4912
Epoch 88: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
148/148 1s 7ms/step - accuracy: 0.7850 - loss: 0.4914 - val_accuracy: 0.7588 - val_loss: 0.5889 - learning_rate: 0.0010
Epoch 89/600
148/148 1s 7ms/step - accuracy: 0.7936 - loss: 0.4769 - val_accuracy: 0.7624 - val_loss: 0.5951 - learning_rate: 5.0000e-04
Epoch 90/600
148/148 1s 7ms/step - accuracy: 0.7952 - loss: 0.4751 - val_accuracy: 0.7623 - val_loss: 0.5874 - learning_rate: 5.0000e-04
Epoch 91/600
148/148 1s 7ms/step - accuracy: 0.7950 - loss: 0.4763 - val_accuracy: 0.7652 - val_loss: 0.5870 - learning_rate: 5.0000e-04
Epoch 92/600
148/148 1s 7ms/step - accuracy: 0.7981 - loss: 0.4710 - val_accuracy: 0.7631 - val_loss: 0.5878 - learning_rate: 5.0000e-04
Epoch 93/600
148/148 1s 7ms/step - accuracy: 0.7994 - loss: 0.4667 - val_accuracy: 0.7625 - val_loss: 0.5936 - learning_rate: 5.0000e-04
Epoch 94/600
148/148 1s 8ms/step - accuracy: 0.7970 - loss: 0.4670 - val_accuracy: 0.7642 - val_loss: 0.5875 - learning_rate: 5.0000e-04
Epoch 95/600
148/148 1s 8ms/step - accuracy: 0.7994 - loss: 0.4620 - val_accuracy: 0.7653 - val_loss: 0.5890 - learning_rate: 5.0000e-04
Epoch 96/600
148/148 1s 8ms/step - accuracy: 0.8005 - loss: 0.4618 - val_accuracy: 0.7642 - val_loss: 0.5916 - learning_rate: 5.0000e-04
Epoch 97/600
148/148 1s 8ms/step - accuracy: 0.8038 - loss: 0.4594 - val_accuracy: 0.7659 - val_loss: 0.5880 - learning_rate: 5.0000e-04
Epoch 98/600
148/148 1s 7ms/step - accuracy: 0.8014 - loss: 0.4642 - val_accuracy: 0.7653 - val_loss: 0.5874 - learning_rate: 5.0000e-04
Epoch 99/600
148/148 1s 8ms/step - accuracy: 0.8030 - loss: 0.4581 - val_accuracy: 0.7645 - val_loss: 0.5866 - learning_rate: 5.0000e-04
Epoch 100/600
148/148 1s 7ms/step - accuracy: 0.8002 - loss: 0.4640 - val_accuracy: 0.7638 - val_loss: 0.5862 - learning_rate: 5.0000e-04
Epoch 101/600
148/148 1s 7ms/step - accuracy: 0.8020 - loss: 0.4628 - val_accuracy: 0.7644 - val_loss:

```
s: 0.5927 - learning_rate: 5.0000e-04
Epoch 102/600
148/148 ----- 1s 8ms/step - accuracy: 0.8042 - loss: 0.4588 - val_accuracy: 0.7632 - val_loss
s: 0.5851 - learning_rate: 5.0000e-04
Epoch 103/600
148/148 ----- 1s 8ms/step - accuracy: 0.8015 - loss: 0.4600 - val_accuracy: 0.7669 - val_loss
s: 0.5871 - learning_rate: 5.0000e-04
Epoch 104/600
148/148 ----- 1s 8ms/step - accuracy: 0.8031 - loss: 0.4560 - val_accuracy: 0.7669 - val_loss
s: 0.5864 - learning_rate: 5.0000e-04
Epoch 105/600
148/148 ----- 1s 9ms/step - accuracy: 0.8050 - loss: 0.4529 - val_accuracy: 0.7667 - val_loss
s: 0.5878 - learning_rate: 5.0000e-04
Epoch 106/600
148/148 ----- 1s 8ms/step - accuracy: 0.8019 - loss: 0.4550 - val_accuracy: 0.7642 - val_loss
s: 0.5942 - learning_rate: 5.0000e-04
Epoch 107/600
148/148 ----- 1s 7ms/step - accuracy: 0.8069 - loss: 0.4505 - val_accuracy: 0.7638 - val_loss
s: 0.5873 - learning_rate: 5.0000e-04
Epoch 108/600
148/148 ----- 1s 8ms/step - accuracy: 0.8013 - loss: 0.4602 - val_accuracy: 0.7662 - val_loss
s: 0.5895 - learning_rate: 5.0000e-04
Epoch 109/600
148/148 ----- 1s 9ms/step - accuracy: 0.8032 - loss: 0.4558 - val_accuracy: 0.7659 - val_loss
s: 0.5920 - learning_rate: 5.0000e-04
Epoch 110/600
148/148 ----- 1s 9ms/step - accuracy: 0.8076 - loss: 0.4485 - val_accuracy: 0.7642 - val_loss
s: 0.5911 - learning_rate: 5.0000e-04
Epoch 111/600
148/148 ----- 1s 8ms/step - accuracy: 0.8044 - loss: 0.4576 - val_accuracy: 0.7642 - val_loss
s: 0.5887 - learning_rate: 5.0000e-04
Epoch 112/600
143/148 ----- 0s 8ms/step - accuracy: 0.8055 - loss: 0.4495
Epoch 112: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
148/148 ----- 1s 8ms/step - accuracy: 0.8055 - loss: 0.4497 - val_accuracy: 0.7656 - val_loss
s: 0.5901 - learning_rate: 5.0000e-04
Epoch 113/600
148/148 ----- 1s 7ms/step - accuracy: 0.8104 - loss: 0.4468 - val_accuracy: 0.7666 - val_loss
s: 0.5886 - learning_rate: 2.5000e-04
Epoch 114/600
148/148 ----- 1s 7ms/step - accuracy: 0.8083 - loss: 0.4453 - val_accuracy: 0.7658 - val_loss
s: 0.5928 - learning_rate: 2.5000e-04
Epoch 115/600
148/148 ----- 1s 9ms/step - accuracy: 0.8089 - loss: 0.4383 - val_accuracy: 0.7660 - val_loss
s: 0.5925 - learning_rate: 2.5000e-04
Epoch 116/600
148/148 ----- 1s 8ms/step - accuracy: 0.8079 - loss: 0.4449 - val_accuracy: 0.7682 - val_loss
s: 0.5903 - learning_rate: 2.5000e-04
Epoch 117/600
148/148 ----- 1s 8ms/step - accuracy: 0.8076 - loss: 0.4431 - val_accuracy: 0.7673 - val_loss
s: 0.5913 - learning_rate: 2.5000e-04
Epoch 118/600
148/148 ----- 1s 7ms/step - accuracy: 0.8137 - loss: 0.4371 - val_accuracy: 0.7672 - val_loss
s: 0.5934 - learning_rate: 2.5000e-04
Epoch 119/600
148/148 ----- 1s 7ms/step - accuracy: 0.8131 - loss: 0.4377 - val_accuracy: 0.7677 - val_loss
s: 0.5911 - learning_rate: 2.5000e-04
Epoch 120/600
148/148 ----- 1s 7ms/step - accuracy: 0.8124 - loss: 0.4392 - val_accuracy: 0.7671 - val_loss
s: 0.5945 - learning_rate: 2.5000e-04
Epoch 121/600
148/148 ----- 1s 8ms/step - accuracy: 0.8112 - loss: 0.4388 - val_accuracy: 0.7683 - val_loss
s: 0.5921 - learning_rate: 2.5000e-04
Epoch 122/600
143/148 ----- 0s 7ms/step - accuracy: 0.8108 - loss: 0.4437
Epoch 122: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
148/148 ----- 1s 7ms/step - accuracy: 0.8108 - loss: 0.4437 - val_accuracy: 0.7690 - val_loss
s: 0.5939 - learning_rate: 2.5000e-04
Epoch 123/600
148/148 ----- 1s 8ms/step - accuracy: 0.8123 - loss: 0.4377 - val_accuracy: 0.7680 - val_loss
s: 0.5940 - learning_rate: 1.2500e-04
Epoch 124/600
148/148 ----- 1s 7ms/step - accuracy: 0.8128 - loss: 0.4351 - val_accuracy: 0.7679 - val_loss
s: 0.5940 - learning_rate: 1.2500e-04
Epoch 125/600
148/148 ----- 1s 7ms/step - accuracy: 0.8109 - loss: 0.4397 - val_accuracy: 0.7683 - val_loss
```

```
s: 0.5946 - learning_rate: 1.2500e-04
Epoch 126/600
148/148 1s 8ms/step - accuracy: 0.8136 - loss: 0.4344 - val_accuracy: 0.7675 - val_loss
s: 0.5940 - learning_rate: 1.2500e-04
Epoch 127/600
148/148 1s 8ms/step - accuracy: 0.8111 - loss: 0.4364 - val_accuracy: 0.7687 - val_loss
s: 0.5941 - learning_rate: 1.2500e-04
Epoch 128/600
148/148 2s 12ms/step - accuracy: 0.8168 - loss: 0.4326 - val_accuracy: 0.7682 - val_loss
s: 0.5957 - learning_rate: 1.2500e-04
Epoch 129/600
148/148 2s 11ms/step - accuracy: 0.8149 - loss: 0.4353 - val_accuracy: 0.7688 - val_loss
s: 0.5943 - learning_rate: 1.2500e-04
Epoch 130/600
148/148 1s 9ms/step - accuracy: 0.8146 - loss: 0.4345 - val_accuracy: 0.7685 - val_loss
s: 0.5946 - learning_rate: 1.2500e-04
Epoch 131/600
148/148 1s 7ms/step - accuracy: 0.8144 - loss: 0.4337 - val_accuracy: 0.7694 - val_loss
s: 0.5949 - learning_rate: 1.2500e-04
Epoch 132/600
144/148 0s 7ms/step - accuracy: 0.8166 - loss: 0.4304
Epoch 132: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
148/148 1s 7ms/step - accuracy: 0.8165 - loss: 0.4306 - val_accuracy: 0.7678 - val_loss
s: 0.5924 - learning_rate: 1.2500e-04
Epoch 133/600
148/148 1s 8ms/step - accuracy: 0.8153 - loss: 0.4313 - val_accuracy: 0.7696 - val_loss
s: 0.5934 - learning_rate: 6.2500e-05
Epoch 134/600
148/148 1s 8ms/step - accuracy: 0.8150 - loss: 0.4357 - val_accuracy: 0.7688 - val_loss
s: 0.5942 - learning_rate: 6.2500e-05
Epoch 135/600
148/148 1s 8ms/step - accuracy: 0.8138 - loss: 0.4317 - val_accuracy: 0.7682 - val_loss
s: 0.5950 - learning_rate: 6.2500e-05
Epoch 136/600
148/148 1s 8ms/step - accuracy: 0.8162 - loss: 0.4299 - val_accuracy: 0.7681 - val_loss
s: 0.5950 - learning_rate: 6.2500e-05
Epoch 137/600
148/148 1s 8ms/step - accuracy: 0.8172 - loss: 0.4302 - val_accuracy: 0.7692 - val_loss
s: 0.5950 - learning_rate: 6.2500e-05
Epoch 138/600
148/148 1s 7ms/step - accuracy: 0.8142 - loss: 0.4308 - val_accuracy: 0.7698 - val_loss
s: 0.5961 - learning_rate: 6.2500e-05
Epoch 139/600
148/148 1s 8ms/step - accuracy: 0.8155 - loss: 0.4329 - val_accuracy: 0.7697 - val_loss
s: 0.5950 - learning_rate: 6.2500e-05
Epoch 140/600
148/148 1s 7ms/step - accuracy: 0.8160 - loss: 0.4313 - val_accuracy: 0.7697 - val_loss
s: 0.5957 - learning_rate: 6.2500e-05
Epoch 141/600
148/148 1s 7ms/step - accuracy: 0.8169 - loss: 0.4299 - val_accuracy: 0.7700 - val_loss
s: 0.5972 - learning_rate: 6.2500e-05
Epoch 142/600
142/148 0s 7ms/step - accuracy: 0.8156 - loss: 0.4292
Epoch 142: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
148/148 1s 7ms/step - accuracy: 0.8155 - loss: 0.4292 - val_accuracy: 0.7699 - val_loss
s: 0.5960 - learning_rate: 6.2500e-05
Epoch 143/600
148/148 1s 7ms/step - accuracy: 0.8147 - loss: 0.4324 - val_accuracy: 0.7697 - val_loss
s: 0.5963 - learning_rate: 3.1250e-05
Epoch 144/600
148/148 1s 9ms/step - accuracy: 0.8155 - loss: 0.4306 - val_accuracy: 0.7694 - val_loss
s: 0.5971 - learning_rate: 3.1250e-05
Epoch 145/600
148/148 1s 8ms/step - accuracy: 0.8176 - loss: 0.4281 - val_accuracy: 0.7691 - val_loss
s: 0.5967 - learning_rate: 3.1250e-05
Epoch 146/600
148/148 1s 8ms/step - accuracy: 0.8124 - loss: 0.4349 - val_accuracy: 0.7695 - val_loss
s: 0.5962 - learning_rate: 3.1250e-05
Epoch 147/600
148/148 1s 9ms/step - accuracy: 0.8188 - loss: 0.4274 - val_accuracy: 0.7694 - val_loss
s: 0.5964 - learning_rate: 3.1250e-05
Epoch 148/600
148/148 1s 8ms/step - accuracy: 0.8150 - loss: 0.4333 - val_accuracy: 0.7690 - val_loss
s: 0.5972 - learning_rate: 3.1250e-05
Epoch 149/600
148/148 1s 8ms/step - accuracy: 0.8166 - loss: 0.4273 - val_accuracy: 0.7690 - val_loss
```

```
s: 0.5966 - learning_rate: 3.1250e-05
Epoch 150/600
148/148 1s 8ms/step - accuracy: 0.8161 - loss: 0.4289 - val_accuracy: 0.7695 - val_loss
s: 0.5968 - learning_rate: 3.1250e-05
Epoch 151/600
148/148 1s 7ms/step - accuracy: 0.8169 - loss: 0.4314 - val_accuracy: 0.7695 - val_loss
s: 0.5964 - learning_rate: 3.1250e-05
Epoch 152/600
147/148 0s 7ms/step - accuracy: 0.8168 - loss: 0.4288
Epoch 152: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
148/148 1s 8ms/step - accuracy: 0.8168 - loss: 0.4288 - val_accuracy: 0.7692 - val_loss
s: 0.5962 - learning_rate: 3.1250e-05
Epoch 153/600
148/148 1s 8ms/step - accuracy: 0.8151 - loss: 0.4274 - val_accuracy: 0.7694 - val_loss
s: 0.5963 - learning_rate: 1.5625e-05
Epoch 154/600
148/148 1s 8ms/step - accuracy: 0.8167 - loss: 0.4284 - val_accuracy: 0.7696 - val_loss
s: 0.5963 - learning_rate: 1.5625e-05
Epoch 155/600
148/148 1s 7ms/step - accuracy: 0.8141 - loss: 0.4306 - val_accuracy: 0.7701 - val_loss
s: 0.5964 - learning_rate: 1.5625e-05
Epoch 156/600
148/148 1s 7ms/step - accuracy: 0.8179 - loss: 0.4281 - val_accuracy: 0.7699 - val_loss
s: 0.5964 - learning_rate: 1.5625e-05
Epoch 157/600
148/148 1s 8ms/step - accuracy: 0.8145 - loss: 0.4340 - val_accuracy: 0.7699 - val_loss
s: 0.5955 - learning_rate: 1.5625e-05
Epoch 158/600
148/148 1s 7ms/step - accuracy: 0.8170 - loss: 0.4241 - val_accuracy: 0.7698 - val_loss
s: 0.5961 - learning_rate: 1.5625e-05
Epoch 159/600
148/148 1s 8ms/step - accuracy: 0.8175 - loss: 0.4266 - val_accuracy: 0.7706 - val_loss
s: 0.5962 - learning_rate: 1.5625e-05
Epoch 160/600
148/148 1s 8ms/step - accuracy: 0.8185 - loss: 0.4255 - val_accuracy: 0.7707 - val_loss
s: 0.5960 - learning_rate: 1.5625e-05
Epoch 161/600
148/148 1s 8ms/step - accuracy: 0.8193 - loss: 0.4241 - val_accuracy: 0.7707 - val_loss
s: 0.5964 - learning_rate: 1.5625e-05
Epoch 162/600
147/148 0s 7ms/step - accuracy: 0.8162 - loss: 0.4285
Epoch 162: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.
148/148 1s 8ms/step - accuracy: 0.8162 - loss: 0.4285 - val_accuracy: 0.7709 - val_loss
s: 0.5962 - learning_rate: 1.5625e-05
Epoch 163/600
148/148 1s 9ms/step - accuracy: 0.8177 - loss: 0.4262 - val_accuracy: 0.7708 - val_loss
s: 0.5966 - learning_rate: 7.8125e-06
Epoch 164/600
148/148 1s 8ms/step - accuracy: 0.8153 - loss: 0.4305 - val_accuracy: 0.7706 - val_loss
s: 0.5966 - learning_rate: 7.8125e-06
Epoch 165/600
148/148 1s 8ms/step - accuracy: 0.8189 - loss: 0.4247 - val_accuracy: 0.7704 - val_loss
s: 0.5967 - learning_rate: 7.8125e-06
Epoch 166/600
148/148 1s 8ms/step - accuracy: 0.8191 - loss: 0.4219 - val_accuracy: 0.7706 - val_loss
s: 0.5969 - learning_rate: 7.8125e-06
Epoch 167/600
148/148 1s 7ms/step - accuracy: 0.8176 - loss: 0.4275 - val_accuracy: 0.7708 - val_loss
s: 0.5969 - learning_rate: 7.8125e-06
Epoch 168/600
148/148 1s 7ms/step - accuracy: 0.8163 - loss: 0.4299 - val_accuracy: 0.7706 - val_loss
s: 0.5966 - learning_rate: 7.8125e-06
Epoch 169/600
148/148 1s 7ms/step - accuracy: 0.8185 - loss: 0.4271 - val_accuracy: 0.7706 - val_loss
s: 0.5970 - learning_rate: 7.8125e-06
Epoch 170/600
148/148 1s 8ms/step - accuracy: 0.8161 - loss: 0.4307 - val_accuracy: 0.7706 - val_loss
s: 0.5967 - learning_rate: 7.8125e-06
Epoch 171/600
148/148 1s 8ms/step - accuracy: 0.8157 - loss: 0.4316 - val_accuracy: 0.7704 - val_loss
s: 0.5966 - learning_rate: 7.8125e-06
Epoch 172/600
145/148 0s 7ms/step - accuracy: 0.8190 - loss: 0.4247
Epoch 172: ReduceLROnPlateau reducing learning rate to 3.906250185536919e-06.
148/148 1s 7ms/step - accuracy: 0.8189 - loss: 0.4248 - val_accuracy: 0.7705 - val_loss
s: 0.5968 - learning_rate: 7.8125e-06
```

Epoch 173/600
148/148 1s 8ms/step - accuracy: 0.8186 - loss: 0.4247 - val_accuracy: 0.7700 - val_loss: 0.5967 - learning_rate: 3.9063e-06
Epoch 174/600
148/148 2s 11ms/step - accuracy: 0.8179 - loss: 0.4263 - val_accuracy: 0.7698 - val_loss: 0.5964 - learning_rate: 3.9063e-06
Epoch 175/600
148/148 1s 8ms/step - accuracy: 0.8168 - loss: 0.4243 - val_accuracy: 0.7698 - val_loss: 0.5963 - learning_rate: 3.9063e-06
Epoch 176/600
148/148 1s 8ms/step - accuracy: 0.8162 - loss: 0.4281 - val_accuracy: 0.7698 - val_loss: 0.5970 - learning_rate: 3.9063e-06
Epoch 177/600
148/148 1s 9ms/step - accuracy: 0.8156 - loss: 0.4307 - val_accuracy: 0.7702 - val_loss: 0.5974 - learning_rate: 3.9063e-06
Epoch 178/600
148/148 1s 8ms/step - accuracy: 0.8159 - loss: 0.4265 - val_accuracy: 0.7701 - val_loss: 0.5969 - learning_rate: 3.9063e-06
Epoch 179/600
148/148 1s 8ms/step - accuracy: 0.8184 - loss: 0.4281 - val_accuracy: 0.7701 - val_loss: 0.5970 - learning_rate: 3.9063e-06
Epoch 180/600
148/148 1s 8ms/step - accuracy: 0.8183 - loss: 0.4260 - val_accuracy: 0.7700 - val_loss: 0.5973 - learning_rate: 3.9063e-06
Epoch 181/600
148/148 1s 8ms/step - accuracy: 0.8154 - loss: 0.4301 - val_accuracy: 0.7705 - val_loss: 0.5970 - learning_rate: 3.9063e-06
Epoch 182/600
148/148 0s 7ms/step - accuracy: 0.8164 - loss: 0.4315
Epoch 182: ReduceLROnPlateau reducing learning rate to 1.9531250927684596e-06.
148/148 1s 8ms/step - accuracy: 0.8164 - loss: 0.4315 - val_accuracy: 0.7701 - val_loss: 0.5968 - learning_rate: 3.9063e-06
Epoch 183/600
148/148 1s 8ms/step - accuracy: 0.8155 - loss: 0.4312 - val_accuracy: 0.7701 - val_loss: 0.5971 - learning_rate: 1.9531e-06
Epoch 184/600
148/148 1s 8ms/step - accuracy: 0.8185 - loss: 0.4264 - val_accuracy: 0.7699 - val_loss: 0.5968 - learning_rate: 1.9531e-06
Epoch 185/600
148/148 1s 7ms/step - accuracy: 0.8186 - loss: 0.4266 - val_accuracy: 0.7699 - val_loss: 0.5968 - learning_rate: 1.9531e-06
Epoch 186/600
148/148 1s 7ms/step - accuracy: 0.8171 - loss: 0.4273 - val_accuracy: 0.7708 - val_loss: 0.5964 - learning_rate: 1.9531e-06
Epoch 187/600
148/148 1s 7ms/step - accuracy: 0.8169 - loss: 0.4288 - val_accuracy: 0.7701 - val_loss: 0.5968 - learning_rate: 1.9531e-06
Epoch 188/600
148/148 1s 8ms/step - accuracy: 0.8175 - loss: 0.4278 - val_accuracy: 0.7699 - val_loss: 0.5968 - learning_rate: 1.9531e-06
Epoch 189/600
148/148 1s 8ms/step - accuracy: 0.8175 - loss: 0.4261 - val_accuracy: 0.7702 - val_loss: 0.5967 - learning_rate: 1.9531e-06
Epoch 190/600
148/148 1s 8ms/step - accuracy: 0.8176 - loss: 0.4251 - val_accuracy: 0.7701 - val_loss: 0.5970 - learning_rate: 1.9531e-06
Epoch 191/600
148/148 1s 8ms/step - accuracy: 0.8167 - loss: 0.4271 - val_accuracy: 0.7703 - val_loss: 0.5966 - learning_rate: 1.9531e-06
Epoch 192/600
142/148 0s 7ms/step - accuracy: 0.8175 - loss: 0.4286
Epoch 192: ReduceLROnPlateau reducing learning rate to 1e-06.
148/148 1s 7ms/step - accuracy: 0.8175 - loss: 0.4286 - val_accuracy: 0.7702 - val_loss: 0.5968 - learning_rate: 1.9531e-06
Epoch 193/600
148/148 1s 7ms/step - accuracy: 0.8187 - loss: 0.4277 - val_accuracy: 0.7707 - val_loss: 0.5972 - learning_rate: 1.0000e-06
Epoch 194/600
148/148 1s 7ms/step - accuracy: 0.8166 - loss: 0.4290 - val_accuracy: 0.7706 - val_loss: 0.5969 - learning_rate: 1.0000e-06
Epoch 195/600
148/148 1s 7ms/step - accuracy: 0.8135 - loss: 0.4272 - val_accuracy: 0.7695 - val_loss: 0.5966 - learning_rate: 1.0000e-06
Epoch 196/600
148/148 1s 7ms/step - accuracy: 0.8166 - loss: 0.4296 - val_accuracy: 0.7698 - val_loss: 0.5971 - learning_rate: 1.0000e-06

Epoch 197/600
148/148 1s 7ms/step - accuracy: 0.8160 - loss: 0.4293 - val_accuracy: 0.7707 - val_loss: 0.5971 - learning_rate: 1.0000e-06
Epoch 198/600
148/148 1s 7ms/step - accuracy: 0.8171 - loss: 0.4298 - val_accuracy: 0.7699 - val_loss: 0.5974 - learning_rate: 1.0000e-06
Epoch 199/600
148/148 1s 7ms/step - accuracy: 0.8163 - loss: 0.4292 - val_accuracy: 0.7704 - val_loss: 0.5968 - learning_rate: 1.0000e-06
Epoch 200/600
148/148 1s 8ms/step - accuracy: 0.8148 - loss: 0.4294 - val_accuracy: 0.7705 - val_loss: 0.5967 - learning_rate: 1.0000e-06
Epoch 201/600
148/148 1s 7ms/step - accuracy: 0.8211 - loss: 0.4218 - val_accuracy: 0.7700 - val_loss: 0.5967 - learning_rate: 1.0000e-06
Epoch 202/600
148/148 1s 8ms/step - accuracy: 0.8160 - loss: 0.4314 - val_accuracy: 0.7699 - val_loss: 0.5971 - learning_rate: 1.0000e-06
Epoch 203/600
148/148 1s 8ms/step - accuracy: 0.8162 - loss: 0.4277 - val_accuracy: 0.7699 - val_loss: 0.5973 - learning_rate: 1.0000e-06
Epoch 204/600
148/148 1s 8ms/step - accuracy: 0.8183 - loss: 0.4260 - val_accuracy: 0.7705 - val_loss: 0.5966 - learning_rate: 1.0000e-06
Epoch 205/600
148/148 1s 7ms/step - accuracy: 0.8181 - loss: 0.4267 - val_accuracy: 0.7703 - val_loss: 0.5966 - learning_rate: 1.0000e-06
Epoch 206/600
148/148 1s 7ms/step - accuracy: 0.8189 - loss: 0.4251 - val_accuracy: 0.7707 - val_loss: 0.5968 - learning_rate: 1.0000e-06
Epoch 207/600
148/148 1s 7ms/step - accuracy: 0.8176 - loss: 0.4278 - val_accuracy: 0.7700 - val_loss: 0.5968 - learning_rate: 1.0000e-06
Epoch 208/600
148/148 1s 7ms/step - accuracy: 0.8174 - loss: 0.4255 - val_accuracy: 0.7698 - val_loss: 0.5970 - learning_rate: 1.0000e-06
Epoch 209/600
148/148 1s 7ms/step - accuracy: 0.8154 - loss: 0.4327 - val_accuracy: 0.7701 - val_loss: 0.5971 - learning_rate: 1.0000e-06
Epoch 210/600
148/148 1s 7ms/step - accuracy: 0.8160 - loss: 0.4302 - val_accuracy: 0.7700 - val_loss: 0.5971 - learning_rate: 1.0000e-06
Epoch 211/600
148/148 1s 7ms/step - accuracy: 0.8167 - loss: 0.4259 - val_accuracy: 0.7700 - val_loss: 0.5969 - learning_rate: 1.0000e-06
Epoch 212/600
148/148 1s 7ms/step - accuracy: 0.8143 - loss: 0.4313 - val_accuracy: 0.7700 - val_loss: 0.5970 - learning_rate: 1.0000e-06
Epoch 213/600
148/148 1s 7ms/step - accuracy: 0.8173 - loss: 0.4278 - val_accuracy: 0.7701 - val_loss: 0.5971 - learning_rate: 1.0000e-06
Epoch 214/600
148/148 1s 7ms/step - accuracy: 0.8177 - loss: 0.4277 - val_accuracy: 0.7700 - val_loss: 0.5968 - learning_rate: 1.0000e-06
Epoch 215/600
148/148 1s 7ms/step - accuracy: 0.8143 - loss: 0.4308 - val_accuracy: 0.7697 - val_loss: 0.5967 - learning_rate: 1.0000e-06
Epoch 216/600
148/148 1s 7ms/step - accuracy: 0.8186 - loss: 0.4244 - val_accuracy: 0.7701 - val_loss: 0.5971 - learning_rate: 1.0000e-06
Epoch 217/600
148/148 1s 8ms/step - accuracy: 0.8172 - loss: 0.4255 - val_accuracy: 0.7701 - val_loss: 0.5969 - learning_rate: 1.0000e-06
Epoch 218/600
148/148 1s 8ms/step - accuracy: 0.8189 - loss: 0.4294 - val_accuracy: 0.7702 - val_loss: 0.5972 - learning_rate: 1.0000e-06
Epoch 219/600
148/148 1s 7ms/step - accuracy: 0.8194 - loss: 0.4265 - val_accuracy: 0.7697 - val_loss: 0.5968 - learning_rate: 1.0000e-06
Epoch 220/600
148/148 1s 8ms/step - accuracy: 0.8203 - loss: 0.4252 - val_accuracy: 0.7699 - val_loss: 0.5965 - learning_rate: 1.0000e-06
Epoch 221/600
148/148 1s 7ms/step - accuracy: 0.8180 - loss: 0.4256 - val_accuracy: 0.7700 - val_loss: 0.5969 - learning_rate: 1.0000e-06
Epoch 222/600

```
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.8172 - loss: 0.4294 - val_accuracy: 0.7701 - val_loss: 0.5971 - learning_rate: 1.0000e-06
```

```
In [49]: ann7.evaluate(X_train, y_train)
```

```
2363/2363 ━━━━━━━━ 2s 899us/step - accuracy: 0.8789 - loss: 0.2887
```

```
Out[49]: [0.28743046522140503, 0.8801348805427551]
```

```
In [50]: ann7.summary()
```

Model: "sequential_4"

Layer (type)	Output Shape	Param #
dense_28 (Dense)	(None, 512)	23,040
batch_normalization_24 (BatchNormalization)	(None, 512)	2,048
dropout_24 (Dropout)	(None, 512)	0
dense_29 (Dense)	(None, 256)	131,328
batch_normalization_25 (BatchNormalization)	(None, 256)	1,024
dropout_25 (Dropout)	(None, 256)	0
dense_30 (Dense)	(None, 128)	32,896
batch_normalization_26 (BatchNormalization)	(None, 128)	512
dropout_26 (Dropout)	(None, 128)	0
dense_31 (Dense)	(None, 64)	8,256
batch_normalization_27 (BatchNormalization)	(None, 64)	256
dropout_27 (Dropout)	(None, 64)	0
dense_32 (Dense)	(None, 3)	195

Total params: 594,827 (2.27 MB)

Trainable params: 197,635 (772.01 KB)

Non-trainable params: 1,920 (7.50 KB)

Optimizer params: 395,272 (1.51 MB)

```
In [51]: eval_metric(ann7, X_train, y_train, X_val, y_val)
```

2363/2363 ————— 3s 1ms/step

591/591 ————— 1s 1ms/step

Test Set:

```
[[4453 889 166]
 [1352 7617 1084]
 [ 74 766 2502]]
```

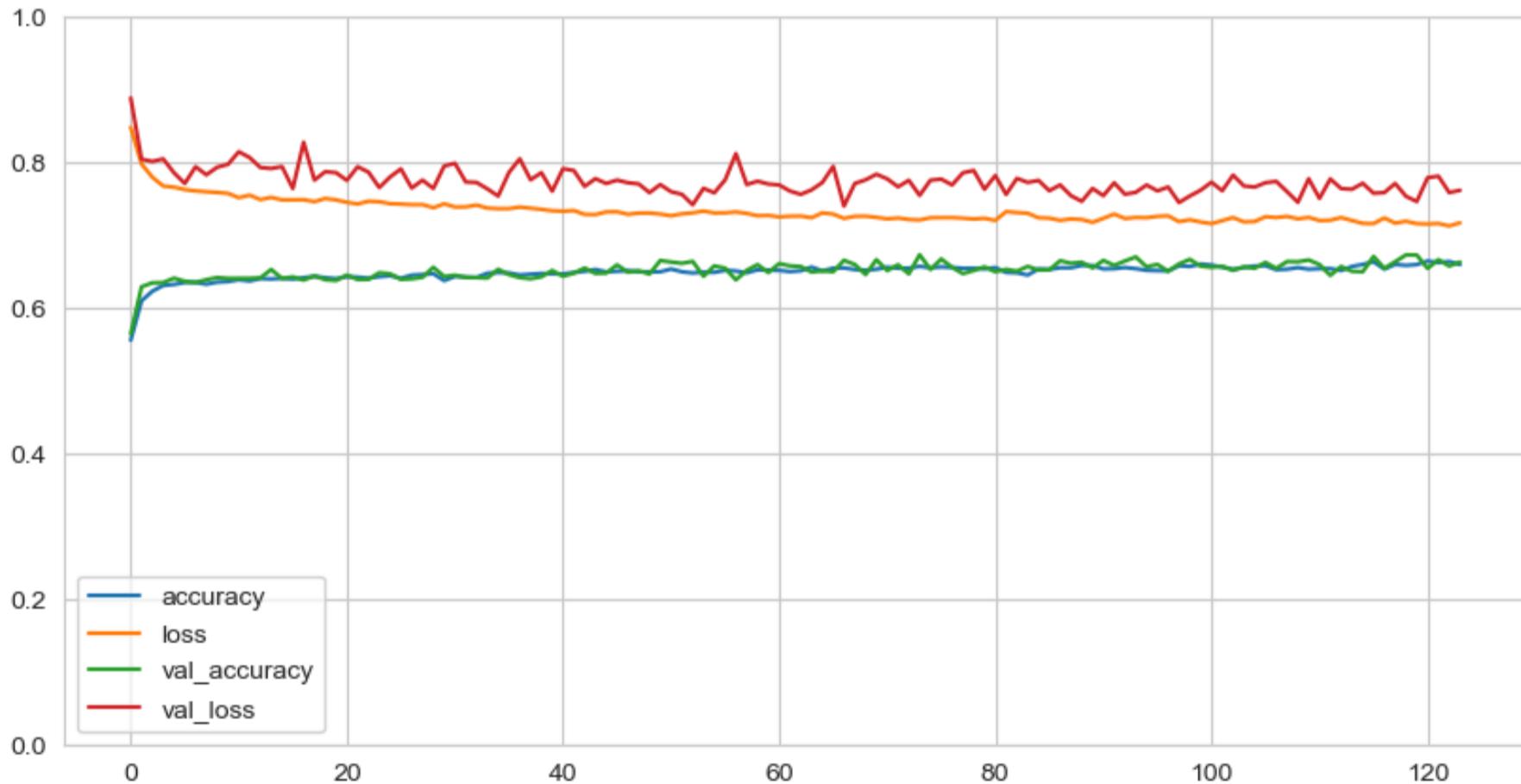
	precision	recall	f1-score	support
0	0.76	0.81	0.78	5508
1	0.82	0.76	0.79	10053
2	0.67	0.75	0.71	3342
accuracy			0.77	18903
macro avg	0.75	0.77	0.76	18903
weighted avg	0.78	0.77	0.77	18903

Train Set:

```
[[20387 1455 189]
 [ 3255 34133 2821]
 [ 42 1301 12027]]
```

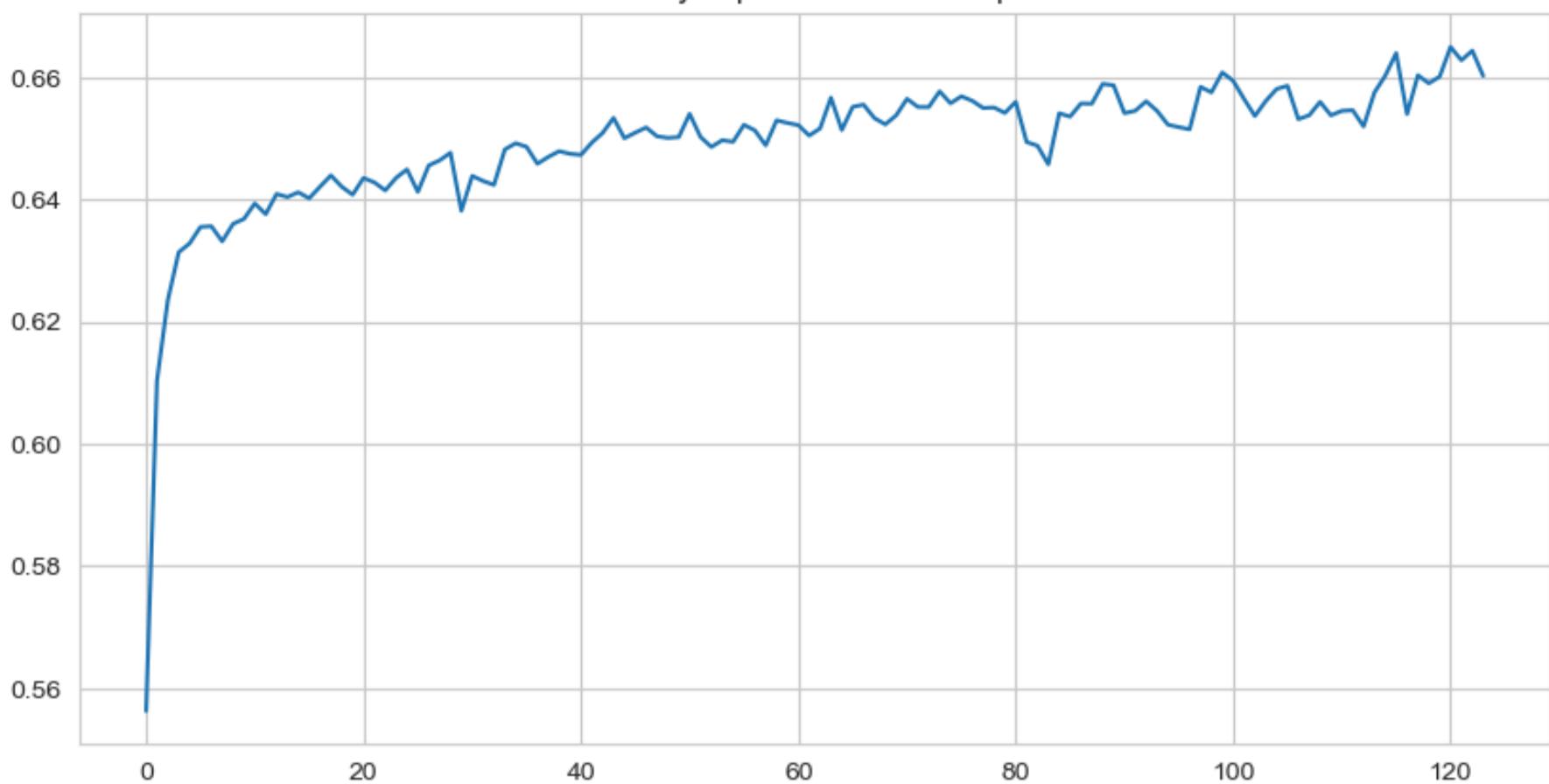
	precision	recall	f1-score	support
0	0.86	0.93	0.89	22031
1	0.93	0.85	0.89	40209
2	0.80	0.90	0.85	13370
accuracy			0.88	75610
macro avg	0.86	0.89	0.87	75610
weighted avg	0.88	0.88	0.88	75610

```
In [52]: pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



```
In [53]: pd.DataFrame(history.history)[“accuracy”].plot(figsize=(10, 5))
plt.title(“Accuracy improvements with Epoch”)
plt.show()
```

Accuracy improvements with Epoch



In [230...]

```
# Save the Model

from tensorflow.keras.models import load_model

# Save the model to 'model.h5' file
ann7.save('ann7_model.h5')

# To Load the model later for further use
loaded_ann7 = load_model('ann7_model.h5')
```

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)` . This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')` .
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.

ANN-4 Model Summary:

- **(Dense) layers: 7 / Neurons:** 512-256-256-128-128-64 / **Dropout:** 30-30-25-25-20-20% / **Learning Rate:** 0.001 / **Batch Size:** 512 / **Epochs:** 600 / **Early Stop (val_accuracy):** 60
- **Accuracy:** 0.9054 / **Val_Accuracy:** 0.7700 / **Loss:** 0.2586 / **Val_Loss:** 0.7300
- **Train Recall (Class 2):** 0.95 / **Test Recall (Class 2):** 0.77

Improvements:

- The model achieves high training accuracy (90.54%) and excellent recall for Class 2 (95% training, 77% testing), showing effective recognition of the minority class.

No Improvement:

- Validation accuracy remains relatively stable but does not show significant improvement, highlighting the challenge of generalizing the complex model to unseen data.

Got Worse:

- Validation loss indicates a potential overfitting issue, as it has increased to 0.7300 despite the improved performance metrics in training. This suggests that while the model is learning the training data well, it struggles to perform similarly on validation data.

ANN-7 Model with SMOTE (%91)

```
In [27]: print('Credi_Score Classes before Smote: ', y_train.value_counts())
print('\nCredi_Score Classes after Smote: ', y_train_smote.value_counts())
```

```
Credi_Score Classes before Smote: Credit_Score
1    40209
0    22031
2   13370
Name: count, dtype: int64
```

```
Credi_Score Classes after Smote: Credit_Score
1    40209
2    40209
0    40209
Name: count, dtype: int64
```

```
In [54]: # ANN-7 Model with SMOTE:
# Added ReduceLROnPlateau to callbacks

# 1) Optimized Model Architecture:
ann7_smote = Sequential([
    Dense(512, input_dim=X_train_smote.shape[1], activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(256, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.25),
    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.2),
    Dense(3, activation='softmax')
])

# 2) Compiling the Model:
ann7_smote.compile(optimizer=Adam(learning_rate=0.001),
                    loss='sparse_categorical_crossentropy',
                    metrics=['accuracy'])

# 3) Early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                                patience=60,
                                mode="auto",
                                restore_best_weights=True)

# 4) ReduceLROnPlateau callback
reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                               factor=0.5, # Learning rate reduced by half if no improvement
                               patience=10, # Wait 10 epochs before reducing Learning rate
                               min_lr=1e-6, # Minimum Learning rate set
                               verbose=1)

# 5) Train the model
history = ann7_smote.fit(
    x=X_train_smote,
    y=y_train_smote,
    validation_data=(X_val, y_val),
    batch_size=512,
    epochs=650,
    verbose=1,
    callbacks=[early_stopping, reduce_lr]) # Added ReduceLROnPlateau to callbacks
```

Epoch 1/650
236/236 6s 11ms/step - accuracy: 0.8167 - loss: 0.4561 - val_accuracy: 0.7404 - val_loss: 0.6682 - learning_rate: 0.0010
Epoch 2/650
236/236 2s 8ms/step - accuracy: 0.8246 - loss: 0.4370 - val_accuracy: 0.7442 - val_loss: 0.6556 - learning_rate: 0.0010
Epoch 3/650
236/236 2s 8ms/step - accuracy: 0.8262 - loss: 0.4338 - val_accuracy: 0.7468 - val_loss: 0.6504 - learning_rate: 0.0010
Epoch 4/650
236/236 2s 8ms/step - accuracy: 0.8266 - loss: 0.4291 - val_accuracy: 0.7491 - val_loss: 0.6470 - learning_rate: 0.0010
Epoch 5/650
236/236 2s 8ms/step - accuracy: 0.8294 - loss: 0.4260 - val_accuracy: 0.7485 - val_loss: 0.6466 - learning_rate: 0.0010
Epoch 6/650
236/236 2s 7ms/step - accuracy: 0.8297 - loss: 0.4259 - val_accuracy: 0.7479 - val_loss: 0.6457 - learning_rate: 0.0010
Epoch 7/650
236/236 2s 8ms/step - accuracy: 0.8286 - loss: 0.4277 - val_accuracy: 0.7433 - val_loss: 0.6569 - learning_rate: 0.0010
Epoch 8/650
236/236 2s 8ms/step - accuracy: 0.8297 - loss: 0.4301 - val_accuracy: 0.7429 - val_loss: 0.6677 - learning_rate: 0.0010
Epoch 9/650
236/236 2s 8ms/step - accuracy: 0.8307 - loss: 0.4232 - val_accuracy: 0.7526 - val_loss: 0.6425 - learning_rate: 0.0010
Epoch 10/650
236/236 2s 7ms/step - accuracy: 0.8319 - loss: 0.4205 - val_accuracy: 0.7525 - val_loss: 0.6373 - learning_rate: 0.0010
Epoch 11/650
236/236 2s 8ms/step - accuracy: 0.8302 - loss: 0.4243 - val_accuracy: 0.7453 - val_loss: 0.6575 - learning_rate: 0.0010
Epoch 12/650
236/236 2s 8ms/step - accuracy: 0.8304 - loss: 0.4232 - val_accuracy: 0.7462 - val_loss: 0.6494 - learning_rate: 0.0010
Epoch 13/650
236/236 2s 8ms/step - accuracy: 0.8329 - loss: 0.4230 - val_accuracy: 0.7424 - val_loss: 0.6648 - learning_rate: 0.0010
Epoch 14/650
236/236 2s 8ms/step - accuracy: 0.8326 - loss: 0.4201 - val_accuracy: 0.7462 - val_loss: 0.6554 - learning_rate: 0.0010
Epoch 15/650
236/236 2s 8ms/step - accuracy: 0.8329 - loss: 0.4195 - val_accuracy: 0.7471 - val_loss: 0.6537 - learning_rate: 0.0010
Epoch 16/650
236/236 2s 7ms/step - accuracy: 0.8347 - loss: 0.4168 - val_accuracy: 0.7488 - val_loss: 0.6505 - learning_rate: 0.0010
Epoch 17/650
236/236 2s 7ms/step - accuracy: 0.8348 - loss: 0.4158 - val_accuracy: 0.7524 - val_loss: 0.6404 - learning_rate: 0.0010
Epoch 18/650
236/236 2s 7ms/step - accuracy: 0.8361 - loss: 0.4124 - val_accuracy: 0.7484 - val_loss: 0.6534 - learning_rate: 0.0010
Epoch 19/650
236/236 2s 7ms/step - accuracy: 0.8354 - loss: 0.4142 - val_accuracy: 0.7488 - val_loss: 0.6542 - learning_rate: 0.0010
Epoch 20/650
234/236 0s 7ms/step - accuracy: 0.8357 - loss: 0.4145
Epoch 20: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
236/236 2s 8ms/step - accuracy: 0.8357 - loss: 0.4146 - val_accuracy: 0.7494 - val_loss: 0.6462 - learning_rate: 0.0010
Epoch 21/650
236/236 2s 9ms/step - accuracy: 0.8387 - loss: 0.4056 - val_accuracy: 0.7514 - val_loss: 0.6531 - learning_rate: 5.0000e-04
Epoch 22/650
236/236 2s 7ms/step - accuracy: 0.8419 - loss: 0.3981 - val_accuracy: 0.7537 - val_loss: 0.6557 - learning_rate: 5.0000e-04
Epoch 23/650
236/236 2s 8ms/step - accuracy: 0.8461 - loss: 0.3898 - val_accuracy: 0.7522 - val_loss: 0.6568 - learning_rate: 5.0000e-04
Epoch 24/650
236/236 2s 8ms/step - accuracy: 0.8457 - loss: 0.3923 - val_accuracy: 0.7518 - val_loss: 0.6551 - learning_rate: 5.0000e-04
Epoch 25/650
236/236 2s 8ms/step - accuracy: 0.8443 - loss: 0.3919 - val_accuracy: 0.7561 - val_loss:

```
s: 0.6434 - learning_rate: 5.0000e-04
Epoch 26/650
236/236 2s 8ms/step - accuracy: 0.8446 - loss: 0.3930 - val_accuracy: 0.7549 - val_loss
s: 0.6510 - learning_rate: 5.0000e-04
Epoch 27/650
236/236 2s 7ms/step - accuracy: 0.8459 - loss: 0.3909 - val_accuracy: 0.7528 - val_loss
s: 0.6449 - learning_rate: 5.0000e-04
Epoch 28/650
236/236 2s 8ms/step - accuracy: 0.8466 - loss: 0.3888 - val_accuracy: 0.7524 - val_loss
s: 0.6555 - learning_rate: 5.0000e-04
Epoch 29/650
236/236 2s 8ms/step - accuracy: 0.8461 - loss: 0.3876 - val_accuracy: 0.7550 - val_loss
s: 0.6542 - learning_rate: 5.0000e-04
Epoch 30/650
236/236 0s 8ms/step - accuracy: 0.8484 - loss: 0.3867
Epoch 30: ReduceLROnPlateau reducing learning rate to 0.000250000118743628.
236/236 2s 8ms/step - accuracy: 0.8484 - loss: 0.3867 - val_accuracy: 0.7543 - val_loss
s: 0.6451 - learning_rate: 5.0000e-04
Epoch 31/650
236/236 2s 8ms/step - accuracy: 0.8480 - loss: 0.3868 - val_accuracy: 0.7551 - val_loss
s: 0.6524 - learning_rate: 2.5000e-04
Epoch 32/650
236/236 2s 8ms/step - accuracy: 0.8504 - loss: 0.3793 - val_accuracy: 0.7559 - val_loss
s: 0.6581 - learning_rate: 2.5000e-04
Epoch 33/650
236/236 2s 8ms/step - accuracy: 0.8511 - loss: 0.3787 - val_accuracy: 0.7573 - val_loss
s: 0.6515 - learning_rate: 2.5000e-04
Epoch 34/650
236/236 2s 7ms/step - accuracy: 0.8494 - loss: 0.3815 - val_accuracy: 0.7549 - val_loss
s: 0.6531 - learning_rate: 2.5000e-04
Epoch 35/650
236/236 2s 7ms/step - accuracy: 0.8505 - loss: 0.3766 - val_accuracy: 0.7564 - val_loss
s: 0.6554 - learning_rate: 2.5000e-04
Epoch 36/650
236/236 2s 7ms/step - accuracy: 0.8508 - loss: 0.3756 - val_accuracy: 0.7567 - val_loss
s: 0.6543 - learning_rate: 2.5000e-04
Epoch 37/650
236/236 2s 8ms/step - accuracy: 0.8531 - loss: 0.3710 - val_accuracy: 0.7572 - val_loss
s: 0.6587 - learning_rate: 2.5000e-04
Epoch 38/650
236/236 2s 8ms/step - accuracy: 0.8532 - loss: 0.3746 - val_accuracy: 0.7565 - val_loss
s: 0.6534 - learning_rate: 2.5000e-04
Epoch 39/650
236/236 2s 8ms/step - accuracy: 0.8535 - loss: 0.3721 - val_accuracy: 0.7560 - val_loss
s: 0.6542 - learning_rate: 2.5000e-04
Epoch 40/650
235/236 0s 7ms/step - accuracy: 0.8516 - loss: 0.3768
Epoch 40: ReduceLROnPlateau reducing learning rate to 0.000125000059371814.
236/236 2s 8ms/step - accuracy: 0.8516 - loss: 0.3768 - val_accuracy: 0.7558 - val_loss
s: 0.6604 - learning_rate: 2.5000e-04
Epoch 41/650
236/236 2s 7ms/step - accuracy: 0.8557 - loss: 0.3679 - val_accuracy: 0.7569 - val_loss
s: 0.6544 - learning_rate: 1.2500e-04
Epoch 42/650
236/236 2s 8ms/step - accuracy: 0.8576 - loss: 0.3642 - val_accuracy: 0.7569 - val_loss
s: 0.6607 - learning_rate: 1.2500e-04
Epoch 43/650
236/236 2s 7ms/step - accuracy: 0.8539 - loss: 0.3701 - val_accuracy: 0.7574 - val_loss
s: 0.6598 - learning_rate: 1.2500e-04
Epoch 44/650
236/236 2s 7ms/step - accuracy: 0.8579 - loss: 0.3648 - val_accuracy: 0.7557 - val_loss
s: 0.6598 - learning_rate: 1.2500e-04
Epoch 45/650
236/236 2s 8ms/step - accuracy: 0.8545 - loss: 0.3694 - val_accuracy: 0.7569 - val_loss
s: 0.6592 - learning_rate: 1.2500e-04
Epoch 46/650
236/236 2s 7ms/step - accuracy: 0.8531 - loss: 0.3719 - val_accuracy: 0.7564 - val_loss
s: 0.6568 - learning_rate: 1.2500e-04
Epoch 47/650
236/236 2s 8ms/step - accuracy: 0.8548 - loss: 0.3688 - val_accuracy: 0.7566 - val_loss
s: 0.6585 - learning_rate: 1.2500e-04
Epoch 48/650
236/236 2s 8ms/step - accuracy: 0.8566 - loss: 0.3643 - val_accuracy: 0.7570 - val_loss
s: 0.6601 - learning_rate: 1.2500e-04
Epoch 49/650
236/236 2s 8ms/step - accuracy: 0.8578 - loss: 0.3655 - val_accuracy: 0.7579 - val_loss
```

```
s: 0.6563 - learning_rate: 1.2500e-04
Epoch 50/650
229/236 0s 8ms/step - accuracy: 0.8578 - loss: 0.3630
Epoch 50: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
236/236 2s 9ms/step - accuracy: 0.8578 - loss: 0.3631 - val_accuracy: 0.7587 - val_loss
s: 0.6569 - learning_rate: 1.2500e-04
Epoch 51/650
236/236 2s 8ms/step - accuracy: 0.8579 - loss: 0.3657 - val_accuracy: 0.7586 - val_loss
s: 0.6579 - learning_rate: 6.2500e-05
Epoch 52/650
236/236 2s 8ms/step - accuracy: 0.8585 - loss: 0.3618 - val_accuracy: 0.7589 - val_loss
s: 0.6603 - learning_rate: 6.2500e-05
Epoch 53/650
236/236 2s 7ms/step - accuracy: 0.8582 - loss: 0.3634 - val_accuracy: 0.7590 - val_loss
s: 0.6613 - learning_rate: 6.2500e-05
Epoch 54/650
236/236 2s 8ms/step - accuracy: 0.8583 - loss: 0.3644 - val_accuracy: 0.7590 - val_loss
s: 0.6595 - learning_rate: 6.2500e-05
Epoch 55/650
236/236 2s 9ms/step - accuracy: 0.8576 - loss: 0.3622 - val_accuracy: 0.7589 - val_loss
s: 0.6615 - learning_rate: 6.2500e-05
Epoch 56/650
236/236 2s 8ms/step - accuracy: 0.8587 - loss: 0.3622 - val_accuracy: 0.7586 - val_loss
s: 0.6618 - learning_rate: 6.2500e-05
Epoch 57/650
236/236 2s 8ms/step - accuracy: 0.8573 - loss: 0.3643 - val_accuracy: 0.7590 - val_loss
s: 0.6603 - learning_rate: 6.2500e-05
Epoch 58/650
236/236 2s 7ms/step - accuracy: 0.8567 - loss: 0.3614 - val_accuracy: 0.7593 - val_loss
s: 0.6588 - learning_rate: 6.2500e-05
Epoch 59/650
236/236 2s 7ms/step - accuracy: 0.8567 - loss: 0.3637 - val_accuracy: 0.7588 - val_loss
s: 0.6583 - learning_rate: 6.2500e-05
Epoch 60/650
230/236 0s 7ms/step - accuracy: 0.8575 - loss: 0.3619
Epoch 60: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
236/236 2s 8ms/step - accuracy: 0.8575 - loss: 0.3620 - val_accuracy: 0.7591 - val_loss
s: 0.6596 - learning_rate: 6.2500e-05
Epoch 61/650
236/236 2s 8ms/step - accuracy: 0.8548 - loss: 0.3666 - val_accuracy: 0.7592 - val_loss
s: 0.6632 - learning_rate: 3.1250e-05
Epoch 62/650
236/236 2s 8ms/step - accuracy: 0.8595 - loss: 0.3603 - val_accuracy: 0.7597 - val_loss
s: 0.6628 - learning_rate: 3.1250e-05
Epoch 63/650
236/236 2s 8ms/step - accuracy: 0.8581 - loss: 0.3598 - val_accuracy: 0.7594 - val_loss
s: 0.6606 - learning_rate: 3.1250e-05
Epoch 64/650
236/236 2s 8ms/step - accuracy: 0.8585 - loss: 0.3626 - val_accuracy: 0.7594 - val_loss
s: 0.6613 - learning_rate: 3.1250e-05
Epoch 65/650
236/236 2s 8ms/step - accuracy: 0.8597 - loss: 0.3587 - val_accuracy: 0.7599 - val_loss
s: 0.6612 - learning_rate: 3.1250e-05
Epoch 66/650
236/236 2s 8ms/step - accuracy: 0.8584 - loss: 0.3614 - val_accuracy: 0.7596 - val_loss
s: 0.6609 - learning_rate: 3.1250e-05
Epoch 67/650
236/236 2s 8ms/step - accuracy: 0.8575 - loss: 0.3624 - val_accuracy: 0.7598 - val_loss
s: 0.6619 - learning_rate: 3.1250e-05
Epoch 68/650
236/236 2s 7ms/step - accuracy: 0.8583 - loss: 0.3631 - val_accuracy: 0.7592 - val_loss
s: 0.6612 - learning_rate: 3.1250e-05
Epoch 69/650
236/236 2s 8ms/step - accuracy: 0.8581 - loss: 0.3629 - val_accuracy: 0.7605 - val_loss
s: 0.6601 - learning_rate: 3.1250e-05
Epoch 70/650
234/236 0s 8ms/step - accuracy: 0.8598 - loss: 0.3581
Epoch 70: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
236/236 2s 8ms/step - accuracy: 0.8598 - loss: 0.3582 - val_accuracy: 0.7598 - val_loss
s: 0.6604 - learning_rate: 3.1250e-05
Epoch 71/650
236/236 2s 9ms/step - accuracy: 0.8603 - loss: 0.3570 - val_accuracy: 0.7592 - val_loss
s: 0.6610 - learning_rate: 1.5625e-05
Epoch 72/650
236/236 2s 8ms/step - accuracy: 0.8589 - loss: 0.3601 - val_accuracy: 0.7589 - val_loss
s: 0.6622 - learning_rate: 1.5625e-05
```

Epoch 73/650
236/236 2s 8ms/step - accuracy: 0.8587 - loss: 0.3594 - val_accuracy: 0.7602 - val_loss: 0.6607 - learning_rate: 1.5625e-05
Epoch 74/650
236/236 2s 7ms/step - accuracy: 0.8605 - loss: 0.3575 - val_accuracy: 0.7600 - val_loss: 0.6617 - learning_rate: 1.5625e-05
Epoch 75/650
236/236 2s 8ms/step - accuracy: 0.8620 - loss: 0.3557 - val_accuracy: 0.7600 - val_loss: 0.6613 - learning_rate: 1.5625e-05
Epoch 76/650
236/236 2s 8ms/step - accuracy: 0.8566 - loss: 0.3613 - val_accuracy: 0.7604 - val_loss: 0.6616 - learning_rate: 1.5625e-05
Epoch 77/650
236/236 2s 8ms/step - accuracy: 0.8602 - loss: 0.3562 - val_accuracy: 0.7602 - val_loss: 0.6619 - learning_rate: 1.5625e-05
Epoch 78/650
236/236 2s 7ms/step - accuracy: 0.8617 - loss: 0.3563 - val_accuracy: 0.7604 - val_loss: 0.6618 - learning_rate: 1.5625e-05
Epoch 79/650
236/236 2s 7ms/step - accuracy: 0.8600 - loss: 0.3586 - val_accuracy: 0.7599 - val_loss: 0.6622 - learning_rate: 1.5625e-05
Epoch 80/650
231/236 0s 7ms/step - accuracy: 0.8606 - loss: 0.3573
Epoch 80: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.
236/236 2s 8ms/step - accuracy: 0.8606 - loss: 0.3573 - val_accuracy: 0.7603 - val_loss: 0.6627 - learning_rate: 1.5625e-05
Epoch 81/650
236/236 2s 8ms/step - accuracy: 0.8592 - loss: 0.3604 - val_accuracy: 0.7602 - val_loss: 0.6619 - learning_rate: 7.8125e-06
Epoch 82/650
236/236 2s 7ms/step - accuracy: 0.8592 - loss: 0.3605 - val_accuracy: 0.7603 - val_loss: 0.6617 - learning_rate: 7.8125e-06
Epoch 83/650
236/236 2s 7ms/step - accuracy: 0.8592 - loss: 0.3584 - val_accuracy: 0.7602 - val_loss: 0.6620 - learning_rate: 7.8125e-06
Epoch 84/650
236/236 2s 7ms/step - accuracy: 0.8600 - loss: 0.3588 - val_accuracy: 0.7600 - val_loss: 0.6615 - learning_rate: 7.8125e-06
Epoch 85/650
236/236 2s 7ms/step - accuracy: 0.8582 - loss: 0.3603 - val_accuracy: 0.7597 - val_loss: 0.6624 - learning_rate: 7.8125e-06
Epoch 86/650
236/236 2s 7ms/step - accuracy: 0.8583 - loss: 0.3632 - val_accuracy: 0.7602 - val_loss: 0.6621 - learning_rate: 7.8125e-06
Epoch 87/650
236/236 2s 7ms/step - accuracy: 0.8603 - loss: 0.3595 - val_accuracy: 0.7597 - val_loss: 0.6625 - learning_rate: 7.8125e-06
Epoch 88/650
236/236 2s 7ms/step - accuracy: 0.8607 - loss: 0.3577 - val_accuracy: 0.7596 - val_loss: 0.6622 - learning_rate: 7.8125e-06
Epoch 89/650
236/236 2s 7ms/step - accuracy: 0.8597 - loss: 0.3557 - val_accuracy: 0.7602 - val_loss: 0.6614 - learning_rate: 7.8125e-06
Epoch 90/650
232/236 0s 7ms/step - accuracy: 0.8583 - loss: 0.3640
Epoch 90: ReduceLROnPlateau reducing learning rate to 3.906250185536919e-06.
236/236 2s 8ms/step - accuracy: 0.8583 - loss: 0.3640 - val_accuracy: 0.7600 - val_loss: 0.6617 - learning_rate: 7.8125e-06
Epoch 91/650
236/236 2s 7ms/step - accuracy: 0.8605 - loss: 0.3574 - val_accuracy: 0.7600 - val_loss: 0.6617 - learning_rate: 3.9063e-06
Epoch 92/650
236/236 2s 7ms/step - accuracy: 0.8596 - loss: 0.3625 - val_accuracy: 0.7601 - val_loss: 0.6614 - learning_rate: 3.9063e-06
Epoch 93/650
236/236 2s 7ms/step - accuracy: 0.8602 - loss: 0.3582 - val_accuracy: 0.7597 - val_loss: 0.6614 - learning_rate: 3.9063e-06
Epoch 94/650
236/236 2s 8ms/step - accuracy: 0.8578 - loss: 0.3640 - val_accuracy: 0.7592 - val_loss: 0.6629 - learning_rate: 3.9063e-06
Epoch 95/650
236/236 2s 7ms/step - accuracy: 0.8614 - loss: 0.3562 - val_accuracy: 0.7597 - val_loss: 0.6624 - learning_rate: 3.9063e-06
Epoch 96/650
236/236 2s 7ms/step - accuracy: 0.8599 - loss: 0.3584 - val_accuracy: 0.7600 - val_loss: 0.6619 - learning_rate: 3.9063e-06

```
Epoch 97/650
236/236 2s 7ms/step - accuracy: 0.8597 - loss: 0.3584 - val_accuracy: 0.7600 - val_loss: 0.6616 - learning_rate: 3.9063e-06
Epoch 98/650
236/236 2s 7ms/step - accuracy: 0.8590 - loss: 0.3600 - val_accuracy: 0.7600 - val_loss: 0.6622 - learning_rate: 3.9063e-06
Epoch 99/650
236/236 2s 8ms/step - accuracy: 0.8606 - loss: 0.3587 - val_accuracy: 0.7598 - val_loss: 0.6621 - learning_rate: 3.9063e-06
Epoch 100/650
235/236 0s 7ms/step - accuracy: 0.8576 - loss: 0.3599
Epoch 100: ReduceLROnPlateau reducing learning rate to 1.9531250927684596e-06.
236/236 2s 7ms/step - accuracy: 0.8576 - loss: 0.3598 - val_accuracy: 0.7599 - val_loss: 0.6620 - learning_rate: 3.9063e-06
Epoch 101/650
236/236 2s 7ms/step - accuracy: 0.8609 - loss: 0.3583 - val_accuracy: 0.7599 - val_loss: 0.6623 - learning_rate: 1.9531e-06
Epoch 102/650
236/236 2s 7ms/step - accuracy: 0.8590 - loss: 0.3589 - val_accuracy: 0.7600 - val_loss: 0.6621 - learning_rate: 1.9531e-06
Epoch 103/650
236/236 2s 7ms/step - accuracy: 0.8585 - loss: 0.3622 - val_accuracy: 0.7601 - val_loss: 0.6617 - learning_rate: 1.9531e-06
Epoch 104/650
236/236 2s 8ms/step - accuracy: 0.8605 - loss: 0.3585 - val_accuracy: 0.7596 - val_loss: 0.6616 - learning_rate: 1.9531e-06
Epoch 105/650
236/236 2s 7ms/step - accuracy: 0.8589 - loss: 0.3606 - val_accuracy: 0.7601 - val_loss: 0.6617 - learning_rate: 1.9531e-06
Epoch 106/650
236/236 2s 7ms/step - accuracy: 0.8580 - loss: 0.3634 - val_accuracy: 0.7599 - val_loss: 0.6624 - learning_rate: 1.9531e-06
Epoch 107/650
236/236 2s 7ms/step - accuracy: 0.8603 - loss: 0.3582 - val_accuracy: 0.7600 - val_loss: 0.6619 - learning_rate: 1.9531e-06
Epoch 108/650
236/236 2s 8ms/step - accuracy: 0.8581 - loss: 0.3619 - val_accuracy: 0.7597 - val_loss: 0.6617 - learning_rate: 1.9531e-06
Epoch 109/650
236/236 2s 7ms/step - accuracy: 0.8606 - loss: 0.3561 - val_accuracy: 0.7600 - val_loss: 0.6625 - learning_rate: 1.9531e-06
Epoch 110/650
236/236 0s 7ms/step - accuracy: 0.8598 - loss: 0.3617
Epoch 110: ReduceLROnPlateau reducing learning rate to 1e-06.
236/236 2s 7ms/step - accuracy: 0.8598 - loss: 0.3617 - val_accuracy: 0.7596 - val_loss: 0.6623 - learning_rate: 1.9531e-06
Epoch 111/650
236/236 2s 7ms/step - accuracy: 0.8595 - loss: 0.3595 - val_accuracy: 0.7599 - val_loss: 0.6624 - learning_rate: 1.0000e-06
Epoch 112/650
236/236 2s 7ms/step - accuracy: 0.8608 - loss: 0.3576 - val_accuracy: 0.7598 - val_loss: 0.6620 - learning_rate: 1.0000e-06
Epoch 113/650
236/236 2s 7ms/step - accuracy: 0.8607 - loss: 0.3587 - val_accuracy: 0.7598 - val_loss: 0.6620 - learning_rate: 1.0000e-06
Epoch 114/650
236/236 2s 7ms/step - accuracy: 0.8582 - loss: 0.3601 - val_accuracy: 0.7600 - val_loss: 0.6630 - learning_rate: 1.0000e-06
Epoch 115/650
236/236 2s 8ms/step - accuracy: 0.8591 - loss: 0.3571 - val_accuracy: 0.7601 - val_loss: 0.6616 - learning_rate: 1.0000e-06
Epoch 116/650
236/236 2s 7ms/step - accuracy: 0.8606 - loss: 0.3565 - val_accuracy: 0.7602 - val_loss: 0.6619 - learning_rate: 1.0000e-06
Epoch 117/650
236/236 2s 8ms/step - accuracy: 0.8600 - loss: 0.3589 - val_accuracy: 0.7600 - val_loss: 0.6614 - learning_rate: 1.0000e-06
Epoch 118/650
236/236 2s 7ms/step - accuracy: 0.8606 - loss: 0.3568 - val_accuracy: 0.7600 - val_loss: 0.6612 - learning_rate: 1.0000e-06
Epoch 119/650
236/236 2s 7ms/step - accuracy: 0.8600 - loss: 0.3562 - val_accuracy: 0.7599 - val_loss: 0.6617 - learning_rate: 1.0000e-06
Epoch 120/650
236/236 2s 7ms/step - accuracy: 0.8604 - loss: 0.3597 - val_accuracy: 0.7598 - val_loss: 0.6623 - learning_rate: 1.0000e-06
```

```

Epoch 121/650
236/236 2s 7ms/step - accuracy: 0.8580 - loss: 0.3606 - val_accuracy: 0.7599 - val_loss
s: 0.6624 - learning_rate: 1.0000e-06
Epoch 122/650
236/236 2s 7ms/step - accuracy: 0.8594 - loss: 0.3580 - val_accuracy: 0.7599 - val_loss
s: 0.6624 - learning_rate: 1.0000e-06
Epoch 123/650
236/236 2s 7ms/step - accuracy: 0.8601 - loss: 0.3588 - val_accuracy: 0.7601 - val_loss
s: 0.6624 - learning_rate: 1.0000e-06
Epoch 124/650
236/236 2s 7ms/step - accuracy: 0.8583 - loss: 0.3607 - val_accuracy: 0.7598 - val_loss
s: 0.6622 - learning_rate: 1.0000e-06
Epoch 125/650
236/236 2s 7ms/step - accuracy: 0.8560 - loss: 0.3663 - val_accuracy: 0.7598 - val_loss
s: 0.6622 - learning_rate: 1.0000e-06
Epoch 126/650
236/236 2s 8ms/step - accuracy: 0.8627 - loss: 0.3550 - val_accuracy: 0.7598 - val_loss
s: 0.6632 - learning_rate: 1.0000e-06
Epoch 127/650
236/236 2s 7ms/step - accuracy: 0.8597 - loss: 0.3591 - val_accuracy: 0.7599 - val_loss
s: 0.6630 - learning_rate: 1.0000e-06
Epoch 128/650
236/236 2s 8ms/step - accuracy: 0.8589 - loss: 0.3595 - val_accuracy: 0.7601 - val_loss
s: 0.6620 - learning_rate: 1.0000e-06
Epoch 129/650
236/236 2s 7ms/step - accuracy: 0.8584 - loss: 0.3621 - val_accuracy: 0.7599 - val_loss
s: 0.6628 - learning_rate: 1.0000e-06

```

In [55]: `ann7_smote.evaluate(X_train_smote, y_train_smote)`

3770/3770 4s 965us/step - accuracy: 0.8783 - loss: 0.2938

Out[55]: [0.224563330411911, 0.9146376848220825]

In [56]: `ann7_smote.summary()`

Model: "sequential_4"

Layer (type)	Output Shape	Param #
dense_28 (Dense)	(None, 512)	23,040
batch_normalization_24 (BatchNormalization)	(None, 512)	2,048
dropout_24 (Dropout)	(None, 512)	0
dense_29 (Dense)	(None, 256)	131,328
batch_normalization_25 (BatchNormalization)	(None, 256)	1,024
dropout_25 (Dropout)	(None, 256)	0
dense_30 (Dense)	(None, 128)	32,896
batch_normalization_26 (BatchNormalization)	(None, 128)	512
dropout_26 (Dropout)	(None, 128)	0
dense_31 (Dense)	(None, 64)	8,256
batch_normalization_27 (BatchNormalization)	(None, 64)	256
dropout_27 (Dropout)	(None, 64)	0
dense_32 (Dense)	(None, 3)	195

Total params: 594,827 (2.27 MB)

Trainable params: 197,635 (772.01 KB)

Non-trainable params: 1,920 (7.50 KB)

Optimizer params: 395,272 (1.51 MB)

```
In [57]: eval_metric(ann7_smote, X_train_smote, y_train_smote, X_val, y_val)
```

3770/3770 ————— 4s 972us/step
591/591 ————— 1s 977us/step

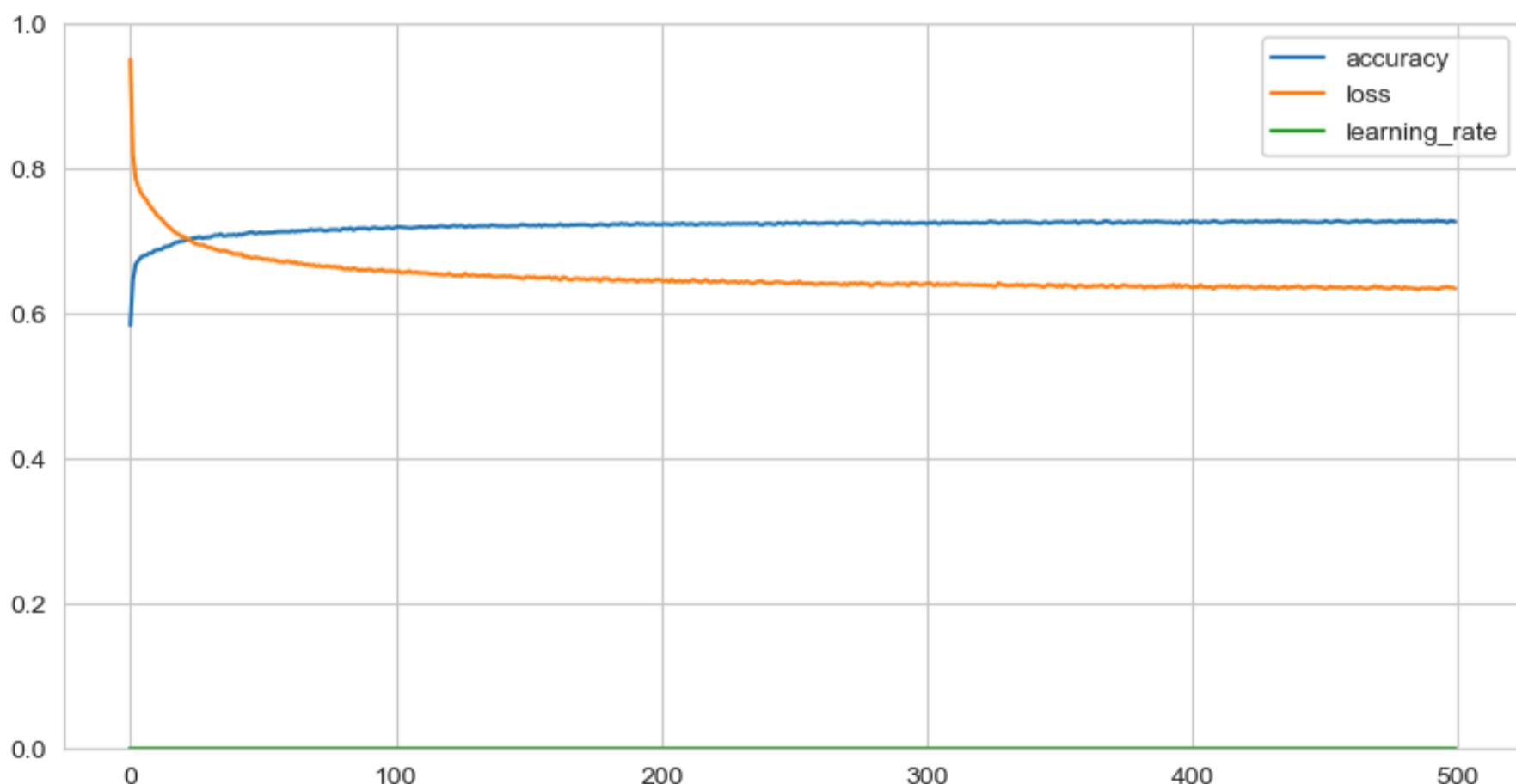
Test Set:

[4615 705 188]			
[1569 6965 1519]			
[85 462 2795]]			
precision	recall	f1-score	support
0 0.74	0.84	0.78	5508
1 0.86	0.69	0.77	10053
2 0.62	0.84	0.71	3342
accuracy		0.76	18903
macro avg	0.74	0.79	0.75
weighted avg	0.78	0.76	0.76

Train Set:

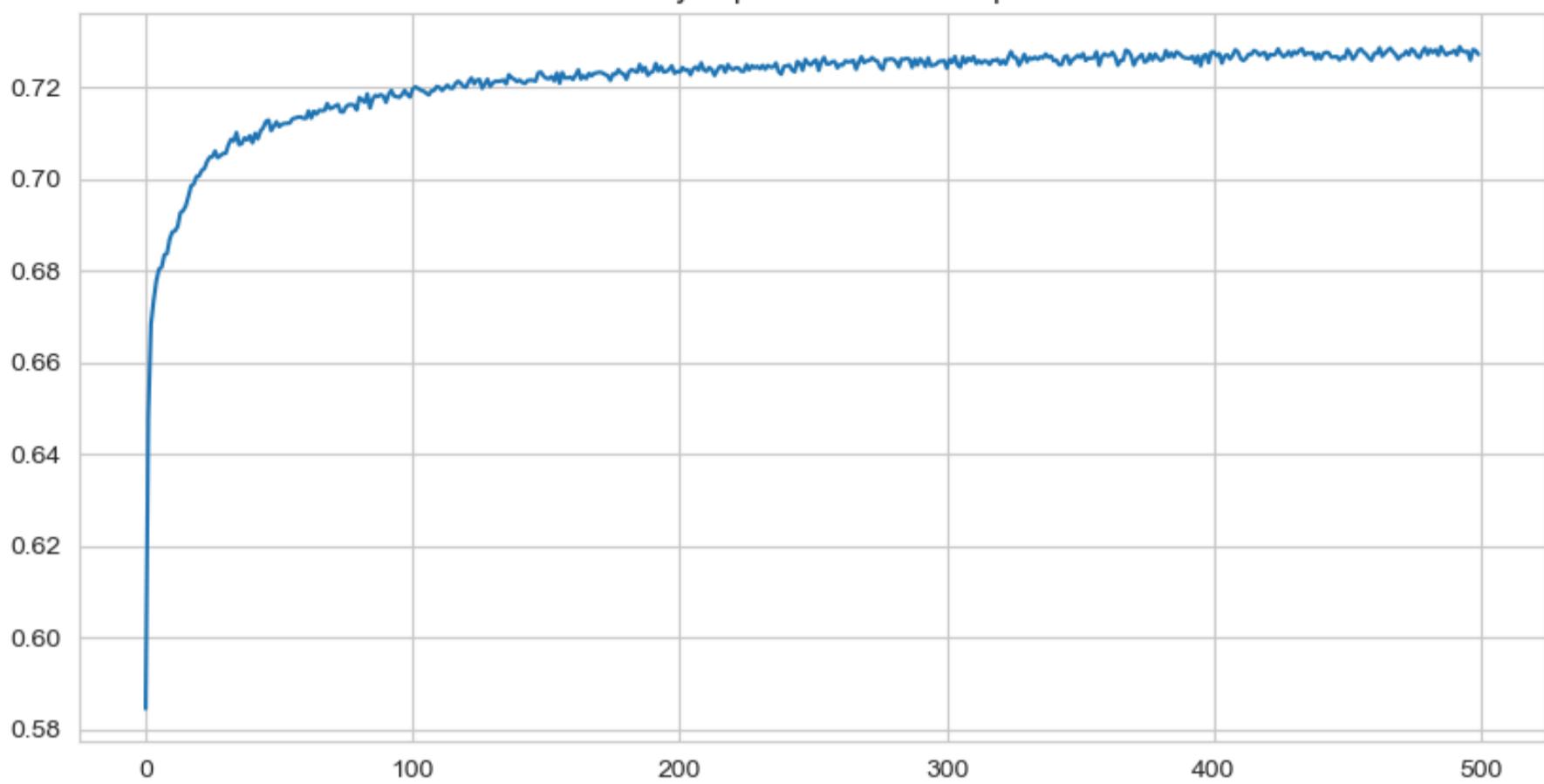
[38808 1021 380]			
[4012 31665 4532]			
[42 310 39857]]			
precision	recall	f1-score	support
0 0.91	0.97	0.93	40209
1 0.96	0.79	0.87	40209
2 0.89	0.99	0.94	40209
accuracy		0.91	120627
macro avg	0.92	0.91	0.91
weighted avg	0.92	0.91	0.91

```
In [215... pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



```
In [214... pd.DataFrame(history.history)[“accuracy”].plot(figsize=(10, 5))
plt.title("Accuracy improvements with Epoch")
plt.show()
```

Accuracy improvements with Epoch



```
In [ ]: # Save the Model

from tensorflow.keras.models import load_model

# Save the model to 'model.h5' file
ann7_smote.save('ann7_smote_model.h5')

# To Load the model Later for further use
loaded_ann7_smote = load_model('ann7_smote_model.h5')
```

ANN-7 Model with SMOTE Summary:

- **(Dense) layers:** 5 / **Neurons:** 512-256-128-64 / **Dropout:** 30-30-25-20% / **Learning Rate:** 0.001 initially reduced to 1e-6 / **Batch Size:** 512 / **Epochs:** 650 / **Early Stop (val_accuracy):** 60
- **Accuracy:** 0.91 / **Val_Accuracy:** 0.76 / **Loss:** 0.2938 / **Val_Loss:** 0.6628 / **Train Recall (Class 2):** 0.99 / **Test Recall (Class 2):** 0.84

Improvements:

- The model exhibits strong performance on the training set with an accuracy of 0.91 and a nearly perfect recall for Class 2 at 0.99. This indicates high sensitivity towards the minority class, enhanced by SMOTE to balance the class distribution.
- The addition of `ReduceLROnPlateau` effectively managed learning rate adjustments, contributing to maintaining model stability over extended training.

No Improvement:

- Validation accuracy shows a noticeable gap compared to the training accuracy, indicating potential overfitting or the need for further adjustments in the model to enhance its generalization to unseen data.
- Despite strong training metrics, the test set performance, especially in precision and recall for certain classes, indicates room for improvement.

Got Worse:

- Validation loss shows slight variability and remains higher compared to the training loss, suggesting that the model might be overly fitted to the training data. This could indicate a need for further tuning in dropout rates or possibly exploring alternative regularization techniques.

ANN-7 Model with Class_Weight (%75)

```
In [217]: y_train.value_counts()
```

```
Out[217]: Credit_Score
1    40209
0    22031
2    13370
Name: count, dtype: int64
```

```
In [47]: from sklearn.utils import class_weight

class_weights = class_weight.compute_class_weight('balanced',
                                                    classes=np.unique(y_train),
                                                    y=y_train)
class_weights_dict = {i: class_weights[i] for i in range(len(class_weights))}

class_weights_dict
```

```
Out[47]: {0: 1.1439940689634303, 1: 0.6268082601739245, 2: 1.8850660683121416}
```

```
In [64]: # ANN-7 Model with Class_Weight
          # BatchNormalization(),
          # ReduceLROnPlateau callback

# 1) Model Architecture:
# Fully connected (Dense) Layers number= 5
ann7_cw = Sequential([
    Dense(256, input_dim=X_train.shape[1], activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(256, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.2),
    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.2),
    Dense(3, activation='softmax')
])

# 2) Compiling the Model:
ann7_cw.compile(optimizer=Adam(learning_rate=0.001),
                 loss='sparse_categorical_crossentropy',
                 metrics=['accuracy'])

# 3) Early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=60,
                               mode="auto",
                               restore_best_weights=True)

# 4) ReduceLROnPlateau callback
reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                             factor=0.5, # Learning rate reduced by half if no improvement
                             patience=10, # Wait 10 epochs before reducing Learning rate
                             min_lr=1e-6, # Minimum Learning rate set
                             verbose=1)

# 5) Train the model
history = ann7_cw.fit(
    x=X_train,
    y=y_train.values,
    validation_data=(X_val, y_val),
    batch_size=512,
    epochs=600,
    verbose=1,
    callbacks=[early_stopping, reduce_lr], # + ReduceLROnPlateau to callbacks
    class_weight = class_weights_dict      # + class_weight is been used
)
```

Epoch 1/600
148/148 5s 8ms/step - accuracy: 0.4641 - loss: 1.1310 - val_accuracy: 0.5967 - val_loss: 0.8656 - learning_rate: 0.0010
Epoch 2/600
148/148 1s 7ms/step - accuracy: 0.5642 - loss: 0.8540 - val_accuracy: 0.6256 - val_loss: 0.8234 - learning_rate: 0.0010
Epoch 3/600
148/148 1s 6ms/step - accuracy: 0.5964 - loss: 0.8077 - val_accuracy: 0.6223 - val_loss: 0.8169 - learning_rate: 0.0010
Epoch 4/600
148/148 1s 7ms/step - accuracy: 0.6150 - loss: 0.7842 - val_accuracy: 0.6346 - val_loss: 0.8200 - learning_rate: 0.0010
Epoch 5/600
148/148 1s 7ms/step - accuracy: 0.6294 - loss: 0.7706 - val_accuracy: 0.6443 - val_loss: 0.7957 - learning_rate: 0.0010
Epoch 6/600
148/148 1s 7ms/step - accuracy: 0.6369 - loss: 0.7540 - val_accuracy: 0.6465 - val_loss: 0.7820 - learning_rate: 0.0010
Epoch 7/600
148/148 1s 6ms/step - accuracy: 0.6444 - loss: 0.7493 - val_accuracy: 0.6477 - val_loss: 0.7828 - learning_rate: 0.0010
Epoch 8/600
148/148 1s 7ms/step - accuracy: 0.6444 - loss: 0.7391 - val_accuracy: 0.6485 - val_loss: 0.7878 - learning_rate: 0.0010
Epoch 9/600
148/148 1s 7ms/step - accuracy: 0.6466 - loss: 0.7358 - val_accuracy: 0.6505 - val_loss: 0.7809 - learning_rate: 0.0010
Epoch 10/600
148/148 1s 6ms/step - accuracy: 0.6528 - loss: 0.7256 - val_accuracy: 0.6564 - val_loss: 0.7809 - learning_rate: 0.0010
Epoch 11/600
148/148 1s 6ms/step - accuracy: 0.6514 - loss: 0.7200 - val_accuracy: 0.6517 - val_loss: 0.7829 - learning_rate: 0.0010
Epoch 12/600
148/148 1s 6ms/step - accuracy: 0.6543 - loss: 0.7190 - val_accuracy: 0.6523 - val_loss: 0.7706 - learning_rate: 0.0010
Epoch 13/600
148/148 1s 6ms/step - accuracy: 0.6550 - loss: 0.7130 - val_accuracy: 0.6565 - val_loss: 0.7652 - learning_rate: 0.0010
Epoch 14/600
148/148 1s 6ms/step - accuracy: 0.6541 - loss: 0.7132 - val_accuracy: 0.6523 - val_loss: 0.7818 - learning_rate: 0.0010
Epoch 15/600
148/148 1s 6ms/step - accuracy: 0.6604 - loss: 0.7085 - val_accuracy: 0.6547 - val_loss: 0.7698 - learning_rate: 0.0010
Epoch 16/600
148/148 1s 6ms/step - accuracy: 0.6555 - loss: 0.7021 - val_accuracy: 0.6581 - val_loss: 0.7691 - learning_rate: 0.0010
Epoch 17/600
148/148 1s 6ms/step - accuracy: 0.6587 - loss: 0.6988 - val_accuracy: 0.6584 - val_loss: 0.7773 - learning_rate: 0.0010
Epoch 18/600
148/148 1s 7ms/step - accuracy: 0.6593 - loss: 0.6957 - val_accuracy: 0.6613 - val_loss: 0.7625 - learning_rate: 0.0010
Epoch 19/600
148/148 1s 6ms/step - accuracy: 0.6607 - loss: 0.6964 - val_accuracy: 0.6587 - val_loss: 0.7638 - learning_rate: 0.0010
Epoch 20/600
148/148 1s 6ms/step - accuracy: 0.6618 - loss: 0.6859 - val_accuracy: 0.6710 - val_loss: 0.7469 - learning_rate: 0.0010
Epoch 21/600
148/148 1s 6ms/step - accuracy: 0.6652 - loss: 0.6867 - val_accuracy: 0.6660 - val_loss: 0.7526 - learning_rate: 0.0010
Epoch 22/600
148/148 1s 6ms/step - accuracy: 0.6631 - loss: 0.6809 - val_accuracy: 0.6628 - val_loss: 0.7671 - learning_rate: 0.0010
Epoch 23/600
148/148 1s 6ms/step - accuracy: 0.6627 - loss: 0.6861 - val_accuracy: 0.6614 - val_loss: 0.7524 - learning_rate: 0.0010
Epoch 24/600
148/148 1s 6ms/step - accuracy: 0.6626 - loss: 0.6805 - val_accuracy: 0.6686 - val_loss: 0.7509 - learning_rate: 0.0010
Epoch 25/600
148/148 1s 6ms/step - accuracy: 0.6689 - loss: 0.6708 - val_accuracy: 0.6646 - val_loss: 0.7498 - learning_rate: 0.0010
Epoch 26/600

```
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.6703 - loss: 0.6656 - val_accuracy: 0.6740 - val_loss: 0.7495 - learning_rate: 0.0010
Epoch 27/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6716 - loss: 0.6644 - val_accuracy: 0.6615 - val_loss: 0.7554 - learning_rate: 0.0010
Epoch 28/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6690 - loss: 0.6635 - val_accuracy: 0.6721 - val_loss: 0.7418 - learning_rate: 0.0010
Epoch 29/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6683 - loss: 0.6605 - val_accuracy: 0.6718 - val_loss: 0.7336 - learning_rate: 0.0010
Epoch 30/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6721 - loss: 0.6550 - val_accuracy: 0.6690 - val_loss: 0.7431 - learning_rate: 0.0010
Epoch 31/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6719 - loss: 0.6568 - val_accuracy: 0.6750 - val_loss: 0.7354 - learning_rate: 0.0010
Epoch 32/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6710 - loss: 0.6510 - val_accuracy: 0.6690 - val_loss: 0.7480 - learning_rate: 0.0010
Epoch 33/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6740 - loss: 0.6468 - val_accuracy: 0.6691 - val_loss: 0.7366 - learning_rate: 0.0010
Epoch 34/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6730 - loss: 0.6452 - val_accuracy: 0.6731 - val_loss: 0.7362 - learning_rate: 0.0010
Epoch 35/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.6736 - loss: 0.6481 - val_accuracy: 0.6779 - val_loss: 0.7274 - learning_rate: 0.0010
Epoch 36/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.6770 - loss: 0.6373 - val_accuracy: 0.6734 - val_loss: 0.7469 - learning_rate: 0.0010
Epoch 37/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6749 - loss: 0.6401 - val_accuracy: 0.6710 - val_loss: 0.7297 - learning_rate: 0.0010
Epoch 38/600
148/148 ━━━━━━━━ 1s 8ms/step - accuracy: 0.6752 - loss: 0.6377 - val_accuracy: 0.6710 - val_loss: 0.7421 - learning_rate: 0.0010
Epoch 39/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.6768 - loss: 0.6339 - val_accuracy: 0.6839 - val_loss: 0.7173 - learning_rate: 0.0010
Epoch 40/600
148/148 ━━━━━━━━ 1s 8ms/step - accuracy: 0.6807 - loss: 0.6275 - val_accuracy: 0.6740 - val_loss: 0.7378 - learning_rate: 0.0010
Epoch 41/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.6799 - loss: 0.6293 - val_accuracy: 0.6765 - val_loss: 0.7225 - learning_rate: 0.0010
Epoch 42/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.6810 - loss: 0.6252 - val_accuracy: 0.6817 - val_loss: 0.7211 - learning_rate: 0.0010
Epoch 43/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6822 - loss: 0.6212 - val_accuracy: 0.6805 - val_loss: 0.7227 - learning_rate: 0.0010
Epoch 44/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6812 - loss: 0.6266 - val_accuracy: 0.6814 - val_loss: 0.7116 - learning_rate: 0.0010
Epoch 45/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6799 - loss: 0.6219 - val_accuracy: 0.6822 - val_loss: 0.7206 - learning_rate: 0.0010
Epoch 46/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6883 - loss: 0.6180 - val_accuracy: 0.6725 - val_loss: 0.7309 - learning_rate: 0.0010
Epoch 47/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.6863 - loss: 0.6187 - val_accuracy: 0.6831 - val_loss: 0.7108 - learning_rate: 0.0010
Epoch 48/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6864 - loss: 0.6149 - val_accuracy: 0.6825 - val_loss: 0.7182 - learning_rate: 0.0010
Epoch 49/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6866 - loss: 0.6117 - val_accuracy: 0.6792 - val_loss: 0.7241 - learning_rate: 0.0010
Epoch 50/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6852 - loss: 0.6114 - val_accuracy: 0.6726 - val_loss: 0.7354 - learning_rate: 0.0010
Epoch 51/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.6856 - loss: 0.6139 - val_accuracy: 0.6752 - val_loss:
```

```
s: 0.7151 - learning_rate: 0.0010
Epoch 52/600
148/148 1s 7ms/step - accuracy: 0.6834 - loss: 0.6093 - val_accuracy: 0.6805 - val_loss
s: 0.7115 - learning_rate: 0.0010
Epoch 53/600
148/148 1s 7ms/step - accuracy: 0.6842 - loss: 0.6065 - val_accuracy: 0.6886 - val_loss
s: 0.6944 - learning_rate: 0.0010
Epoch 54/600
148/148 1s 7ms/step - accuracy: 0.6888 - loss: 0.6076 - val_accuracy: 0.6894 - val_loss
s: 0.6971 - learning_rate: 0.0010
Epoch 55/600
148/148 1s 7ms/step - accuracy: 0.6892 - loss: 0.6040 - val_accuracy: 0.6782 - val_loss
s: 0.7115 - learning_rate: 0.0010
Epoch 56/600
148/148 1s 7ms/step - accuracy: 0.6869 - loss: 0.6020 - val_accuracy: 0.6864 - val_loss
s: 0.6942 - learning_rate: 0.0010
Epoch 57/600
148/148 1s 7ms/step - accuracy: 0.6891 - loss: 0.6007 - val_accuracy: 0.6820 - val_loss
s: 0.7089 - learning_rate: 0.0010
Epoch 58/600
148/148 1s 6ms/step - accuracy: 0.6888 - loss: 0.6007 - val_accuracy: 0.6853 - val_loss
s: 0.7005 - learning_rate: 0.0010
Epoch 59/600
148/148 1s 6ms/step - accuracy: 0.6933 - loss: 0.5964 - val_accuracy: 0.6891 - val_loss
s: 0.6964 - learning_rate: 0.0010
Epoch 60/600
148/148 1s 6ms/step - accuracy: 0.6882 - loss: 0.5987 - val_accuracy: 0.6837 - val_loss
s: 0.7088 - learning_rate: 0.0010
Epoch 61/600
148/148 1s 6ms/step - accuracy: 0.6979 - loss: 0.5890 - val_accuracy: 0.6811 - val_loss
s: 0.7030 - learning_rate: 0.0010
Epoch 62/600
148/148 1s 7ms/step - accuracy: 0.6910 - loss: 0.5935 - val_accuracy: 0.6846 - val_loss
s: 0.7175 - learning_rate: 0.0010
Epoch 63/600
148/148 1s 7ms/step - accuracy: 0.6933 - loss: 0.5917 - val_accuracy: 0.6874 - val_loss
s: 0.6971 - learning_rate: 0.0010
Epoch 64/600
148/148 1s 7ms/step - accuracy: 0.6975 - loss: 0.5870 - val_accuracy: 0.6892 - val_loss
s: 0.6945 - learning_rate: 0.0010
Epoch 65/600
148/148 1s 6ms/step - accuracy: 0.6919 - loss: 0.5904 - val_accuracy: 0.6946 - val_loss
s: 0.6852 - learning_rate: 0.0010
Epoch 66/600
148/148 1s 6ms/step - accuracy: 0.6941 - loss: 0.5912 - val_accuracy: 0.6930 - val_loss
s: 0.6886 - learning_rate: 0.0010
Epoch 67/600
148/148 1s 6ms/step - accuracy: 0.7010 - loss: 0.5849 - val_accuracy: 0.6860 - val_loss
s: 0.7166 - learning_rate: 0.0010
Epoch 68/600
148/148 1s 7ms/step - accuracy: 0.6965 - loss: 0.5892 - val_accuracy: 0.6861 - val_loss
s: 0.7022 - learning_rate: 0.0010
Epoch 69/600
148/148 1s 6ms/step - accuracy: 0.6951 - loss: 0.5869 - val_accuracy: 0.6892 - val_loss
s: 0.6973 - learning_rate: 0.0010
Epoch 70/600
148/148 1s 7ms/step - accuracy: 0.6977 - loss: 0.5822 - val_accuracy: 0.6870 - val_loss
s: 0.7028 - learning_rate: 0.0010
Epoch 71/600
148/148 1s 6ms/step - accuracy: 0.6993 - loss: 0.5795 - val_accuracy: 0.6916 - val_loss
s: 0.7016 - learning_rate: 0.0010
Epoch 72/600
148/148 1s 6ms/step - accuracy: 0.6994 - loss: 0.5788 - val_accuracy: 0.6910 - val_loss
s: 0.6942 - learning_rate: 0.0010
Epoch 73/600
148/148 1s 6ms/step - accuracy: 0.6999 - loss: 0.5824 - val_accuracy: 0.6908 - val_loss
s: 0.6954 - learning_rate: 0.0010
Epoch 74/600
148/148 1s 6ms/step - accuracy: 0.6981 - loss: 0.5810 - val_accuracy: 0.6954 - val_loss
s: 0.6963 - learning_rate: 0.0010
Epoch 75/600
145/148 0s 6ms/step - accuracy: 0.6993 - loss: 0.5798
Epoch 75: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
148/148 1s 6ms/step - accuracy: 0.6993 - loss: 0.5798 - val_accuracy: 0.6882 - val_loss
s: 0.6908 - learning_rate: 0.0010
Epoch 76/600
```

148/148 1s 6ms/step - accuracy: 0.7011 - loss: 0.5754 - val_accuracy: 0.6963 - val_loss: 0.6771 - learning_rate: 5.0000e-04
Epoch 77/600
148/148 1s 6ms/step - accuracy: 0.7025 - loss: 0.5715 - val_accuracy: 0.6980 - val_loss: 0.6867 - learning_rate: 5.0000e-04
Epoch 78/600
148/148 1s 6ms/step - accuracy: 0.7094 - loss: 0.5628 - val_accuracy: 0.6962 - val_loss: 0.6865 - learning_rate: 5.0000e-04
Epoch 79/600
148/148 1s 6ms/step - accuracy: 0.7060 - loss: 0.5616 - val_accuracy: 0.6967 - val_loss: 0.6855 - learning_rate: 5.0000e-04
Epoch 80/600
148/148 1s 6ms/step - accuracy: 0.7052 - loss: 0.5648 - val_accuracy: 0.6965 - val_loss: 0.6818 - learning_rate: 5.0000e-04
Epoch 81/600
148/148 1s 7ms/step - accuracy: 0.7067 - loss: 0.5604 - val_accuracy: 0.6954 - val_loss: 0.6839 - learning_rate: 5.0000e-04
Epoch 82/600
148/148 1s 6ms/step - accuracy: 0.7046 - loss: 0.5620 - val_accuracy: 0.6943 - val_loss: 0.6946 - learning_rate: 5.0000e-04
Epoch 83/600
148/148 1s 7ms/step - accuracy: 0.7045 - loss: 0.5591 - val_accuracy: 0.6970 - val_loss: 0.6867 - learning_rate: 5.0000e-04
Epoch 84/600
148/148 1s 7ms/step - accuracy: 0.7058 - loss: 0.5641 - val_accuracy: 0.6981 - val_loss: 0.6784 - learning_rate: 5.0000e-04
Epoch 85/600
148/148 1s 7ms/step - accuracy: 0.7097 - loss: 0.5602 - val_accuracy: 0.6976 - val_loss: 0.6834 - learning_rate: 5.0000e-04
Epoch 86/600
144/148 0s 6ms/step - accuracy: 0.7038 - loss: 0.5631
Epoch 86: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
148/148 1s 7ms/step - accuracy: 0.7040 - loss: 0.5630 - val_accuracy: 0.7003 - val_loss: 0.6862 - learning_rate: 5.0000e-04
Epoch 87/600
148/148 1s 6ms/step - accuracy: 0.7109 - loss: 0.5517 - val_accuracy: 0.6995 - val_loss: 0.6817 - learning_rate: 2.5000e-04
Epoch 88/600
148/148 1s 6ms/step - accuracy: 0.7066 - loss: 0.5553 - val_accuracy: 0.7032 - val_loss: 0.6718 - learning_rate: 2.5000e-04
Epoch 89/600
148/148 1s 6ms/step - accuracy: 0.7124 - loss: 0.5517 - val_accuracy: 0.6999 - val_loss: 0.6795 - learning_rate: 2.5000e-04
Epoch 90/600
148/148 1s 6ms/step - accuracy: 0.7120 - loss: 0.5519 - val_accuracy: 0.6998 - val_loss: 0.6805 - learning_rate: 2.5000e-04
Epoch 91/600
148/148 1s 6ms/step - accuracy: 0.7081 - loss: 0.5551 - val_accuracy: 0.7053 - val_loss: 0.6747 - learning_rate: 2.5000e-04
Epoch 92/600
148/148 1s 6ms/step - accuracy: 0.7126 - loss: 0.5529 - val_accuracy: 0.7036 - val_loss: 0.6781 - learning_rate: 2.5000e-04
Epoch 93/600
148/148 1s 6ms/step - accuracy: 0.7131 - loss: 0.5455 - val_accuracy: 0.7052 - val_loss: 0.6788 - learning_rate: 2.5000e-04
Epoch 94/600
148/148 1s 6ms/step - accuracy: 0.7111 - loss: 0.5496 - val_accuracy: 0.7020 - val_loss: 0.6805 - learning_rate: 2.5000e-04
Epoch 95/600
148/148 1s 6ms/step - accuracy: 0.7136 - loss: 0.5477 - val_accuracy: 0.7030 - val_loss: 0.6756 - learning_rate: 2.5000e-04
Epoch 96/600
148/148 1s 6ms/step - accuracy: 0.7103 - loss: 0.5525 - val_accuracy: 0.7042 - val_loss: 0.6821 - learning_rate: 2.5000e-04
Epoch 97/600
148/148 1s 6ms/step - accuracy: 0.7102 - loss: 0.5478 - val_accuracy: 0.7028 - val_loss: 0.6755 - learning_rate: 2.5000e-04
Epoch 98/600
148/148 0s 5ms/step - accuracy: 0.7140 - loss: 0.5441
Epoch 98: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
148/148 1s 6ms/step - accuracy: 0.7140 - loss: 0.5442 - val_accuracy: 0.7043 - val_loss: 0.6747 - learning_rate: 2.5000e-04
Epoch 99/600
148/148 1s 6ms/step - accuracy: 0.7163 - loss: 0.5416 - val_accuracy: 0.7050 - val_loss: 0.6723 - learning_rate: 1.2500e-04
Epoch 100/600

```
148/148 ----- 1s 6ms/step - accuracy: 0.7129 - loss: 0.5441 - val_accuracy: 0.7039 - val_loss: 0.6754 - learning_rate: 1.2500e-04
Epoch 101/600
148/148 ----- 1s 6ms/step - accuracy: 0.7142 - loss: 0.5465 - val_accuracy: 0.7031 - val_loss: 0.6776 - learning_rate: 1.2500e-04
Epoch 102/600
148/148 ----- 1s 6ms/step - accuracy: 0.7145 - loss: 0.5458 - val_accuracy: 0.7031 - val_loss: 0.6764 - learning_rate: 1.2500e-04
Epoch 103/600
148/148 ----- 1s 7ms/step - accuracy: 0.7132 - loss: 0.5419 - val_accuracy: 0.7042 - val_loss: 0.6755 - learning_rate: 1.2500e-04
Epoch 104/600
148/148 ----- 1s 6ms/step - accuracy: 0.7141 - loss: 0.5448 - val_accuracy: 0.7044 - val_loss: 0.6751 - learning_rate: 1.2500e-04
Epoch 105/600
148/148 ----- 1s 6ms/step - accuracy: 0.7152 - loss: 0.5427 - val_accuracy: 0.7038 - val_loss: 0.6757 - learning_rate: 1.2500e-04
Epoch 106/600
148/148 ----- 1s 6ms/step - accuracy: 0.7161 - loss: 0.5417 - val_accuracy: 0.7048 - val_loss: 0.6751 - learning_rate: 1.2500e-04
Epoch 107/600
148/148 ----- 1s 6ms/step - accuracy: 0.7156 - loss: 0.5386 - val_accuracy: 0.7033 - val_loss: 0.6751 - learning_rate: 1.2500e-04
Epoch 108/600
140/148 ----- 0s 5ms/step - accuracy: 0.7146 - loss: 0.5384
Epoch 108: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
148/148 ----- 1s 6ms/step - accuracy: 0.7147 - loss: 0.5387 - val_accuracy: 0.7036 - val_loss: 0.6762 - learning_rate: 1.2500e-04
Epoch 109/600
148/148 ----- 1s 6ms/step - accuracy: 0.7153 - loss: 0.5388 - val_accuracy: 0.7050 - val_loss: 0.6723 - learning_rate: 6.2500e-05
Epoch 110/600
148/148 ----- 1s 6ms/step - accuracy: 0.7148 - loss: 0.5396 - val_accuracy: 0.7044 - val_loss: 0.6745 - learning_rate: 6.2500e-05
Epoch 111/600
148/148 ----- 1s 6ms/step - accuracy: 0.7177 - loss: 0.5365 - val_accuracy: 0.7054 - val_loss: 0.6752 - learning_rate: 6.2500e-05
Epoch 112/600
148/148 ----- 1s 6ms/step - accuracy: 0.7143 - loss: 0.5408 - val_accuracy: 0.7066 - val_loss: 0.6730 - learning_rate: 6.2500e-05
Epoch 113/600
148/148 ----- 1s 6ms/step - accuracy: 0.7176 - loss: 0.5407 - val_accuracy: 0.7071 - val_loss: 0.6738 - learning_rate: 6.2500e-05
Epoch 114/600
148/148 ----- 1s 6ms/step - accuracy: 0.7167 - loss: 0.5399 - val_accuracy: 0.7068 - val_loss: 0.6712 - learning_rate: 6.2500e-05
Epoch 115/600
148/148 ----- 1s 6ms/step - accuracy: 0.7165 - loss: 0.5403 - val_accuracy: 0.7068 - val_loss: 0.6713 - learning_rate: 6.2500e-05
Epoch 116/600
148/148 ----- 1s 6ms/step - accuracy: 0.7183 - loss: 0.5413 - val_accuracy: 0.7066 - val_loss: 0.6724 - learning_rate: 6.2500e-05
Epoch 117/600
148/148 ----- 1s 7ms/step - accuracy: 0.7166 - loss: 0.5402 - val_accuracy: 0.7066 - val_loss: 0.6727 - learning_rate: 6.2500e-05
Epoch 118/600
148/148 ----- 1s 6ms/step - accuracy: 0.7138 - loss: 0.5383 - val_accuracy: 0.7065 - val_loss: 0.6724 - learning_rate: 6.2500e-05
Epoch 119/600
148/148 ----- 1s 6ms/step - accuracy: 0.7180 - loss: 0.5362 - val_accuracy: 0.7067 - val_loss: 0.6720 - learning_rate: 6.2500e-05
Epoch 120/600
148/148 ----- 1s 7ms/step - accuracy: 0.7198 - loss: 0.5344 - val_accuracy: 0.7069 - val_loss: 0.6753 - learning_rate: 6.2500e-05
Epoch 121/600
148/148 ----- 1s 6ms/step - accuracy: 0.7161 - loss: 0.5362 - val_accuracy: 0.7069 - val_loss: 0.6741 - learning_rate: 6.2500e-05
Epoch 122/600
148/148 ----- 1s 6ms/step - accuracy: 0.7173 - loss: 0.5373 - val_accuracy: 0.7062 - val_loss: 0.6739 - learning_rate: 6.2500e-05
Epoch 123/600
148/148 ----- 1s 6ms/step - accuracy: 0.7157 - loss: 0.5415 - val_accuracy: 0.7060 - val_loss: 0.6743 - learning_rate: 6.2500e-05
Epoch 124/600
143/148 ----- 0s 6ms/step - accuracy: 0.7153 - loss: 0.5430
Epoch 124: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
```

```
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7154 - loss: 0.5429 - val_accuracy: 0.7058 - val_loss: 0.6743 - learning_rate: 6.2500e-05
Epoch 125/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7143 - loss: 0.5405 - val_accuracy: 0.7062 - val_loss: 0.6731 - learning_rate: 3.1250e-05
Epoch 126/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7183 - loss: 0.5348 - val_accuracy: 0.7063 - val_loss: 0.6737 - learning_rate: 3.1250e-05
Epoch 127/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7162 - loss: 0.5389 - val_accuracy: 0.7061 - val_loss: 0.6723 - learning_rate: 3.1250e-05
Epoch 128/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7202 - loss: 0.5327 - val_accuracy: 0.7063 - val_loss: 0.6740 - learning_rate: 3.1250e-05
Epoch 129/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7171 - loss: 0.5396 - val_accuracy: 0.7066 - val_loss: 0.6734 - learning_rate: 3.1250e-05
Epoch 130/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7194 - loss: 0.5360 - val_accuracy: 0.7069 - val_loss: 0.6734 - learning_rate: 3.1250e-05
Epoch 131/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7182 - loss: 0.5372 - val_accuracy: 0.7068 - val_loss: 0.6733 - learning_rate: 3.1250e-05
Epoch 132/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7173 - loss: 0.5381 - val_accuracy: 0.7073 - val_loss: 0.6727 - learning_rate: 3.1250e-05
Epoch 133/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7177 - loss: 0.5378 - val_accuracy: 0.7072 - val_loss: 0.6721 - learning_rate: 3.1250e-05
Epoch 134/600
144/148 ━━━━ 0s 6ms/step - accuracy: 0.7169 - loss: 0.5342
Epoch 134: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7169 - loss: 0.5342 - val_accuracy: 0.7073 - val_loss: 0.6731 - learning_rate: 3.1250e-05
Epoch 135/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7199 - loss: 0.5327 - val_accuracy: 0.7069 - val_loss: 0.6731 - learning_rate: 1.5625e-05
Epoch 136/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7184 - loss: 0.5376 - val_accuracy: 0.7066 - val_loss: 0.6734 - learning_rate: 1.5625e-05
Epoch 137/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7155 - loss: 0.5355 - val_accuracy: 0.7070 - val_loss: 0.6733 - learning_rate: 1.5625e-05
Epoch 138/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7173 - loss: 0.5346 - val_accuracy: 0.7070 - val_loss: 0.6732 - learning_rate: 1.5625e-05
Epoch 139/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7179 - loss: 0.5344 - val_accuracy: 0.7068 - val_loss: 0.6726 - learning_rate: 1.5625e-05
Epoch 140/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.7173 - loss: 0.5357 - val_accuracy: 0.7071 - val_loss: 0.6724 - learning_rate: 1.5625e-05
Epoch 141/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7181 - loss: 0.5383 - val_accuracy: 0.7074 - val_loss: 0.6729 - learning_rate: 1.5625e-05
Epoch 142/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7163 - loss: 0.5411 - val_accuracy: 0.7070 - val_loss: 0.6728 - learning_rate: 1.5625e-05
Epoch 143/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7204 - loss: 0.5353 - val_accuracy: 0.7072 - val_loss: 0.6728 - learning_rate: 1.5625e-05
Epoch 144/600
145/148 ━━━━ 0s 6ms/step - accuracy: 0.7190 - loss: 0.5330
Epoch 144: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7190 - loss: 0.5330 - val_accuracy: 0.7067 - val_loss: 0.6736 - learning_rate: 1.5625e-05
Epoch 145/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7179 - loss: 0.5399 - val_accuracy: 0.7067 - val_loss: 0.6732 - learning_rate: 7.8125e-06
Epoch 146/600
148/148 ━━━━━━ 1s 6ms/step - accuracy: 0.7193 - loss: 0.5338 - val_accuracy: 0.7070 - val_loss: 0.6734 - learning_rate: 7.8125e-06
Epoch 147/600
148/148 ━━━━━━ 1s 7ms/step - accuracy: 0.7164 - loss: 0.5349 - val_accuracy: 0.7077 - val_loss: 0.6718 - learning_rate: 7.8125e-06
Epoch 148/600
```

```
148/148 ----- 1s 6ms/step - accuracy: 0.7162 - loss: 0.5386 - val_accuracy: 0.7075 - val_loss: 0.6726 - learning_rate: 7.8125e-06
Epoch 149/600
148/148 ----- 1s 6ms/step - accuracy: 0.7196 - loss: 0.5326 - val_accuracy: 0.7073 - val_loss: 0.6720 - learning_rate: 7.8125e-06
Epoch 150/600
148/148 ----- 1s 6ms/step - accuracy: 0.7168 - loss: 0.5361 - val_accuracy: 0.7070 - val_loss: 0.6731 - learning_rate: 7.8125e-06
Epoch 151/600
148/148 ----- 1s 6ms/step - accuracy: 0.7192 - loss: 0.5368 - val_accuracy: 0.7072 - val_loss: 0.6725 - learning_rate: 7.8125e-06
Epoch 152/600
148/148 ----- 1s 7ms/step - accuracy: 0.7156 - loss: 0.5372 - val_accuracy: 0.7067 - val_loss: 0.6731 - learning_rate: 7.8125e-06
Epoch 153/600
148/148 ----- 1s 7ms/step - accuracy: 0.7160 - loss: 0.5383 - val_accuracy: 0.7072 - val_loss: 0.6724 - learning_rate: 7.8125e-06
Epoch 154/600
146/148 ----- 0s 6ms/step - accuracy: 0.7219 - loss: 0.5292
Epoch 154: ReduceLROnPlateau reducing learning rate to 3.906250185536919e-06.
148/148 ----- 1s 6ms/step - accuracy: 0.7218 - loss: 0.5292 - val_accuracy: 0.7072 - val_loss: 0.6724 - learning_rate: 7.8125e-06
Epoch 155/600
148/148 ----- 1s 6ms/step - accuracy: 0.7207 - loss: 0.5331 - val_accuracy: 0.7072 - val_loss: 0.6726 - learning_rate: 3.9063e-06
Epoch 156/600
148/148 ----- 1s 6ms/step - accuracy: 0.7203 - loss: 0.5320 - val_accuracy: 0.7072 - val_loss: 0.6727 - learning_rate: 3.9063e-06
Epoch 157/600
148/148 ----- 1s 7ms/step - accuracy: 0.7191 - loss: 0.5376 - val_accuracy: 0.7071 - val_loss: 0.6727 - learning_rate: 3.9063e-06
Epoch 158/600
148/148 ----- 1s 7ms/step - accuracy: 0.7186 - loss: 0.5352 - val_accuracy: 0.7076 - val_loss: 0.6721 - learning_rate: 3.9063e-06
Epoch 159/600
148/148 ----- 1s 8ms/step - accuracy: 0.7188 - loss: 0.5328 - val_accuracy: 0.7076 - val_loss: 0.6721 - learning_rate: 3.9063e-06
Epoch 160/600
148/148 ----- 1s 7ms/step - accuracy: 0.7171 - loss: 0.5371 - val_accuracy: 0.7069 - val_loss: 0.6726 - learning_rate: 3.9063e-06
Epoch 161/600
148/148 ----- 1s 7ms/step - accuracy: 0.7163 - loss: 0.5371 - val_accuracy: 0.7071 - val_loss: 0.6724 - learning_rate: 3.9063e-06
Epoch 162/600
148/148 ----- 1s 7ms/step - accuracy: 0.7174 - loss: 0.5333 - val_accuracy: 0.7076 - val_loss: 0.6721 - learning_rate: 3.9063e-06
Epoch 163/600
148/148 ----- 1s 7ms/step - accuracy: 0.7185 - loss: 0.5343 - val_accuracy: 0.7076 - val_loss: 0.6718 - learning_rate: 3.9063e-06
Epoch 164/600
141/148 ----- 0s 6ms/step - accuracy: 0.7206 - loss: 0.5336
Epoch 164: ReduceLROnPlateau reducing learning rate to 1.9531250927684596e-06.
148/148 ----- 1s 7ms/step - accuracy: 0.7206 - loss: 0.5337 - val_accuracy: 0.7076 - val_loss: 0.6722 - learning_rate: 3.9063e-06
Epoch 165/600
148/148 ----- 1s 7ms/step - accuracy: 0.7183 - loss: 0.5347 - val_accuracy: 0.7076 - val_loss: 0.6723 - learning_rate: 1.9531e-06
Epoch 166/600
148/148 ----- 1s 8ms/step - accuracy: 0.7180 - loss: 0.5351 - val_accuracy: 0.7075 - val_loss: 0.6722 - learning_rate: 1.9531e-06
Epoch 167/600
148/148 ----- 1s 7ms/step - accuracy: 0.7187 - loss: 0.5389 - val_accuracy: 0.7075 - val_loss: 0.6720 - learning_rate: 1.9531e-06
Epoch 168/600
148/148 ----- 1s 7ms/step - accuracy: 0.7187 - loss: 0.5343 - val_accuracy: 0.7072 - val_loss: 0.6725 - learning_rate: 1.9531e-06
Epoch 169/600
148/148 ----- 1s 7ms/step - accuracy: 0.7177 - loss: 0.5384 - val_accuracy: 0.7075 - val_loss: 0.6727 - learning_rate: 1.9531e-06
Epoch 170/600
148/148 ----- 1s 6ms/step - accuracy: 0.7170 - loss: 0.5374 - val_accuracy: 0.7073 - val_loss: 0.6727 - learning_rate: 1.9531e-06
Epoch 171/600
148/148 ----- 1s 6ms/step - accuracy: 0.7160 - loss: 0.5371 - val_accuracy: 0.7073 - val_loss: 0.6729 - learning_rate: 1.9531e-06
Epoch 172/600
```

```
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7167 - loss: 0.5359 - val_accuracy: 0.7075 - val_loss: 0.6725 - learning_rate: 1.9531e-06
Epoch 173/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7199 - loss: 0.5358 - val_accuracy: 0.7076 - val_loss: 0.6720 - learning_rate: 1.9531e-06
Epoch 174/600
146/148 ━━━━ 0s 5ms/step - accuracy: 0.7187 - loss: 0.5373
Epoch 174: ReduceLROnPlateau reducing learning rate to 1e-06.
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7187 - loss: 0.5373 - val_accuracy: 0.7075 - val_loss: 0.6721 - learning_rate: 1.9531e-06
Epoch 175/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.7150 - loss: 0.5380 - val_accuracy: 0.7076 - val_loss: 0.6721 - learning_rate: 1.0000e-06
Epoch 176/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7195 - loss: 0.5305 - val_accuracy: 0.7072 - val_loss: 0.6730 - learning_rate: 1.0000e-06
Epoch 177/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7202 - loss: 0.5325 - val_accuracy: 0.7071 - val_loss: 0.6728 - learning_rate: 1.0000e-06
Epoch 178/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7195 - loss: 0.5349 - val_accuracy: 0.7072 - val_loss: 0.6724 - learning_rate: 1.0000e-06
Epoch 179/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7158 - loss: 0.5385 - val_accuracy: 0.7077 - val_loss: 0.6726 - learning_rate: 1.0000e-06
Epoch 180/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7171 - loss: 0.5355 - val_accuracy: 0.7076 - val_loss: 0.6726 - learning_rate: 1.0000e-06
Epoch 181/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.7211 - loss: 0.5363 - val_accuracy: 0.7073 - val_loss: 0.6726 - learning_rate: 1.0000e-06
Epoch 182/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.7152 - loss: 0.5379 - val_accuracy: 0.7070 - val_loss: 0.6729 - learning_rate: 1.0000e-06
Epoch 183/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.7185 - loss: 0.5342 - val_accuracy: 0.7071 - val_loss: 0.6730 - learning_rate: 1.0000e-06
Epoch 184/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.7202 - loss: 0.5277 - val_accuracy: 0.7076 - val_loss: 0.6725 - learning_rate: 1.0000e-06
Epoch 185/600
148/148 ━━━━━━━━ 1s 8ms/step - accuracy: 0.7176 - loss: 0.5378 - val_accuracy: 0.7075 - val_loss: 0.6722 - learning_rate: 1.0000e-06
Epoch 186/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.7196 - loss: 0.5337 - val_accuracy: 0.7071 - val_loss: 0.6727 - learning_rate: 1.0000e-06
Epoch 187/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.7215 - loss: 0.5293 - val_accuracy: 0.7073 - val_loss: 0.6724 - learning_rate: 1.0000e-06
Epoch 188/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7159 - loss: 0.5350 - val_accuracy: 0.7070 - val_loss: 0.6726 - learning_rate: 1.0000e-06
Epoch 189/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.7183 - loss: 0.5343 - val_accuracy: 0.7069 - val_loss: 0.6730 - learning_rate: 1.0000e-06
Epoch 190/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.7193 - loss: 0.5341 - val_accuracy: 0.7070 - val_loss: 0.6731 - learning_rate: 1.0000e-06
Epoch 191/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7205 - loss: 0.5326 - val_accuracy: 0.7072 - val_loss: 0.6729 - learning_rate: 1.0000e-06
Epoch 192/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7198 - loss: 0.5321 - val_accuracy: 0.7069 - val_loss: 0.6734 - learning_rate: 1.0000e-06
Epoch 193/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.7181 - loss: 0.5347 - val_accuracy: 0.7072 - val_loss: 0.6728 - learning_rate: 1.0000e-06
Epoch 194/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.7189 - loss: 0.5318 - val_accuracy: 0.7071 - val_loss: 0.6722 - learning_rate: 1.0000e-06
Epoch 195/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.7178 - loss: 0.5315 - val_accuracy: 0.7074 - val_loss: 0.6723 - learning_rate: 1.0000e-06
Epoch 196/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.7189 - loss: 0.5324 - val_accuracy: 0.7071 - val_loss: 0.6728 - learning_rate: 1.0000e-06
```

Epoch 197/600
148/148 1s 6ms/step - accuracy: 0.7166 - loss: 0.5391 - val_accuracy: 0.7077 - val_loss: 0.6718 - learning_rate: 1.0000e-06
Epoch 198/600
148/148 1s 7ms/step - accuracy: 0.7206 - loss: 0.5371 - val_accuracy: 0.7073 - val_loss: 0.6726 - learning_rate: 1.0000e-06
Epoch 199/600
148/148 1s 7ms/step - accuracy: 0.7197 - loss: 0.5333 - val_accuracy: 0.7071 - val_loss: 0.6731 - learning_rate: 1.0000e-06
Epoch 200/600
148/148 1s 7ms/step - accuracy: 0.7194 - loss: 0.5363 - val_accuracy: 0.7067 - val_loss: 0.6735 - learning_rate: 1.0000e-06
Epoch 201/600
148/148 1s 6ms/step - accuracy: 0.7214 - loss: 0.5363 - val_accuracy: 0.7072 - val_loss: 0.6729 - learning_rate: 1.0000e-06
Epoch 202/600
148/148 1s 6ms/step - accuracy: 0.7154 - loss: 0.5358 - val_accuracy: 0.7071 - val_loss: 0.6728 - learning_rate: 1.0000e-06
Epoch 203/600
148/148 1s 6ms/step - accuracy: 0.7188 - loss: 0.5367 - val_accuracy: 0.7070 - val_loss: 0.6727 - learning_rate: 1.0000e-06
Epoch 204/600
148/148 1s 7ms/step - accuracy: 0.7170 - loss: 0.5375 - val_accuracy: 0.7069 - val_loss: 0.6730 - learning_rate: 1.0000e-06
Epoch 205/600
148/148 1s 6ms/step - accuracy: 0.7194 - loss: 0.5322 - val_accuracy: 0.7076 - val_loss: 0.6721 - learning_rate: 1.0000e-06
Epoch 206/600
148/148 1s 6ms/step - accuracy: 0.7191 - loss: 0.5387 - val_accuracy: 0.7075 - val_loss: 0.6718 - learning_rate: 1.0000e-06
Epoch 207/600
148/148 1s 6ms/step - accuracy: 0.7212 - loss: 0.5334 - val_accuracy: 0.7071 - val_loss: 0.6722 - learning_rate: 1.0000e-06
Epoch 208/600
148/148 1s 7ms/step - accuracy: 0.7196 - loss: 0.5336 - val_accuracy: 0.7070 - val_loss: 0.6726 - learning_rate: 1.0000e-06
Epoch 209/600
148/148 1s 6ms/step - accuracy: 0.7223 - loss: 0.5288 - val_accuracy: 0.7076 - val_loss: 0.6718 - learning_rate: 1.0000e-06
Epoch 210/600
148/148 1s 6ms/step - accuracy: 0.7195 - loss: 0.5360 - val_accuracy: 0.7071 - val_loss: 0.6729 - learning_rate: 1.0000e-06
Epoch 211/600
148/148 1s 6ms/step - accuracy: 0.7190 - loss: 0.5360 - val_accuracy: 0.7071 - val_loss: 0.6733 - learning_rate: 1.0000e-06
Epoch 212/600
148/148 1s 7ms/step - accuracy: 0.7161 - loss: 0.5354 - val_accuracy: 0.7072 - val_loss: 0.6725 - learning_rate: 1.0000e-06
Epoch 213/600
148/148 1s 7ms/step - accuracy: 0.7215 - loss: 0.5334 - val_accuracy: 0.7075 - val_loss: 0.6724 - learning_rate: 1.0000e-06
Epoch 214/600
148/148 1s 7ms/step - accuracy: 0.7186 - loss: 0.5359 - val_accuracy: 0.7071 - val_loss: 0.6729 - learning_rate: 1.0000e-06
Epoch 215/600
148/148 1s 7ms/step - accuracy: 0.7143 - loss: 0.5421 - val_accuracy: 0.7074 - val_loss: 0.6729 - learning_rate: 1.0000e-06
Epoch 216/600
148/148 1s 6ms/step - accuracy: 0.7197 - loss: 0.5355 - val_accuracy: 0.7068 - val_loss: 0.6732 - learning_rate: 1.0000e-06
Epoch 217/600
148/148 1s 7ms/step - accuracy: 0.7170 - loss: 0.5352 - val_accuracy: 0.7070 - val_loss: 0.6725 - learning_rate: 1.0000e-06
Epoch 218/600
148/148 1s 7ms/step - accuracy: 0.7200 - loss: 0.5309 - val_accuracy: 0.7071 - val_loss: 0.6726 - learning_rate: 1.0000e-06
Epoch 219/600
148/148 1s 7ms/step - accuracy: 0.7204 - loss: 0.5294 - val_accuracy: 0.7070 - val_loss: 0.6730 - learning_rate: 1.0000e-06
Epoch 220/600
148/148 1s 7ms/step - accuracy: 0.7227 - loss: 0.5292 - val_accuracy: 0.7071 - val_loss: 0.6729 - learning_rate: 1.0000e-06
Epoch 221/600
148/148 1s 7ms/step - accuracy: 0.7214 - loss: 0.5340 - val_accuracy: 0.7070 - val_loss: 0.6729 - learning_rate: 1.0000e-06
Epoch 222/600

```
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7185 - loss: 0.5412 - val_accuracy: 0.7072 - val_loss: 0.6726 - learning_rate: 1.0000e-06
Epoch 223/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7189 - loss: 0.5333 - val_accuracy: 0.7072 - val_loss: 0.6720 - learning_rate: 1.0000e-06
Epoch 224/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.7143 - loss: 0.5374 - val_accuracy: 0.7069 - val_loss: 0.6728 - learning_rate: 1.0000e-06
Epoch 225/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.7171 - loss: 0.5344 - val_accuracy: 0.7069 - val_loss: 0.6726 - learning_rate: 1.0000e-06
Epoch 226/600
148/148 ━━━━━━━━ 1s 7ms/step - accuracy: 0.7184 - loss: 0.5386 - val_accuracy: 0.7070 - val_loss: 0.6723 - learning_rate: 1.0000e-06
Epoch 227/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7219 - loss: 0.5341 - val_accuracy: 0.7069 - val_loss: 0.6730 - learning_rate: 1.0000e-06
Epoch 228/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7178 - loss: 0.5351 - val_accuracy: 0.7071 - val_loss: 0.6725 - learning_rate: 1.0000e-06
Epoch 229/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7209 - loss: 0.5319 - val_accuracy: 0.7076 - val_loss: 0.6720 - learning_rate: 1.0000e-06
Epoch 230/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7175 - loss: 0.5344 - val_accuracy: 0.7073 - val_loss: 0.6728 - learning_rate: 1.0000e-06
Epoch 231/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7205 - loss: 0.5343 - val_accuracy: 0.7070 - val_loss: 0.6731 - learning_rate: 1.0000e-06
Epoch 232/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7212 - loss: 0.5287 - val_accuracy: 0.7068 - val_loss: 0.6730 - learning_rate: 1.0000e-06
Epoch 233/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7167 - loss: 0.5388 - val_accuracy: 0.7069 - val_loss: 0.6732 - learning_rate: 1.0000e-06
Epoch 234/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7188 - loss: 0.5319 - val_accuracy: 0.7069 - val_loss: 0.6729 - learning_rate: 1.0000e-06
Epoch 235/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7166 - loss: 0.5336 - val_accuracy: 0.7073 - val_loss: 0.6729 - learning_rate: 1.0000e-06
Epoch 237/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7186 - loss: 0.5407 - val_accuracy: 0.7073 - val_loss: 0.6726 - learning_rate: 1.0000e-06
Epoch 238/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7184 - loss: 0.5348 - val_accuracy: 0.7071 - val_loss: 0.6729 - learning_rate: 1.0000e-06
Epoch 239/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7168 - loss: 0.5362 - val_accuracy: 0.7072 - val_loss: 0.6732 - learning_rate: 1.0000e-06
```

```
In [79]: ann7_cw.evaluate(X_train, y_train)
```

```
2363/2363 ━━━━━━━━ 2s 926us/step - accuracy: 0.7605 - loss: 0.5261
```

```
Out[79]: [0.5274642705917358, 0.7589207887649536]
```

```
In [69]: ann7_cw.summary()
```

```
Model: "sequential_8"
```

Layer (type)	Output Shape	Param #
dense_48 (Dense)	(None, 256)	11,520
batch_normalization_32 (BatchNormalization)	(None, 256)	1,024
dropout_40 (Dropout)	(None, 256)	0
dense_49 (Dense)	(None, 256)	65,792
batch_normalization_33 (BatchNormalization)	(None, 256)	1,024
dropout_41 (Dropout)	(None, 256)	0
dense_50 (Dense)	(None, 128)	32,896
batch_normalization_34 (BatchNormalization)	(None, 128)	512
dropout_42 (Dropout)	(None, 128)	0
dense_51 (Dense)	(None, 64)	8,256
batch_normalization_35 (BatchNormalization)	(None, 64)	256
dropout_43 (Dropout)	(None, 64)	0
dense_52 (Dense)	(None, 3)	195

Total params: 361,611 (1.38 MB)

Trainable params: 120,067 (469.01 KB)

Non-trainable params: 1,408 (5.50 KB)

Optimizer params: 240,136 (938.04 KB)

In [72]: eval_metric(ann7_cw, X_train, y_train, X_val, y_val)

2363/2363 ━━━━━━━━ 3s 1ms/step
591/591 ━━━━━━ 1s 974us/step

Test Set:

```
[[4610 550 348]
 [2056 5824 2173]
 [ 84 314 2944]]
```

	precision	recall	f1-score	support
0	0.68	0.84	0.75	5508
1	0.87	0.58	0.70	10053
2	0.54	0.88	0.67	3342
accuracy			0.71	18903
macro avg	0.70	0.77	0.71	18903
weighted avg	0.76	0.71	0.71	18903

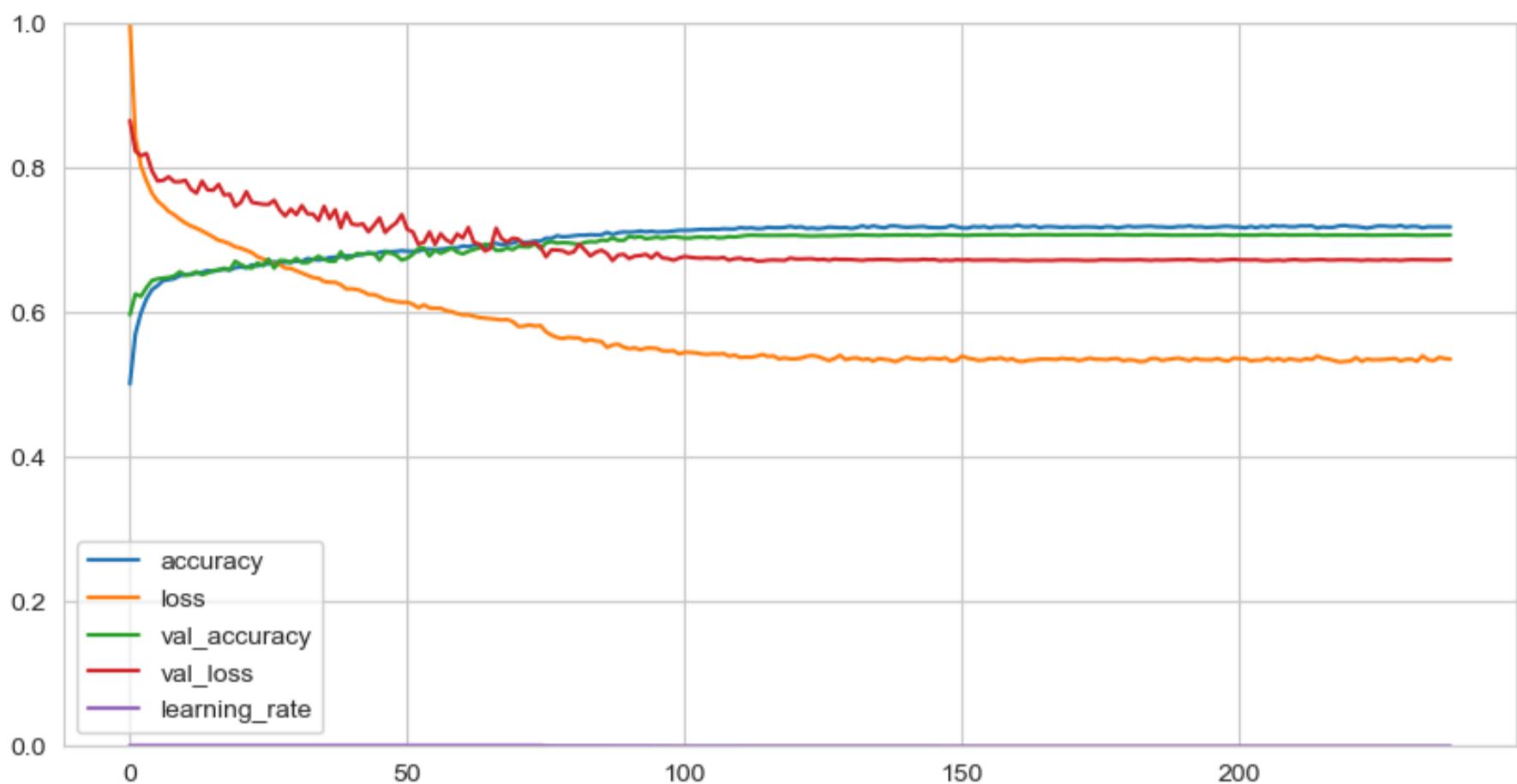
Train Set:

```
[[19712 1387 932]
 [ 7264 24703 8242]
 [ 44 359 12967]]
```

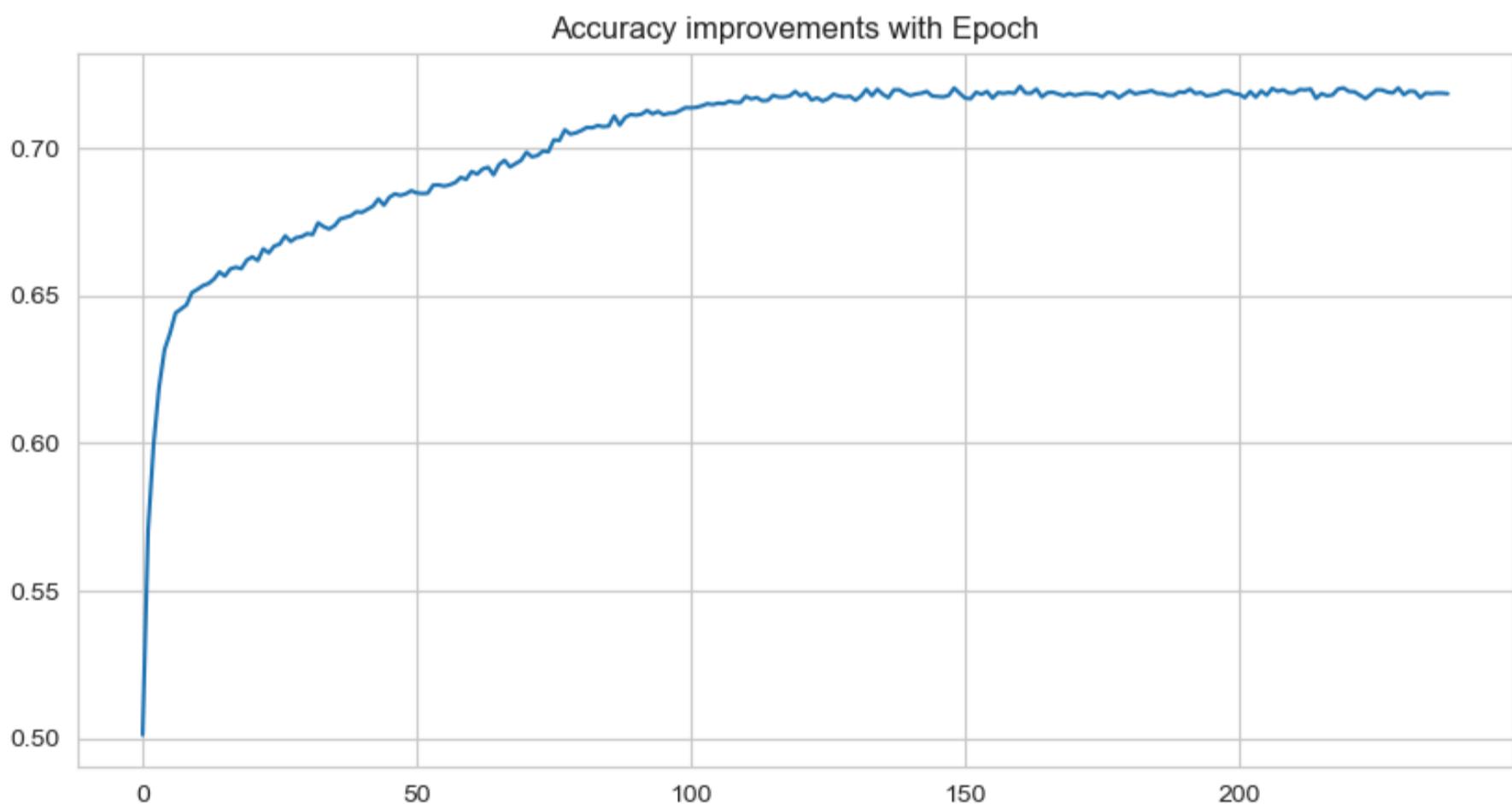
	precision	recall	f1-score	support
0	0.73	0.89	0.80	22031
1	0.93	0.61	0.74	40209
2	0.59	0.97	0.73	13370
accuracy			0.76	75610
macro avg	0.75	0.83	0.76	75610
weighted avg	0.81	0.76	0.76	75610

In [70]: pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)

```
plt.gca().set_ylim(0, 1)
plt.show()
```



```
In [71]: pd.DataFrame(history.history)["accuracy"].plot(figsize=(10, 5))
plt.title("Accuracy improvements with Epoch")
plt.show()
```



ANN-7 Model with Class Weight:

- **(Dense) layers:** 5 / **Neurons:** 256-256-128-64 / **Dropout:** 30-30-20-20% / **Learning Rate:** 0.001 / **Batch Size:** 512 / **Epochs:** 600
- **Accuracy:** 0.76 / **Val_Accuracy:** 0.71 / **Loss:** 0.5261 / **Val_Loss:** 0.6729 / **Train Recall (Class 2):** 0.97 / **Test Recall (Class 2):** 0.88
- Both models (SMOTE and Class Weight ann7) excel in recall for Class 2, yet the SMOTE model shows higher overall accuracy and training performance.
- The Class Weight model demonstrates better generalization, with validation metrics more closely aligned with training results.
- Overfitting is more significant in the SMOTE model due to a wider gap between training and validation results.

ANN-8 Model -Dense +SGD +Momentum (%74)

```
In [81]: from tensorflow.keras.optimizers import SGD

# ANN-8 Model Architecture:
ann8 = Sequential([
    Dense(256, input_dim=X_train.shape[1], activation='relu'), # Reduced neurons
    BatchNormalization(),
    Dropout(0.3),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.25),
    Dense(3, activation='softmax')
])

# Compiling the Model:
ann8.compile(optimizer=SGD(learning_rate=0.01, momentum=0.9), # Using SGD with momentum
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=50, # Slightly reduced patience
                               mode="auto",
                               restore_best_weights=True)

# ReduceLROnPlateau callback
reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                             factor=0.5, # Learning rate reduced by half if no improvement
                             patience=15, # Adjusted patience
                             min_lr=1e-6,
                             verbose=1)

# Train the model
history = ann8.fit(
    x=X_train,
    y=y_train,
    validation_data=(X_val, y_val),
    batch_size=512,
    epochs=600,
    verbose=1,
    callbacks=[early_stopping, reduce_lr]) # Using callbacks
```

Epoch 1/600
148/148 4s 7ms/step - accuracy: 0.4745 - loss: 1.1332 - val_accuracy: 0.5318 - val_loss: 0.9262 - learning_rate: 0.0100
Epoch 2/600
148/148 1s 5ms/step - accuracy: 0.5693 - loss: 0.8486 - val_accuracy: 0.5464 - val_loss: 0.8398 - learning_rate: 0.0100
Epoch 3/600
148/148 1s 5ms/step - accuracy: 0.5909 - loss: 0.8195 - val_accuracy: 0.6008 - val_loss: 0.7960 - learning_rate: 0.0100
Epoch 4/600
148/148 1s 5ms/step - accuracy: 0.6011 - loss: 0.8023 - val_accuracy: 0.6264 - val_loss: 0.7757 - learning_rate: 0.0100
Epoch 5/600
148/148 1s 5ms/step - accuracy: 0.6078 - loss: 0.7924 - val_accuracy: 0.6306 - val_loss: 0.7639 - learning_rate: 0.0100
Epoch 6/600
148/148 1s 6ms/step - accuracy: 0.6166 - loss: 0.7820 - val_accuracy: 0.6460 - val_loss: 0.7580 - learning_rate: 0.0100
Epoch 7/600
148/148 1s 6ms/step - accuracy: 0.6206 - loss: 0.7771 - val_accuracy: 0.6492 - val_loss: 0.7498 - learning_rate: 0.0100
Epoch 8/600
148/148 1s 6ms/step - accuracy: 0.6283 - loss: 0.7662 - val_accuracy: 0.6547 - val_loss: 0.7441 - learning_rate: 0.0100
Epoch 9/600
148/148 1s 5ms/step - accuracy: 0.6333 - loss: 0.7633 - val_accuracy: 0.6576 - val_loss: 0.7348 - learning_rate: 0.0100
Epoch 10/600
148/148 1s 6ms/step - accuracy: 0.6404 - loss: 0.7553 - val_accuracy: 0.6579 - val_loss: 0.7306 - learning_rate: 0.0100
Epoch 11/600
148/148 1s 5ms/step - accuracy: 0.6413 - loss: 0.7514 - val_accuracy: 0.6593 - val_loss: 0.7276 - learning_rate: 0.0100
Epoch 12/600
148/148 1s 5ms/step - accuracy: 0.6409 - loss: 0.7508 - val_accuracy: 0.6622 - val_loss: 0.7250 - learning_rate: 0.0100
Epoch 13/600
148/148 1s 5ms/step - accuracy: 0.6477 - loss: 0.7467 - val_accuracy: 0.6615 - val_loss: 0.7252 - learning_rate: 0.0100
Epoch 14/600
148/148 1s 5ms/step - accuracy: 0.6521 - loss: 0.7357 - val_accuracy: 0.6638 - val_loss: 0.7227 - learning_rate: 0.0100
Epoch 15/600
148/148 1s 5ms/step - accuracy: 0.6521 - loss: 0.7375 - val_accuracy: 0.6669 - val_loss: 0.7199 - learning_rate: 0.0100
Epoch 16/600
148/148 1s 5ms/step - accuracy: 0.6562 - loss: 0.7355 - val_accuracy: 0.6683 - val_loss: 0.7175 - learning_rate: 0.0100
Epoch 17/600
148/148 1s 5ms/step - accuracy: 0.6571 - loss: 0.7310 - val_accuracy: 0.6688 - val_loss: 0.7182 - learning_rate: 0.0100
Epoch 18/600
148/148 1s 6ms/step - accuracy: 0.6588 - loss: 0.7287 - val_accuracy: 0.6706 - val_loss: 0.7156 - learning_rate: 0.0100
Epoch 19/600
148/148 1s 6ms/step - accuracy: 0.6559 - loss: 0.7302 - val_accuracy: 0.6712 - val_loss: 0.7141 - learning_rate: 0.0100
Epoch 20/600
148/148 1s 5ms/step - accuracy: 0.6651 - loss: 0.7228 - val_accuracy: 0.6728 - val_loss: 0.7119 - learning_rate: 0.0100
Epoch 21/600
148/148 1s 5ms/step - accuracy: 0.6631 - loss: 0.7249 - val_accuracy: 0.6727 - val_loss: 0.7118 - learning_rate: 0.0100
Epoch 22/600
148/148 1s 5ms/step - accuracy: 0.6613 - loss: 0.7228 - val_accuracy: 0.6712 - val_loss: 0.7135 - learning_rate: 0.0100
Epoch 23/600
148/148 1s 5ms/step - accuracy: 0.6627 - loss: 0.7232 - val_accuracy: 0.6725 - val_loss: 0.7116 - learning_rate: 0.0100
Epoch 24/600
148/148 1s 5ms/step - accuracy: 0.6665 - loss: 0.7254 - val_accuracy: 0.6750 - val_loss: 0.7096 - learning_rate: 0.0100
Epoch 25/600
148/148 1s 6ms/step - accuracy: 0.6659 - loss: 0.7194 - val_accuracy: 0.6752 - val_loss: 0.7110 - learning_rate: 0.0100
Epoch 26/600

148/148 1s 6ms/step - accuracy: 0.6618 - loss: 0.7248 - val_accuracy: 0.6756 - val_loss: 0.7095 - learning_rate: 0.0100
Epoch 27/600
148/148 1s 5ms/step - accuracy: 0.6648 - loss: 0.7228 - val_accuracy: 0.6762 - val_loss: 0.7117 - learning_rate: 0.0100
Epoch 28/600
148/148 1s 5ms/step - accuracy: 0.6671 - loss: 0.7178 - val_accuracy: 0.6770 - val_loss: 0.7083 - learning_rate: 0.0100
Epoch 29/600
148/148 1s 5ms/step - accuracy: 0.6674 - loss: 0.7187 - val_accuracy: 0.6796 - val_loss: 0.7087 - learning_rate: 0.0100
Epoch 30/600
148/148 1s 5ms/step - accuracy: 0.6662 - loss: 0.7190 - val_accuracy: 0.6784 - val_loss: 0.7080 - learning_rate: 0.0100
Epoch 31/600
148/148 1s 5ms/step - accuracy: 0.6722 - loss: 0.7164 - val_accuracy: 0.6788 - val_loss: 0.7082 - learning_rate: 0.0100
Epoch 32/600
148/148 1s 5ms/step - accuracy: 0.6733 - loss: 0.7098 - val_accuracy: 0.6795 - val_loss: 0.7052 - learning_rate: 0.0100
Epoch 33/600
148/148 1s 5ms/step - accuracy: 0.6685 - loss: 0.7131 - val_accuracy: 0.6795 - val_loss: 0.7076 - learning_rate: 0.0100
Epoch 34/600
148/148 1s 6ms/step - accuracy: 0.6710 - loss: 0.7095 - val_accuracy: 0.6810 - val_loss: 0.7042 - learning_rate: 0.0100
Epoch 35/600
148/148 1s 5ms/step - accuracy: 0.6690 - loss: 0.7151 - val_accuracy: 0.6803 - val_loss: 0.7058 - learning_rate: 0.0100
Epoch 36/600
148/148 1s 6ms/step - accuracy: 0.6741 - loss: 0.7108 - val_accuracy: 0.6798 - val_loss: 0.7075 - learning_rate: 0.0100
Epoch 37/600
148/148 1s 5ms/step - accuracy: 0.6743 - loss: 0.7074 - val_accuracy: 0.6802 - val_loss: 0.7048 - learning_rate: 0.0100
Epoch 38/600
148/148 1s 5ms/step - accuracy: 0.6756 - loss: 0.7074 - val_accuracy: 0.6812 - val_loss: 0.7020 - learning_rate: 0.0100
Epoch 39/600
148/148 1s 7ms/step - accuracy: 0.6749 - loss: 0.7070 - val_accuracy: 0.6824 - val_loss: 0.7053 - learning_rate: 0.0100
Epoch 40/600
148/148 1s 6ms/step - accuracy: 0.6740 - loss: 0.7104 - val_accuracy: 0.6821 - val_loss: 0.7020 - learning_rate: 0.0100
Epoch 41/600
148/148 1s 6ms/step - accuracy: 0.6750 - loss: 0.7086 - val_accuracy: 0.6808 - val_loss: 0.7030 - learning_rate: 0.0100
Epoch 42/600
148/148 1s 5ms/step - accuracy: 0.6783 - loss: 0.7075 - val_accuracy: 0.6827 - val_loss: 0.7001 - learning_rate: 0.0100
Epoch 43/600
148/148 1s 6ms/step - accuracy: 0.6755 - loss: 0.7077 - val_accuracy: 0.6839 - val_loss: 0.7000 - learning_rate: 0.0100
Epoch 44/600
148/148 1s 6ms/step - accuracy: 0.6817 - loss: 0.7006 - val_accuracy: 0.6835 - val_loss: 0.6993 - learning_rate: 0.0100
Epoch 45/600
148/148 1s 6ms/step - accuracy: 0.6774 - loss: 0.7052 - val_accuracy: 0.6833 - val_loss: 0.7004 - learning_rate: 0.0100
Epoch 46/600
148/148 1s 6ms/step - accuracy: 0.6781 - loss: 0.7088 - val_accuracy: 0.6834 - val_loss: 0.6994 - learning_rate: 0.0100
Epoch 47/600
148/148 1s 6ms/step - accuracy: 0.6804 - loss: 0.7066 - val_accuracy: 0.6844 - val_loss: 0.7001 - learning_rate: 0.0100
Epoch 48/600
148/148 1s 5ms/step - accuracy: 0.6781 - loss: 0.7065 - val_accuracy: 0.6829 - val_loss: 0.7018 - learning_rate: 0.0100
Epoch 49/600
148/148 1s 5ms/step - accuracy: 0.6808 - loss: 0.7000 - val_accuracy: 0.6830 - val_loss: 0.6978 - learning_rate: 0.0100
Epoch 50/600
148/148 1s 5ms/step - accuracy: 0.6808 - loss: 0.7018 - val_accuracy: 0.6838 - val_loss: 0.6971 - learning_rate: 0.0100
Epoch 51/600
148/148 1s 5ms/step - accuracy: 0.6808 - loss: 0.6988 - val_accuracy: 0.6822 - val_loss:

```
s: 0.6983 - learning_rate: 0.0100
Epoch 52/600
148/148 1s 6ms/step - accuracy: 0.6824 - loss: 0.6980 - val_accuracy: 0.6840 - val_loss
s: 0.6970 - learning_rate: 0.0100
Epoch 53/600
148/148 1s 6ms/step - accuracy: 0.6821 - loss: 0.6996 - val_accuracy: 0.6841 - val_loss
s: 0.6976 - learning_rate: 0.0100
Epoch 54/600
148/148 1s 6ms/step - accuracy: 0.6799 - loss: 0.7009 - val_accuracy: 0.6847 - val_loss
s: 0.6987 - learning_rate: 0.0100
Epoch 55/600
148/148 1s 5ms/step - accuracy: 0.6835 - loss: 0.7022 - val_accuracy: 0.6847 - val_loss
s: 0.6965 - learning_rate: 0.0100
Epoch 56/600
148/148 1s 5ms/step - accuracy: 0.6852 - loss: 0.6955 - val_accuracy: 0.6843 - val_loss
s: 0.6961 - learning_rate: 0.0100
Epoch 57/600
148/148 1s 5ms/step - accuracy: 0.6811 - loss: 0.7014 - val_accuracy: 0.6859 - val_loss
s: 0.6937 - learning_rate: 0.0100
Epoch 58/600
148/148 1s 5ms/step - accuracy: 0.6845 - loss: 0.6967 - val_accuracy: 0.6857 - val_loss
s: 0.6934 - learning_rate: 0.0100
Epoch 59/600
148/148 1s 6ms/step - accuracy: 0.6877 - loss: 0.6934 - val_accuracy: 0.6860 - val_loss
s: 0.6946 - learning_rate: 0.0100
Epoch 60/600
148/148 1s 5ms/step - accuracy: 0.6845 - loss: 0.6969 - val_accuracy: 0.6858 - val_loss
s: 0.6950 - learning_rate: 0.0100
Epoch 61/600
148/148 1s 5ms/step - accuracy: 0.6867 - loss: 0.6971 - val_accuracy: 0.6851 - val_loss
s: 0.6959 - learning_rate: 0.0100
Epoch 62/600
148/148 1s 5ms/step - accuracy: 0.6849 - loss: 0.6955 - val_accuracy: 0.6880 - val_loss
s: 0.6927 - learning_rate: 0.0100
Epoch 63/600
148/148 1s 6ms/step - accuracy: 0.6882 - loss: 0.6916 - val_accuracy: 0.6865 - val_loss
s: 0.6950 - learning_rate: 0.0100
Epoch 64/600
148/148 1s 5ms/step - accuracy: 0.6856 - loss: 0.6965 - val_accuracy: 0.6873 - val_loss
s: 0.6975 - learning_rate: 0.0100
Epoch 65/600
148/148 1s 5ms/step - accuracy: 0.6845 - loss: 0.6953 - val_accuracy: 0.6872 - val_loss
s: 0.6918 - learning_rate: 0.0100
Epoch 66/600
148/148 1s 5ms/step - accuracy: 0.6879 - loss: 0.6899 - val_accuracy: 0.6891 - val_loss
s: 0.6915 - learning_rate: 0.0100
Epoch 67/600
148/148 1s 5ms/step - accuracy: 0.6874 - loss: 0.6927 - val_accuracy: 0.6862 - val_loss
s: 0.6942 - learning_rate: 0.0100
Epoch 68/600
148/148 1s 5ms/step - accuracy: 0.6884 - loss: 0.6930 - val_accuracy: 0.6863 - val_loss
s: 0.6939 - learning_rate: 0.0100
Epoch 69/600
148/148 1s 5ms/step - accuracy: 0.6850 - loss: 0.6949 - val_accuracy: 0.6896 - val_loss
s: 0.6904 - learning_rate: 0.0100
Epoch 70/600
148/148 1s 5ms/step - accuracy: 0.6904 - loss: 0.6877 - val_accuracy: 0.6891 - val_loss
s: 0.6907 - learning_rate: 0.0100
Epoch 71/600
148/148 1s 6ms/step - accuracy: 0.6884 - loss: 0.6911 - val_accuracy: 0.6869 - val_loss
s: 0.6927 - learning_rate: 0.0100
Epoch 72/600
148/148 1s 5ms/step - accuracy: 0.6897 - loss: 0.6858 - val_accuracy: 0.6856 - val_loss
s: 0.6922 - learning_rate: 0.0100
Epoch 73/600
148/148 1s 5ms/step - accuracy: 0.6907 - loss: 0.6905 - val_accuracy: 0.6891 - val_loss
s: 0.6908 - learning_rate: 0.0100
Epoch 74/600
148/148 1s 5ms/step - accuracy: 0.6870 - loss: 0.6918 - val_accuracy: 0.6889 - val_loss
s: 0.6904 - learning_rate: 0.0100
Epoch 75/600
148/148 1s 5ms/step - accuracy: 0.6881 - loss: 0.6888 - val_accuracy: 0.6881 - val_loss
s: 0.6931 - learning_rate: 0.0100
Epoch 76/600
148/148 1s 5ms/step - accuracy: 0.6855 - loss: 0.6922 - val_accuracy: 0.6898 - val_loss
s: 0.6889 - learning_rate: 0.0100
```

Epoch 77/600
148/148 1s 5ms/step - accuracy: 0.6912 - loss: 0.6849 - val_accuracy: 0.6917 - val_loss: 0.6888 - learning_rate: 0.0100
Epoch 78/600
148/148 1s 7ms/step - accuracy: 0.6870 - loss: 0.6925 - val_accuracy: 0.6894 - val_loss: 0.6896 - learning_rate: 0.0100
Epoch 79/600
148/148 1s 6ms/step - accuracy: 0.6908 - loss: 0.6859 - val_accuracy: 0.6916 - val_loss: 0.6890 - learning_rate: 0.0100
Epoch 80/600
148/148 1s 6ms/step - accuracy: 0.6877 - loss: 0.6871 - val_accuracy: 0.6899 - val_loss: 0.6889 - learning_rate: 0.0100
Epoch 81/600
148/148 1s 7ms/step - accuracy: 0.6896 - loss: 0.6880 - val_accuracy: 0.6891 - val_loss: 0.6885 - learning_rate: 0.0100
Epoch 82/600
148/148 1s 6ms/step - accuracy: 0.6865 - loss: 0.6872 - val_accuracy: 0.6906 - val_loss: 0.6879 - learning_rate: 0.0100
Epoch 83/600
148/148 1s 5ms/step - accuracy: 0.6896 - loss: 0.6893 - val_accuracy: 0.6892 - val_loss: 0.6880 - learning_rate: 0.0100
Epoch 84/600
148/148 1s 6ms/step - accuracy: 0.6908 - loss: 0.6851 - val_accuracy: 0.6908 - val_loss: 0.6870 - learning_rate: 0.0100
Epoch 85/600
148/148 1s 6ms/step - accuracy: 0.6901 - loss: 0.6863 - val_accuracy: 0.6900 - val_loss: 0.6868 - learning_rate: 0.0100
Epoch 86/600
148/148 1s 5ms/step - accuracy: 0.6904 - loss: 0.6859 - val_accuracy: 0.6902 - val_loss: 0.6862 - learning_rate: 0.0100
Epoch 87/600
148/148 1s 5ms/step - accuracy: 0.6928 - loss: 0.6803 - val_accuracy: 0.6876 - val_loss: 0.6890 - learning_rate: 0.0100
Epoch 88/600
148/148 1s 6ms/step - accuracy: 0.6934 - loss: 0.6839 - val_accuracy: 0.6917 - val_loss: 0.6865 - learning_rate: 0.0100
Epoch 89/600
148/148 1s 5ms/step - accuracy: 0.6925 - loss: 0.6834 - val_accuracy: 0.6895 - val_loss: 0.6879 - learning_rate: 0.0100
Epoch 90/600
148/148 1s 5ms/step - accuracy: 0.6917 - loss: 0.6852 - val_accuracy: 0.6914 - val_loss: 0.6878 - learning_rate: 0.0100
Epoch 91/600
148/148 1s 5ms/step - accuracy: 0.6943 - loss: 0.6850 - val_accuracy: 0.6904 - val_loss: 0.6909 - learning_rate: 0.0100
Epoch 92/600
148/148 1s 6ms/step - accuracy: 0.6945 - loss: 0.6790 - val_accuracy: 0.6920 - val_loss: 0.6875 - learning_rate: 0.0100
Epoch 93/600
148/148 1s 5ms/step - accuracy: 0.6912 - loss: 0.6846 - val_accuracy: 0.6928 - val_loss: 0.6843 - learning_rate: 0.0100
Epoch 94/600
148/148 1s 5ms/step - accuracy: 0.6950 - loss: 0.6812 - val_accuracy: 0.6943 - val_loss: 0.6862 - learning_rate: 0.0100
Epoch 95/600
148/148 1s 6ms/step - accuracy: 0.6958 - loss: 0.6815 - val_accuracy: 0.6931 - val_loss: 0.6841 - learning_rate: 0.0100
Epoch 96/600
148/148 1s 6ms/step - accuracy: 0.6965 - loss: 0.6776 - val_accuracy: 0.6918 - val_loss: 0.6864 - learning_rate: 0.0100
Epoch 97/600
148/148 1s 5ms/step - accuracy: 0.6960 - loss: 0.6802 - val_accuracy: 0.6878 - val_loss: 0.6879 - learning_rate: 0.0100
Epoch 98/600
148/148 1s 6ms/step - accuracy: 0.6954 - loss: 0.6783 - val_accuracy: 0.6935 - val_loss: 0.6832 - learning_rate: 0.0100
Epoch 99/600
148/148 1s 6ms/step - accuracy: 0.6971 - loss: 0.6779 - val_accuracy: 0.6908 - val_loss: 0.6885 - learning_rate: 0.0100
Epoch 100/600
148/148 1s 5ms/step - accuracy: 0.6968 - loss: 0.6770 - val_accuracy: 0.6934 - val_loss: 0.6840 - learning_rate: 0.0100
Epoch 101/600
148/148 1s 5ms/step - accuracy: 0.6984 - loss: 0.6778 - val_accuracy: 0.6911 - val_loss: 0.6871 - learning_rate: 0.0100
Epoch 102/600

```
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6939 - loss: 0.6798 - val_accuracy: 0.6926 - val_loss: 0.6849 - learning_rate: 0.0100
Epoch 103/600
148/148 ━━━━━━━━ 1s 5ms/step - accuracy: 0.6961 - loss: 0.6770 - val_accuracy: 0.6917 - val_loss: 0.6854 - learning_rate: 0.0100
Epoch 104/600
148/148 ━━━━━━━━ 1s 5ms/step - accuracy: 0.6951 - loss: 0.6800 - val_accuracy: 0.6932 - val_loss: 0.6854 - learning_rate: 0.0100
Epoch 105/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6964 - loss: 0.6768 - val_accuracy: 0.6932 - val_loss: 0.6864 - learning_rate: 0.0100
Epoch 106/600
148/148 ━━━━━━━━ 1s 5ms/step - accuracy: 0.6979 - loss: 0.6765 - val_accuracy: 0.6922 - val_loss: 0.6848 - learning_rate: 0.0100
Epoch 107/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6993 - loss: 0.6734 - val_accuracy: 0.6917 - val_loss: 0.6847 - learning_rate: 0.0100
Epoch 108/600
148/148 ━━━━━━━━ 1s 5ms/step - accuracy: 0.6964 - loss: 0.6738 - val_accuracy: 0.6939 - val_loss: 0.6812 - learning_rate: 0.0100
Epoch 109/600
148/148 ━━━━━━━━ 1s 5ms/step - accuracy: 0.6973 - loss: 0.6784 - val_accuracy: 0.6945 - val_loss: 0.6807 - learning_rate: 0.0100
Epoch 110/600
148/148 ━━━━━━━━ 1s 5ms/step - accuracy: 0.6997 - loss: 0.6727 - val_accuracy: 0.6949 - val_loss: 0.6819 - learning_rate: 0.0100
Epoch 111/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7000 - loss: 0.6742 - val_accuracy: 0.6953 - val_loss: 0.6805 - learning_rate: 0.0100
Epoch 112/600
148/148 ━━━━━━━━ 1s 5ms/step - accuracy: 0.6983 - loss: 0.6752 - val_accuracy: 0.6957 - val_loss: 0.6818 - learning_rate: 0.0100
Epoch 113/600
148/148 ━━━━━━━━ 1s 5ms/step - accuracy: 0.6962 - loss: 0.6766 - val_accuracy: 0.6960 - val_loss: 0.6809 - learning_rate: 0.0100
Epoch 114/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7042 - loss: 0.6675 - val_accuracy: 0.6957 - val_loss: 0.6795 - learning_rate: 0.0100
Epoch 115/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6977 - loss: 0.6777 - val_accuracy: 0.6946 - val_loss: 0.6876 - learning_rate: 0.0100
Epoch 116/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7013 - loss: 0.6711 - val_accuracy: 0.6937 - val_loss: 0.6822 - learning_rate: 0.0100
Epoch 117/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6983 - loss: 0.6738 - val_accuracy: 0.6956 - val_loss: 0.6841 - learning_rate: 0.0100
Epoch 118/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6983 - loss: 0.6719 - val_accuracy: 0.6966 - val_loss: 0.6795 - learning_rate: 0.0100
Epoch 119/600
148/148 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7008 - loss: 0.6714 - val_accuracy: 0.6949 - val_loss: 0.6790 - learning_rate: 0.0100
Epoch 120/600
148/148 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7004 - loss: 0.6700 - val_accuracy: 0.6975 - val_loss: 0.6787 - learning_rate: 0.0100
Epoch 121/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7014 - loss: 0.6685 - val_accuracy: 0.6971 - val_loss: 0.6823 - learning_rate: 0.0100
Epoch 122/600
148/148 ━━━━━━━━ 1s 5ms/step - accuracy: 0.6985 - loss: 0.6723 - val_accuracy: 0.6972 - val_loss: 0.6785 - learning_rate: 0.0100
Epoch 123/600
148/148 ━━━━━━━━ 1s 5ms/step - accuracy: 0.6979 - loss: 0.6730 - val_accuracy: 0.6983 - val_loss: 0.6766 - learning_rate: 0.0100
Epoch 124/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6994 - loss: 0.6683 - val_accuracy: 0.6970 - val_loss: 0.6755 - learning_rate: 0.0100
Epoch 125/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.6979 - loss: 0.6683 - val_accuracy: 0.6958 - val_loss: 0.6760 - learning_rate: 0.0100
Epoch 126/600
148/148 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7033 - loss: 0.6673 - val_accuracy: 0.6985 - val_loss: 0.6780 - learning_rate: 0.0100
Epoch 127/600
148/148 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7034 - loss: 0.6649 - val_accuracy: 0.6975 - val_loss:
```

```
s: 0.6760 - learning_rate: 0.0100
Epoch 128/600
148/148 1s 5ms/step - accuracy: 0.7012 - loss: 0.6670 - val_accuracy: 0.6984 - val_loss
s: 0.6779 - learning_rate: 0.0100
Epoch 129/600
148/148 1s 5ms/step - accuracy: 0.6995 - loss: 0.6702 - val_accuracy: 0.6976 - val_loss
s: 0.6797 - learning_rate: 0.0100
Epoch 130/600
148/148 1s 5ms/step - accuracy: 0.7001 - loss: 0.6675 - val_accuracy: 0.6982 - val_loss
s: 0.6751 - learning_rate: 0.0100
Epoch 131/600
148/148 1s 6ms/step - accuracy: 0.7034 - loss: 0.6662 - val_accuracy: 0.6940 - val_loss
s: 0.6808 - learning_rate: 0.0100
Epoch 132/600
148/148 1s 5ms/step - accuracy: 0.7021 - loss: 0.6688 - val_accuracy: 0.6978 - val_loss
s: 0.6767 - learning_rate: 0.0100
Epoch 133/600
148/148 1s 5ms/step - accuracy: 0.7030 - loss: 0.6662 - val_accuracy: 0.6966 - val_loss
s: 0.6765 - learning_rate: 0.0100
Epoch 134/600
148/148 1s 6ms/step - accuracy: 0.7028 - loss: 0.6663 - val_accuracy: 0.6978 - val_loss
s: 0.6784 - learning_rate: 0.0100
Epoch 135/600
148/148 1s 5ms/step - accuracy: 0.7027 - loss: 0.6658 - val_accuracy: 0.6978 - val_loss
s: 0.6761 - learning_rate: 0.0100
Epoch 136/600
148/148 1s 5ms/step - accuracy: 0.7010 - loss: 0.6659 - val_accuracy: 0.6972 - val_loss
s: 0.6777 - learning_rate: 0.0100
Epoch 137/600
148/148 1s 5ms/step - accuracy: 0.7036 - loss: 0.6661 - val_accuracy: 0.6972 - val_loss
s: 0.6791 - learning_rate: 0.0100
Epoch 138/600
148/148 1s 5ms/step - accuracy: 0.7013 - loss: 0.6625 - val_accuracy: 0.6994 - val_loss
s: 0.6754 - learning_rate: 0.0100
Epoch 139/600
148/148 1s 5ms/step - accuracy: 0.7055 - loss: 0.6616 - val_accuracy: 0.6995 - val_loss
s: 0.6765 - learning_rate: 0.0100
Epoch 140/600
148/148 1s 5ms/step - accuracy: 0.7057 - loss: 0.6618 - val_accuracy: 0.6996 - val_loss
s: 0.6801 - learning_rate: 0.0100
Epoch 141/600
148/148 1s 5ms/step - accuracy: 0.7025 - loss: 0.6648 - val_accuracy: 0.6993 - val_loss
s: 0.6750 - learning_rate: 0.0100
Epoch 142/600
148/148 1s 5ms/step - accuracy: 0.7012 - loss: 0.6680 - val_accuracy: 0.7000 - val_loss
s: 0.6753 - learning_rate: 0.0100
Epoch 143/600
148/148 1s 5ms/step - accuracy: 0.7028 - loss: 0.6662 - val_accuracy: 0.7003 - val_loss
s: 0.6745 - learning_rate: 0.0100
Epoch 144/600
148/148 1s 5ms/step - accuracy: 0.7057 - loss: 0.6645 - val_accuracy: 0.6996 - val_loss
s: 0.6768 - learning_rate: 0.0100
Epoch 145/600
148/148 1s 5ms/step - accuracy: 0.7044 - loss: 0.6625 - val_accuracy: 0.7015 - val_loss
s: 0.6734 - learning_rate: 0.0100
Epoch 146/600
148/148 1s 6ms/step - accuracy: 0.7056 - loss: 0.6657 - val_accuracy: 0.6996 - val_loss
s: 0.6763 - learning_rate: 0.0100
Epoch 147/600
148/148 1s 5ms/step - accuracy: 0.7084 - loss: 0.6588 - val_accuracy: 0.7002 - val_loss
s: 0.6730 - learning_rate: 0.0100
Epoch 148/600
148/148 1s 5ms/step - accuracy: 0.7069 - loss: 0.6590 - val_accuracy: 0.6985 - val_loss
s: 0.6764 - learning_rate: 0.0100
Epoch 149/600
148/148 1s 6ms/step - accuracy: 0.7031 - loss: 0.6631 - val_accuracy: 0.7000 - val_loss
s: 0.6750 - learning_rate: 0.0100
Epoch 150/600
148/148 1s 5ms/step - accuracy: 0.7089 - loss: 0.6599 - val_accuracy: 0.7008 - val_loss
s: 0.6726 - learning_rate: 0.0100
Epoch 151/600
148/148 1s 5ms/step - accuracy: 0.7087 - loss: 0.6589 - val_accuracy: 0.7014 - val_loss
s: 0.6708 - learning_rate: 0.0100
Epoch 152/600
148/148 1s 5ms/step - accuracy: 0.7064 - loss: 0.6594 - val_accuracy: 0.6991 - val_loss
s: 0.6748 - learning_rate: 0.0100
```

Epoch 153/600
148/148 1s 5ms/step - accuracy: 0.7086 - loss: 0.6567 - val_accuracy: 0.7015 - val_loss: 0.6701 - learning_rate: 0.0100
Epoch 154/600
148/148 1s 6ms/step - accuracy: 0.7074 - loss: 0.6611 - val_accuracy: 0.7009 - val_loss: 0.6761 - learning_rate: 0.0100
Epoch 155/600
148/148 1s 5ms/step - accuracy: 0.7054 - loss: 0.6637 - val_accuracy: 0.7005 - val_loss: 0.6717 - learning_rate: 0.0100
Epoch 156/600
148/148 1s 5ms/step - accuracy: 0.7087 - loss: 0.6555 - val_accuracy: 0.7006 - val_loss: 0.6715 - learning_rate: 0.0100
Epoch 157/600
148/148 1s 5ms/step - accuracy: 0.7101 - loss: 0.6554 - val_accuracy: 0.6995 - val_loss: 0.6808 - learning_rate: 0.0100
Epoch 158/600
148/148 1s 5ms/step - accuracy: 0.7076 - loss: 0.6648 - val_accuracy: 0.7007 - val_loss: 0.6732 - learning_rate: 0.0100
Epoch 159/600
148/148 1s 5ms/step - accuracy: 0.7089 - loss: 0.6531 - val_accuracy: 0.7003 - val_loss: 0.6744 - learning_rate: 0.0100
Epoch 160/600
148/148 1s 5ms/step - accuracy: 0.7112 - loss: 0.6536 - val_accuracy: 0.7022 - val_loss: 0.6712 - learning_rate: 0.0100
Epoch 161/600
148/148 1s 5ms/step - accuracy: 0.7093 - loss: 0.6565 - val_accuracy: 0.7011 - val_loss: 0.6692 - learning_rate: 0.0100
Epoch 162/600
148/148 1s 6ms/step - accuracy: 0.7092 - loss: 0.6565 - val_accuracy: 0.7013 - val_loss: 0.6721 - learning_rate: 0.0100
Epoch 163/600
148/148 1s 6ms/step - accuracy: 0.7073 - loss: 0.6590 - val_accuracy: 0.7014 - val_loss: 0.6732 - learning_rate: 0.0100
Epoch 164/600
148/148 1s 5ms/step - accuracy: 0.7084 - loss: 0.6573 - val_accuracy: 0.7008 - val_loss: 0.6703 - learning_rate: 0.0100
Epoch 165/600
148/148 1s 5ms/step - accuracy: 0.7087 - loss: 0.6575 - val_accuracy: 0.7042 - val_loss: 0.6751 - learning_rate: 0.0100
Epoch 166/600
148/148 1s 5ms/step - accuracy: 0.7075 - loss: 0.6567 - val_accuracy: 0.7005 - val_loss: 0.6765 - learning_rate: 0.0100
Epoch 167/600
148/148 1s 5ms/step - accuracy: 0.7099 - loss: 0.6572 - val_accuracy: 0.7036 - val_loss: 0.6676 - learning_rate: 0.0100
Epoch 168/600
148/148 1s 5ms/step - accuracy: 0.7068 - loss: 0.6565 - val_accuracy: 0.7012 - val_loss: 0.6776 - learning_rate: 0.0100
Epoch 169/600
148/148 1s 5ms/step - accuracy: 0.7094 - loss: 0.6549 - val_accuracy: 0.7023 - val_loss: 0.6699 - learning_rate: 0.0100
Epoch 170/600
148/148 1s 5ms/step - accuracy: 0.7071 - loss: 0.6576 - val_accuracy: 0.7006 - val_loss: 0.6721 - learning_rate: 0.0100
Epoch 171/600
148/148 1s 5ms/step - accuracy: 0.7138 - loss: 0.6491 - val_accuracy: 0.7029 - val_loss: 0.6689 - learning_rate: 0.0100
Epoch 172/600
148/148 1s 5ms/step - accuracy: 0.7116 - loss: 0.6537 - val_accuracy: 0.7021 - val_loss: 0.6751 - learning_rate: 0.0100
Epoch 173/600
148/148 1s 5ms/step - accuracy: 0.7104 - loss: 0.6554 - val_accuracy: 0.7042 - val_loss: 0.6667 - learning_rate: 0.0100
Epoch 174/600
148/148 1s 5ms/step - accuracy: 0.7116 - loss: 0.6528 - val_accuracy: 0.7029 - val_loss: 0.6664 - learning_rate: 0.0100
Epoch 175/600
148/148 1s 5ms/step - accuracy: 0.7117 - loss: 0.6533 - val_accuracy: 0.7032 - val_loss: 0.6679 - learning_rate: 0.0100
Epoch 176/600
148/148 1s 5ms/step - accuracy: 0.7080 - loss: 0.6543 - val_accuracy: 0.7030 - val_loss: 0.6694 - learning_rate: 0.0100
Epoch 177/600
148/148 1s 5ms/step - accuracy: 0.7106 - loss: 0.6499 - val_accuracy: 0.7036 - val_loss: 0.6752 - learning_rate: 0.0100
Epoch 178/600

148/148 1s 5ms/step - accuracy: 0.7155 - loss: 0.6461 - val_accuracy: 0.7054 - val_loss: 0.6668 - learning_rate: 0.0100
Epoch 179/600
148/148 1s 5ms/step - accuracy: 0.7135 - loss: 0.6468 - val_accuracy: 0.7051 - val_loss: 0.6690 - learning_rate: 0.0100
Epoch 180/600
148/148 1s 5ms/step - accuracy: 0.7101 - loss: 0.6502 - val_accuracy: 0.7037 - val_loss: 0.6707 - learning_rate: 0.0100
Epoch 181/600
148/148 1s 5ms/step - accuracy: 0.7139 - loss: 0.6476 - val_accuracy: 0.7065 - val_loss: 0.6664 - learning_rate: 0.0100
Epoch 182/600
148/148 1s 5ms/step - accuracy: 0.7103 - loss: 0.6515 - val_accuracy: 0.7042 - val_loss: 0.6713 - learning_rate: 0.0100
Epoch 183/600
148/148 1s 5ms/step - accuracy: 0.7112 - loss: 0.6534 - val_accuracy: 0.7060 - val_loss: 0.6678 - learning_rate: 0.0100
Epoch 184/600
148/148 1s 5ms/step - accuracy: 0.7111 - loss: 0.6478 - val_accuracy: 0.7036 - val_loss: 0.6698 - learning_rate: 0.0100
Epoch 185/600
148/148 1s 5ms/step - accuracy: 0.7123 - loss: 0.6470 - val_accuracy: 0.7053 - val_loss: 0.6699 - learning_rate: 0.0100
Epoch 186/600
148/148 1s 5ms/step - accuracy: 0.7115 - loss: 0.6497 - val_accuracy: 0.7040 - val_loss: 0.6650 - learning_rate: 0.0100
Epoch 187/600
148/148 1s 5ms/step - accuracy: 0.7129 - loss: 0.6475 - val_accuracy: 0.7034 - val_loss: 0.6690 - learning_rate: 0.0100
Epoch 188/600
148/148 1s 5ms/step - accuracy: 0.7127 - loss: 0.6523 - val_accuracy: 0.7057 - val_loss: 0.6640 - learning_rate: 0.0100
Epoch 189/600
148/148 1s 5ms/step - accuracy: 0.7132 - loss: 0.6529 - val_accuracy: 0.7036 - val_loss: 0.6678 - learning_rate: 0.0100
Epoch 190/600
148/148 1s 5ms/step - accuracy: 0.7122 - loss: 0.6481 - val_accuracy: 0.7060 - val_loss: 0.6668 - learning_rate: 0.0100
Epoch 191/600
148/148 1s 5ms/step - accuracy: 0.7132 - loss: 0.6495 - val_accuracy: 0.7042 - val_loss: 0.6663 - learning_rate: 0.0100
Epoch 192/600
148/148 1s 5ms/step - accuracy: 0.7149 - loss: 0.6446 - val_accuracy: 0.7057 - val_loss: 0.6661 - learning_rate: 0.0100
Epoch 193/600
148/148 1s 5ms/step - accuracy: 0.7137 - loss: 0.6460 - val_accuracy: 0.7062 - val_loss: 0.6613 - learning_rate: 0.0100
Epoch 194/600
148/148 1s 5ms/step - accuracy: 0.7145 - loss: 0.6487 - val_accuracy: 0.7071 - val_loss: 0.6675 - learning_rate: 0.0100
Epoch 195/600
148/148 1s 5ms/step - accuracy: 0.7153 - loss: 0.6437 - val_accuracy: 0.7032 - val_loss: 0.6687 - learning_rate: 0.0100
Epoch 196/600
148/148 1s 5ms/step - accuracy: 0.7133 - loss: 0.6478 - val_accuracy: 0.7081 - val_loss: 0.6659 - learning_rate: 0.0100
Epoch 197/600
148/148 1s 5ms/step - accuracy: 0.7125 - loss: 0.6482 - val_accuracy: 0.7048 - val_loss: 0.6702 - learning_rate: 0.0100
Epoch 198/600
148/148 1s 5ms/step - accuracy: 0.7110 - loss: 0.6497 - val_accuracy: 0.7064 - val_loss: 0.6623 - learning_rate: 0.0100
Epoch 199/600
148/148 1s 5ms/step - accuracy: 0.7145 - loss: 0.6476 - val_accuracy: 0.7051 - val_loss: 0.6669 - learning_rate: 0.0100
Epoch 200/600
148/148 1s 5ms/step - accuracy: 0.7110 - loss: 0.6522 - val_accuracy: 0.7062 - val_loss: 0.6658 - learning_rate: 0.0100
Epoch 201/600
148/148 1s 5ms/step - accuracy: 0.7125 - loss: 0.6492 - val_accuracy: 0.7067 - val_loss: 0.6656 - learning_rate: 0.0100
Epoch 202/600
148/148 1s 5ms/step - accuracy: 0.7170 - loss: 0.6439 - val_accuracy: 0.7057 - val_loss: 0.6618 - learning_rate: 0.0100
Epoch 203/600
148/148 1s 5ms/step - accuracy: 0.7151 - loss: 0.6439 - val_accuracy: 0.7061 - val_loss:

```
s: 0.6657 - learning_rate: 0.0100
Epoch 204/600
148/148 1s 5ms/step - accuracy: 0.7151 - loss: 0.6434 - val_accuracy: 0.7064 - val_loss
s: 0.6646 - learning_rate: 0.0100
Epoch 205/600
148/148 1s 5ms/step - accuracy: 0.7139 - loss: 0.6426 - val_accuracy: 0.7067 - val_loss
s: 0.6632 - learning_rate: 0.0100
Epoch 206/600
148/148 1s 5ms/step - accuracy: 0.7168 - loss: 0.6431 - val_accuracy: 0.7061 - val_loss
s: 0.6657 - learning_rate: 0.0100
Epoch 207/600
148/148 1s 5ms/step - accuracy: 0.7156 - loss: 0.6455 - val_accuracy: 0.7082 - val_loss
s: 0.6628 - learning_rate: 0.0100
Epoch 208/600
146/148 0s 5ms/step - accuracy: 0.7143 - loss: 0.6471
Epoch 208: ReduceLROnPlateau reducing learning rate to 0.004999999888241291.
148/148 1s 5ms/step - accuracy: 0.7143 - loss: 0.6471 - val_accuracy: 0.7070 - val_loss
s: 0.6642 - learning_rate: 0.0100
Epoch 209/600
148/148 1s 5ms/step - accuracy: 0.7174 - loss: 0.6419 - val_accuracy: 0.7083 - val_loss
s: 0.6606 - learning_rate: 0.0050
Epoch 210/600
148/148 1s 5ms/step - accuracy: 0.7180 - loss: 0.6403 - val_accuracy: 0.7106 - val_loss
s: 0.6586 - learning_rate: 0.0050
Epoch 211/600
148/148 1s 5ms/step - accuracy: 0.7157 - loss: 0.6401 - val_accuracy: 0.7095 - val_loss
s: 0.6624 - learning_rate: 0.0050
Epoch 212/600
148/148 1s 5ms/step - accuracy: 0.7175 - loss: 0.6412 - val_accuracy: 0.7081 - val_loss
s: 0.6636 - learning_rate: 0.0050
Epoch 213/600
148/148 1s 5ms/step - accuracy: 0.7215 - loss: 0.6354 - val_accuracy: 0.7096 - val_loss
s: 0.6581 - learning_rate: 0.0050
Epoch 214/600
148/148 1s 5ms/step - accuracy: 0.7194 - loss: 0.6348 - val_accuracy: 0.7087 - val_loss
s: 0.6599 - learning_rate: 0.0050
Epoch 215/600
148/148 1s 5ms/step - accuracy: 0.7195 - loss: 0.6365 - val_accuracy: 0.7087 - val_loss
s: 0.6599 - learning_rate: 0.0050
Epoch 216/600
148/148 1s 5ms/step - accuracy: 0.7167 - loss: 0.6440 - val_accuracy: 0.7095 - val_loss
s: 0.6597 - learning_rate: 0.0050
Epoch 217/600
148/148 1s 5ms/step - accuracy: 0.7204 - loss: 0.6355 - val_accuracy: 0.7084 - val_loss
s: 0.6602 - learning_rate: 0.0050
Epoch 218/600
148/148 1s 5ms/step - accuracy: 0.7242 - loss: 0.6316 - val_accuracy: 0.7089 - val_loss
s: 0.6622 - learning_rate: 0.0050
Epoch 219/600
148/148 1s 5ms/step - accuracy: 0.7213 - loss: 0.6355 - val_accuracy: 0.7097 - val_loss
s: 0.6612 - learning_rate: 0.0050
Epoch 220/600
148/148 1s 5ms/step - accuracy: 0.7201 - loss: 0.6354 - val_accuracy: 0.7082 - val_loss
s: 0.6633 - learning_rate: 0.0050
Epoch 221/600
148/148 1s 5ms/step - accuracy: 0.7197 - loss: 0.6385 - val_accuracy: 0.7098 - val_loss
s: 0.6592 - learning_rate: 0.0050
Epoch 222/600
148/148 1s 5ms/step - accuracy: 0.7222 - loss: 0.6316 - val_accuracy: 0.7097 - val_loss
s: 0.6582 - learning_rate: 0.0050
Epoch 223/600
148/148 1s 5ms/step - accuracy: 0.7191 - loss: 0.6352 - val_accuracy: 0.7107 - val_loss
s: 0.6587 - learning_rate: 0.0050
Epoch 224/600
148/148 1s 5ms/step - accuracy: 0.7208 - loss: 0.6330 - val_accuracy: 0.7105 - val_loss
s: 0.6587 - learning_rate: 0.0050
Epoch 225/600
148/148 1s 6ms/step - accuracy: 0.7179 - loss: 0.6380 - val_accuracy: 0.7076 - val_loss
s: 0.6611 - learning_rate: 0.0050
Epoch 226/600
148/148 1s 5ms/step - accuracy: 0.7208 - loss: 0.6324 - val_accuracy: 0.7093 - val_loss
s: 0.6586 - learning_rate: 0.0050
Epoch 227/600
148/148 1s 5ms/step - accuracy: 0.7211 - loss: 0.6367 - val_accuracy: 0.7093 - val_loss
s: 0.6585 - learning_rate: 0.0050
Epoch 228/600
```

```
146/148 ----- 0s 5ms/step - accuracy: 0.7189 - loss: 0.6385
Epoch 228: ReduceLROnPlateau reducing learning rate to 0.002499999441206455.
148/148 ----- 1s 5ms/step - accuracy: 0.7189 - loss: 0.6385 - val_accuracy: 0.7081 - val_loss
s: 0.6616 - learning_rate: 0.0050
Epoch 229/600
148/148 ----- 1s 5ms/step - accuracy: 0.7230 - loss: 0.6339 - val_accuracy: 0.7116 - val_loss
s: 0.6587 - learning_rate: 0.0025
Epoch 230/600
148/148 ----- 1s 5ms/step - accuracy: 0.7220 - loss: 0.6329 - val_accuracy: 0.7102 - val_loss
s: 0.6580 - learning_rate: 0.0025
Epoch 231/600
148/148 ----- 1s 7ms/step - accuracy: 0.7228 - loss: 0.6316 - val_accuracy: 0.7112 - val_loss
s: 0.6572 - learning_rate: 0.0025
Epoch 232/600
148/148 ----- 1s 5ms/step - accuracy: 0.7206 - loss: 0.6322 - val_accuracy: 0.7108 - val_loss
s: 0.6582 - learning_rate: 0.0025
Epoch 233/600
148/148 ----- 1s 5ms/step - accuracy: 0.7223 - loss: 0.6326 - val_accuracy: 0.7103 - val_loss
s: 0.6574 - learning_rate: 0.0025
Epoch 234/600
148/148 ----- 1s 5ms/step - accuracy: 0.7239 - loss: 0.6286 - val_accuracy: 0.7100 - val_loss
s: 0.6588 - learning_rate: 0.0025
Epoch 235/600
148/148 ----- 1s 5ms/step - accuracy: 0.7250 - loss: 0.6250 - val_accuracy: 0.7106 - val_loss
s: 0.6589 - learning_rate: 0.0025
Epoch 236/600
148/148 ----- 1s 5ms/step - accuracy: 0.7216 - loss: 0.6309 - val_accuracy: 0.7094 - val_loss
s: 0.6596 - learning_rate: 0.0025
Epoch 237/600
148/148 ----- 1s 5ms/step - accuracy: 0.7214 - loss: 0.6335 - val_accuracy: 0.7107 - val_loss
s: 0.6571 - learning_rate: 0.0025
Epoch 238/600
148/148 ----- 1s 5ms/step - accuracy: 0.7191 - loss: 0.6349 - val_accuracy: 0.7098 - val_loss
s: 0.6586 - learning_rate: 0.0025
Epoch 239/600
148/148 ----- 1s 5ms/step - accuracy: 0.7241 - loss: 0.6310 - val_accuracy: 0.7099 - val_loss
s: 0.6586 - learning_rate: 0.0025
Epoch 240/600
148/148 ----- 1s 5ms/step - accuracy: 0.7244 - loss: 0.6304 - val_accuracy: 0.7099 - val_loss
s: 0.6593 - learning_rate: 0.0025
Epoch 241/600
148/148 ----- 1s 5ms/step - accuracy: 0.7224 - loss: 0.6307 - val_accuracy: 0.7109 - val_loss
s: 0.6583 - learning_rate: 0.0025
Epoch 242/600
148/148 ----- 1s 5ms/step - accuracy: 0.7233 - loss: 0.6263 - val_accuracy: 0.7096 - val_loss
s: 0.6579 - learning_rate: 0.0025
Epoch 243/600
148/148 ----- 1s 5ms/step - accuracy: 0.7206 - loss: 0.6322 - val_accuracy: 0.7095 - val_loss
s: 0.6596 - learning_rate: 0.0025
Epoch 244/600
148/148 ----- 1s 5ms/step - accuracy: 0.7239 - loss: 0.6285 - val_accuracy: 0.7096 - val_loss
s: 0.6586 - learning_rate: 0.0025
Epoch 245/600
148/148 ----- 1s 5ms/step - accuracy: 0.7231 - loss: 0.6308 - val_accuracy: 0.7099 - val_loss
s: 0.6581 - learning_rate: 0.0025
Epoch 246/600
140/148 ----- 0s 5ms/step - accuracy: 0.7248 - loss: 0.6253
Epoch 246: ReduceLROnPlateau reducing learning rate to 0.001249999720603228.
148/148 ----- 1s 5ms/step - accuracy: 0.7247 - loss: 0.6256 - val_accuracy: 0.7090 - val_loss
s: 0.6606 - learning_rate: 0.0025
Epoch 247/600
148/148 ----- 1s 5ms/step - accuracy: 0.7230 - loss: 0.6283 - val_accuracy: 0.7106 - val_loss
s: 0.6586 - learning_rate: 0.0012
Epoch 248/600
148/148 ----- 1s 5ms/step - accuracy: 0.7223 - loss: 0.6311 - val_accuracy: 0.7116 - val_loss
s: 0.6586 - learning_rate: 0.0012
Epoch 249/600
148/148 ----- 1s 5ms/step - accuracy: 0.7214 - loss: 0.6359 - val_accuracy: 0.7103 - val_loss
s: 0.6576 - learning_rate: 0.0012
Epoch 250/600
148/148 ----- 1s 5ms/step - accuracy: 0.7235 - loss: 0.6322 - val_accuracy: 0.7102 - val_loss
s: 0.6577 - learning_rate: 0.0012
Epoch 251/600
148/148 ----- 1s 6ms/step - accuracy: 0.7273 - loss: 0.6255 - val_accuracy: 0.7104 - val_loss
s: 0.6587 - learning_rate: 0.0012
Epoch 252/600
```

```
148/148 ----- 1s 5ms/step - accuracy: 0.7253 - loss: 0.6287 - val_accuracy: 0.7096 - val_loss: 0.6578 - learning_rate: 0.0012
Epoch 253/600
148/148 ----- 1s 5ms/step - accuracy: 0.7202 - loss: 0.6333 - val_accuracy: 0.7104 - val_loss: 0.6573 - learning_rate: 0.0012
Epoch 254/600
148/148 ----- 1s 5ms/step - accuracy: 0.7231 - loss: 0.6292 - val_accuracy: 0.7116 - val_loss: 0.6570 - learning_rate: 0.0012
Epoch 255/600
148/148 ----- 1s 5ms/step - accuracy: 0.7236 - loss: 0.6312 - val_accuracy: 0.7104 - val_loss: 0.6591 - learning_rate: 0.0012
Epoch 256/600
148/148 ----- 1s 5ms/step - accuracy: 0.7231 - loss: 0.6295 - val_accuracy: 0.7105 - val_loss: 0.6574 - learning_rate: 0.0012
Epoch 257/600
148/148 ----- 1s 6ms/step - accuracy: 0.7233 - loss: 0.6308 - val_accuracy: 0.7113 - val_loss: 0.6566 - learning_rate: 0.0012
Epoch 258/600
148/148 ----- 1s 5ms/step - accuracy: 0.7204 - loss: 0.6333 - val_accuracy: 0.7104 - val_loss: 0.6578 - learning_rate: 0.0012
Epoch 259/600
148/148 ----- 1s 5ms/step - accuracy: 0.7229 - loss: 0.6282 - val_accuracy: 0.7097 - val_loss: 0.6585 - learning_rate: 0.0012
Epoch 260/600
148/148 ----- 1s 5ms/step - accuracy: 0.7260 - loss: 0.6262 - val_accuracy: 0.7106 - val_loss: 0.6573 - learning_rate: 0.0012
Epoch 261/600
148/148 ----- 1s 5ms/step - accuracy: 0.7234 - loss: 0.6281 - val_accuracy: 0.7110 - val_loss: 0.6570 - learning_rate: 0.0012
Epoch 262/600
148/148 ----- 1s 5ms/step - accuracy: 0.7266 - loss: 0.6248 - val_accuracy: 0.7106 - val_loss: 0.6558 - learning_rate: 0.0012
Epoch 263/600
148/148 ----- 1s 5ms/step - accuracy: 0.7230 - loss: 0.6314 - val_accuracy: 0.7118 - val_loss: 0.6581 - learning_rate: 0.0012
Epoch 264/600
148/148 ----- 1s 5ms/step - accuracy: 0.7263 - loss: 0.6291 - val_accuracy: 0.7110 - val_loss: 0.6564 - learning_rate: 0.0012
Epoch 265/600
148/148 ----- 1s 5ms/step - accuracy: 0.7211 - loss: 0.6320 - val_accuracy: 0.7113 - val_loss: 0.6571 - learning_rate: 0.0012
Epoch 266/600
148/148 ----- 1s 5ms/step - accuracy: 0.7215 - loss: 0.6296 - val_accuracy: 0.7110 - val_loss: 0.6571 - learning_rate: 0.0012
Epoch 267/600
148/148 ----- 1s 5ms/step - accuracy: 0.7231 - loss: 0.6276 - val_accuracy: 0.7113 - val_loss: 0.6567 - learning_rate: 0.0012
Epoch 268/600
148/148 ----- 1s 5ms/step - accuracy: 0.7249 - loss: 0.6247 - val_accuracy: 0.7102 - val_loss: 0.6567 - learning_rate: 0.0012
Epoch 269/600
148/148 ----- 1s 5ms/step - accuracy: 0.7208 - loss: 0.6296 - val_accuracy: 0.7125 - val_loss: 0.6566 - learning_rate: 0.0012
Epoch 270/600
148/148 ----- 1s 5ms/step - accuracy: 0.7256 - loss: 0.6267 - val_accuracy: 0.7113 - val_loss: 0.6565 - learning_rate: 0.0012
Epoch 271/600
148/148 ----- 1s 5ms/step - accuracy: 0.7238 - loss: 0.6282 - val_accuracy: 0.7106 - val_loss: 0.6586 - learning_rate: 0.0012
Epoch 272/600
148/148 ----- 1s 5ms/step - accuracy: 0.7232 - loss: 0.6299 - val_accuracy: 0.7097 - val_loss: 0.6575 - learning_rate: 0.0012
Epoch 273/600
148/148 ----- 1s 5ms/step - accuracy: 0.7230 - loss: 0.6257 - val_accuracy: 0.7106 - val_loss: 0.6581 - learning_rate: 0.0012
Epoch 274/600
148/148 ----- 1s 6ms/step - accuracy: 0.7266 - loss: 0.6258 - val_accuracy: 0.7111 - val_loss: 0.6560 - learning_rate: 0.0012
Epoch 275/600
148/148 ----- 1s 5ms/step - accuracy: 0.7237 - loss: 0.6257 - val_accuracy: 0.7113 - val_loss: 0.6570 - learning_rate: 0.0012
Epoch 276/600
148/148 ----- 1s 5ms/step - accuracy: 0.7241 - loss: 0.6280 - val_accuracy: 0.7118 - val_loss: 0.6564 - learning_rate: 0.0012
Epoch 277/600
139/148 ----- 0s 5ms/step - accuracy: 0.7225 - loss: 0.6301
```

```
Epoch 277: ReduceLROnPlateau reducing learning rate to 0.0006249999860301614.
148/148 1s 5ms/step - accuracy: 0.7225 - loss: 0.6300 - val_accuracy: 0.7108 - val_loss: 0.6564 - learning_rate: 0.0012
Epoch 278/600
148/148 1s 5ms/step - accuracy: 0.7233 - loss: 0.6266 - val_accuracy: 0.7118 - val_loss: 0.6565 - learning_rate: 6.2500e-04
Epoch 279/600
148/148 1s 5ms/step - accuracy: 0.7217 - loss: 0.6284 - val_accuracy: 0.7112 - val_loss: 0.6565 - learning_rate: 6.2500e-04
Epoch 280/600
148/148 1s 5ms/step - accuracy: 0.7266 - loss: 0.6278 - val_accuracy: 0.7111 - val_loss: 0.6562 - learning_rate: 6.2500e-04
Epoch 281/600
148/148 1s 5ms/step - accuracy: 0.7233 - loss: 0.6271 - val_accuracy: 0.7115 - val_loss: 0.6572 - learning_rate: 6.2500e-04
Epoch 282/600
148/148 1s 6ms/step - accuracy: 0.7231 - loss: 0.6340 - val_accuracy: 0.7102 - val_loss: 0.6572 - learning_rate: 6.2500e-04
Epoch 283/600
148/148 1s 6ms/step - accuracy: 0.7262 - loss: 0.6229 - val_accuracy: 0.7115 - val_loss: 0.6563 - learning_rate: 6.2500e-04
Epoch 284/600
148/148 1s 5ms/step - accuracy: 0.7248 - loss: 0.6251 - val_accuracy: 0.7106 - val_loss: 0.6564 - learning_rate: 6.2500e-04
Epoch 285/600
148/148 1s 5ms/step - accuracy: 0.7219 - loss: 0.6315 - val_accuracy: 0.7106 - val_loss: 0.6565 - learning_rate: 6.2500e-04
Epoch 286/600
148/148 1s 5ms/step - accuracy: 0.7243 - loss: 0.6297 - val_accuracy: 0.7111 - val_loss: 0.6562 - learning_rate: 6.2500e-04
Epoch 287/600
148/148 1s 5ms/step - accuracy: 0.7254 - loss: 0.6271 - val_accuracy: 0.7111 - val_loss: 0.6563 - learning_rate: 6.2500e-04
Epoch 288/600
148/148 1s 5ms/step - accuracy: 0.7251 - loss: 0.6249 - val_accuracy: 0.7109 - val_loss: 0.6562 - learning_rate: 6.2500e-04
Epoch 289/600
148/148 1s 5ms/step - accuracy: 0.7232 - loss: 0.6285 - val_accuracy: 0.7109 - val_loss: 0.6567 - learning_rate: 6.2500e-04
Epoch 290/600
148/148 1s 5ms/step - accuracy: 0.7233 - loss: 0.6267 - val_accuracy: 0.7112 - val_loss: 0.6562 - learning_rate: 6.2500e-04
Epoch 291/600
148/148 1s 6ms/step - accuracy: 0.7217 - loss: 0.6331 - val_accuracy: 0.7114 - val_loss: 0.6565 - learning_rate: 6.2500e-04
Epoch 292/600
143/148 0s 5ms/step - accuracy: 0.7232 - loss: 0.6281
Epoch 292: ReduceLROnPlateau reducing learning rate to 0.0003124999930150807.
148/148 1s 5ms/step - accuracy: 0.7232 - loss: 0.6280 - val_accuracy: 0.7109 - val_loss: 0.6574 - learning_rate: 6.2500e-04
Epoch 293/600
148/148 1s 6ms/step - accuracy: 0.7264 - loss: 0.6242 - val_accuracy: 0.7113 - val_loss: 0.6568 - learning_rate: 3.1250e-04
Epoch 294/600
148/148 1s 5ms/step - accuracy: 0.7204 - loss: 0.6304 - val_accuracy: 0.7111 - val_loss: 0.6564 - learning_rate: 3.1250e-04
Epoch 295/600
148/148 1s 5ms/step - accuracy: 0.7265 - loss: 0.6272 - val_accuracy: 0.7108 - val_loss: 0.6566 - learning_rate: 3.1250e-04
Epoch 296/600
148/148 1s 5ms/step - accuracy: 0.7235 - loss: 0.6245 - val_accuracy: 0.7111 - val_loss: 0.6564 - learning_rate: 3.1250e-04
Epoch 297/600
148/148 1s 5ms/step - accuracy: 0.7234 - loss: 0.6294 - val_accuracy: 0.7112 - val_loss: 0.6566 - learning_rate: 3.1250e-04
Epoch 298/600
148/148 1s 5ms/step - accuracy: 0.7237 - loss: 0.6298 - val_accuracy: 0.7114 - val_loss: 0.6561 - learning_rate: 3.1250e-04
Epoch 299/600
148/148 1s 5ms/step - accuracy: 0.7267 - loss: 0.6255 - val_accuracy: 0.7112 - val_loss: 0.6562 - learning_rate: 3.1250e-04
Epoch 300/600
148/148 1s 5ms/step - accuracy: 0.7245 - loss: 0.6263 - val_accuracy: 0.7108 - val_loss: 0.6567 - learning_rate: 3.1250e-04
Epoch 301/600
148/148 1s 5ms/step - accuracy: 0.7248 - loss: 0.6269 - val_accuracy: 0.7105 - val_loss:
```

```
s: 0.6570 - learning_rate: 3.1250e-04
Epoch 302/600
148/148 1s 5ms/step - accuracy: 0.7241 - loss: 0.6262 - val_accuracy: 0.7104 - val_loss
s: 0.6573 - learning_rate: 3.1250e-04
Epoch 303/600
148/148 1s 5ms/step - accuracy: 0.7249 - loss: 0.6229 - val_accuracy: 0.7103 - val_loss
s: 0.6562 - learning_rate: 3.1250e-04
Epoch 304/600
148/148 1s 5ms/step - accuracy: 0.7232 - loss: 0.6270 - val_accuracy: 0.7100 - val_loss
s: 0.6568 - learning_rate: 3.1250e-04
Epoch 305/600
148/148 1s 5ms/step - accuracy: 0.7241 - loss: 0.6298 - val_accuracy: 0.7112 - val_loss
s: 0.6569 - learning_rate: 3.1250e-04
Epoch 306/600
148/148 1s 5ms/step - accuracy: 0.7240 - loss: 0.6237 - val_accuracy: 0.7110 - val_loss
s: 0.6567 - learning_rate: 3.1250e-04
Epoch 307/600
137/148 0s 5ms/step - accuracy: 0.7258 - loss: 0.6279
Epoch 307: ReduceLROnPlateau reducing learning rate to 0.00015624999650754035.
148/148 1s 5ms/step - accuracy: 0.7257 - loss: 0.6279 - val_accuracy: 0.7113 - val_loss
s: 0.6563 - learning_rate: 3.1250e-04
Epoch 308/600
148/148 1s 5ms/step - accuracy: 0.7231 - loss: 0.6265 - val_accuracy: 0.7115 - val_loss
s: 0.6560 - learning_rate: 1.5625e-04
Epoch 309/600
148/148 1s 5ms/step - accuracy: 0.7256 - loss: 0.6245 - val_accuracy: 0.7110 - val_loss
s: 0.6566 - learning_rate: 1.5625e-04
Epoch 310/600
148/148 1s 5ms/step - accuracy: 0.7265 - loss: 0.6214 - val_accuracy: 0.7108 - val_loss
s: 0.6565 - learning_rate: 1.5625e-04
Epoch 311/600
148/148 1s 5ms/step - accuracy: 0.7240 - loss: 0.6260 - val_accuracy: 0.7111 - val_loss
s: 0.6564 - learning_rate: 1.5625e-04
Epoch 312/600
148/148 1s 5ms/step - accuracy: 0.7245 - loss: 0.6257 - val_accuracy: 0.7110 - val_loss
s: 0.6563 - learning_rate: 1.5625e-04
Epoch 313/600
148/148 1s 6ms/step - accuracy: 0.7292 - loss: 0.6212 - val_accuracy: 0.7108 - val_loss
s: 0.6567 - learning_rate: 1.5625e-04
Epoch 314/600
148/148 1s 5ms/step - accuracy: 0.7261 - loss: 0.6260 - val_accuracy: 0.7113 - val_loss
s: 0.6563 - learning_rate: 1.5625e-04
Epoch 315/600
148/148 1s 5ms/step - accuracy: 0.7239 - loss: 0.6275 - val_accuracy: 0.7112 - val_loss
s: 0.6567 - learning_rate: 1.5625e-04
Epoch 316/600
148/148 1s 5ms/step - accuracy: 0.7248 - loss: 0.6269 - val_accuracy: 0.7112 - val_loss
s: 0.6564 - learning_rate: 1.5625e-04
Epoch 317/600
148/148 1s 5ms/step - accuracy: 0.7233 - loss: 0.6308 - val_accuracy: 0.7111 - val_loss
s: 0.6566 - learning_rate: 1.5625e-04
Epoch 318/600
148/148 1s 5ms/step - accuracy: 0.7268 - loss: 0.6238 - val_accuracy: 0.7111 - val_loss
s: 0.6567 - learning_rate: 1.5625e-04
Epoch 319/600
148/148 1s 5ms/step - accuracy: 0.7222 - loss: 0.6300 - val_accuracy: 0.7109 - val_loss
s: 0.6564 - learning_rate: 1.5625e-04
```

```
In [82]: ann8.evaluate(X_train, y_train)
```

```
2363/2363 2s 867us/step - accuracy: 0.7453 - loss: 0.5751
```

```
Out[82]: [0.5729674696922302, 0.746144711971283]
```

```
In [83]: ann8.summary()
```

```
Model: "sequential_10"
```

Layer (type)	Output Shape	Param #
dense_58 (Dense)	(None, 256)	11,520
batch_normalization_40 (BatchNormalization)	(None, 256)	1,024
dropout_48 (Dropout)	(None, 256)	0
dense_59 (Dense)	(None, 128)	32,896
batch_normalization_41 (BatchNormalization)	(None, 128)	512
dropout_49 (Dropout)	(None, 128)	0
dense_60 (Dense)	(None, 128)	16,512
batch_normalization_42 (BatchNormalization)	(None, 128)	512
dropout_50 (Dropout)	(None, 128)	0
dense_61 (Dense)	(None, 64)	8,256
batch_normalization_43 (BatchNormalization)	(None, 64)	256
dropout_51 (Dropout)	(None, 64)	0
dense_62 (Dense)	(None, 3)	195

Total params: 142,216 (555.54 KB)

Trainable params: 70,531 (275.51 KB)

Non-trainable params: 1,152 (4.50 KB)

Optimizer params: 70,533 (275.52 KB)

In [84]: eval_metric(ann8, X_train, y_train, X_val, y_val)

2363/2363 ━━━━━━━━ 3s 1ms/step
591/591 ━━━━━━ 1s 905us/step

Test Set:

```
[[3843 1282 383]
 [1351 7155 1547]
 [ 90  781 2471]]
```

	precision	recall	f1-score	support
0	0.73	0.70	0.71	5508
1	0.78	0.71	0.74	10053
2	0.56	0.74	0.64	3342
accuracy			0.71	18903
macro avg	0.69	0.72	0.70	18903
weighted avg	0.72	0.71	0.72	18903

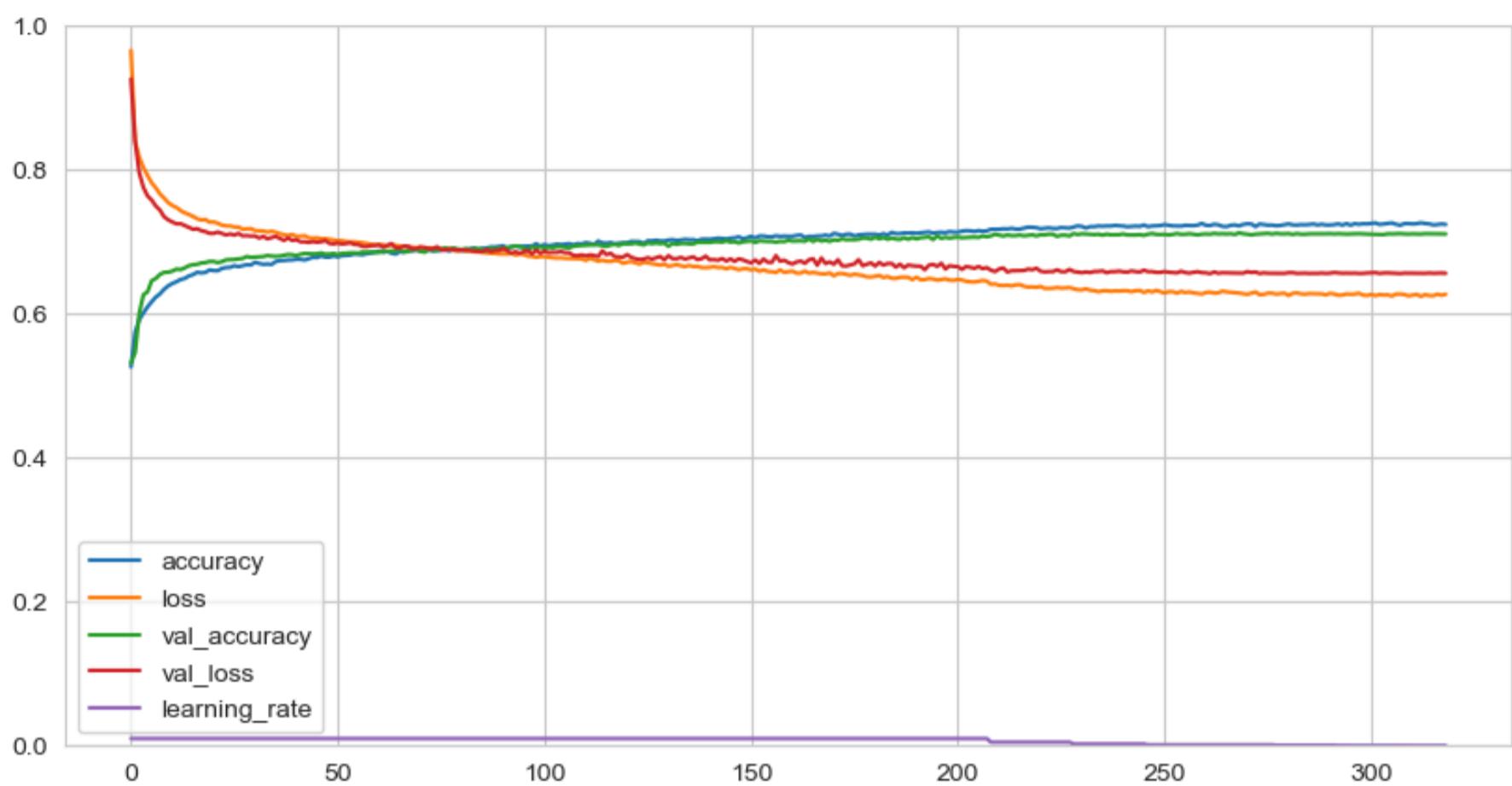
Train Set:

```
[[16183 4525 1323]
 [ 4491 29760 5958]
 [ 158 2739 10473]]
```

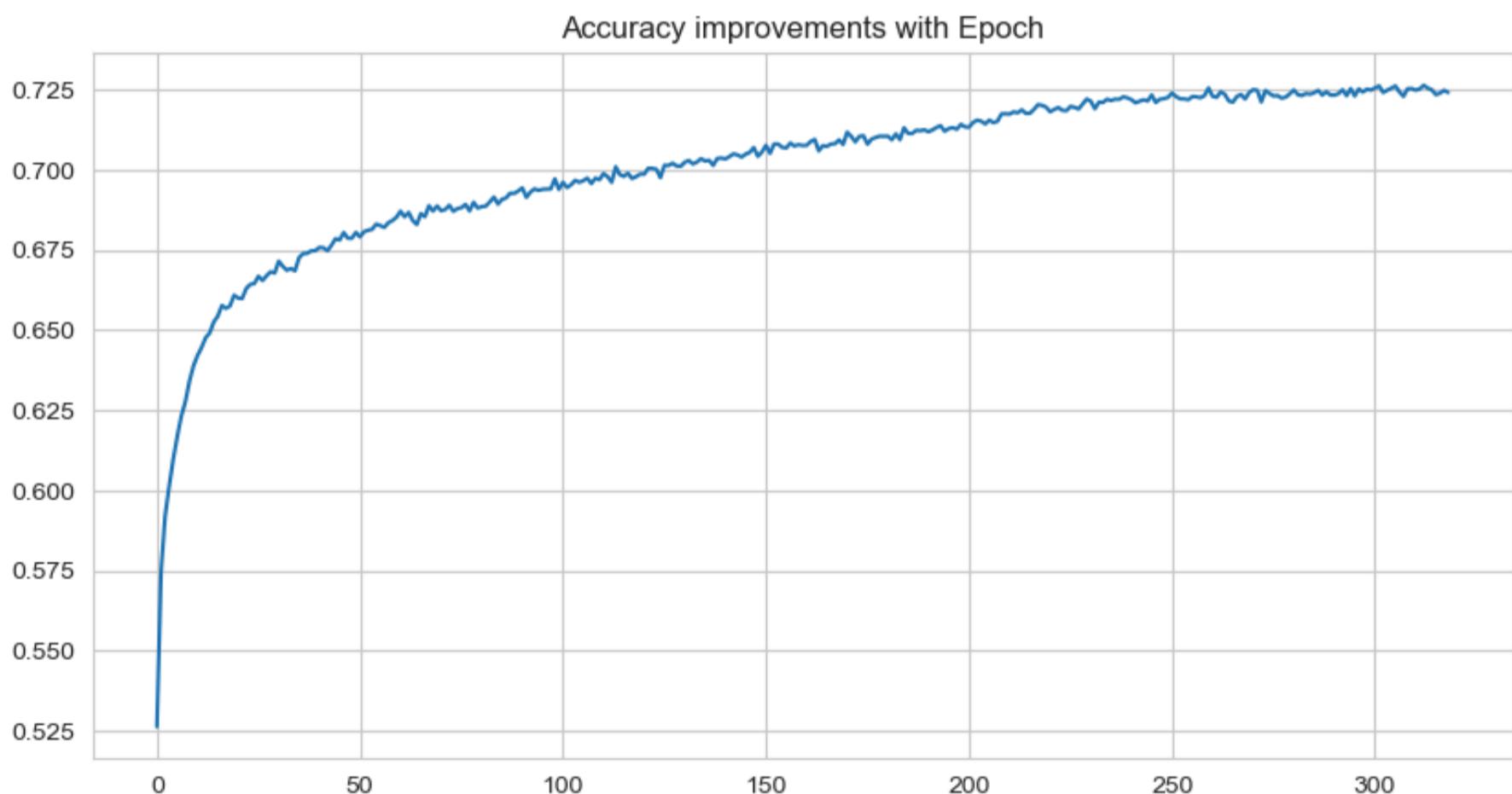
	precision	recall	f1-score	support
0	0.78	0.73	0.76	22031
1	0.80	0.74	0.77	40209
2	0.59	0.78	0.67	13370
accuracy			0.75	75610
macro avg	0.72	0.75	0.73	75610
weighted avg	0.76	0.75	0.75	75610

In [85]: pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)

```
plt.gca().set_ylim(0, 1)
plt.show()
```



```
In [86]: pd.DataFrame(history.history)["accuracy"].plot(figsize=(10, 5))
plt.title("Accuracy improvements with Epoch")
plt.show()
```



```
In [230...]: # Save the Model

from tensorflow.keras.models import load_model

# Save the model to 'model.h5' file
ann8.save('ann8_model.h5')

# To Load the model later for further use
loaded_ann8 = load_model('ann8_model.h5')
```

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)` . This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')` .
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.

ANN-8 Model Summary:

- **(Dense) layers:** 5 / **Neurons:** 256-128-128-64 / **Dropout:** 30-30-30-25% / **Learning Rate:** Initially 0.01, adjusted to 0.00015625 / **Batch Size:** 512 / **Epochs:** 600 / **Early Stop (val_accuracy):** 50
- **Accuracy:** 0.75 / **Val_Accuracy:** 0.71 / **Loss:** 0.5751 / **Val_Loss:** 0.6564 / **Train Recall (Class 2):** 0.78 / **Test Recall (Class 2):** 0.74

Improvements:

- Strong training performance with effective control of overfitting through balanced dropout and batch normalization.
- Consistent validation metrics show the model's good generalization ability.

No Improvement:

- There is a gap between training and validation performance, indicating potential areas for further enhancement to boost generalization.

Got Worse:

- The difference in performance metrics between training and validation suggests possible mild overfitting, pointing to a need for additional tuning in model complexity or further regularization.

ANN-8 Model with SMOTE (%80)

```
In [27]: print('Credi_Score Classes before Smote: ', y_train.value_counts())
print('\nCredi_Score Classes after Smote: ', y_train_smote.value_counts())
```

```
Credi_Score Classes before Smote: Credit_Score
1    40209
0    22031
2    13370
Name: count, dtype: int64
```

```
Credi_Score Classes after Smote: Credit_Score
1    40209
2    40209
0    40209
Name: count, dtype: int64
```

```
In [95]: # ANN-8 with Smote Model Architecture:
ann8_smote = Sequential([
    Dense(256, input_dim=X_train_smote.shape[1], activation='relu'), # Reduced neurons
    BatchNormalization(),
    Dropout(0.3),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.25),
    Dense(3, activation='softmax')
])

# Compiling the Model:
ann8_smote.compile(optimizer=SGD(learning_rate=0.01, momentum=0.9), # Using SGD with momentum
                    loss='sparse_categorical_crossentropy',
                    metrics=['accuracy'])

# Early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                                patience=50, # Slightly reduced patience
                                mode="auto",
                                restore_best_weights=True)

# ReduceLROnPlateau callback
reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                              factor=0.5, # Learning rate reduced by half if no improvement
                              patience=15, # Adjusted patience
                              min_lr=1e-6,
                              verbose=1)
```

```
# Train the model
history = ann8_smote.fit(
    x=X_train_smote,
    y=y_train_smote.values,
    validation_data=(X_val, y_val),
    batch_size=512,
    epochs=600,
    verbose=1,
    callbacks=[early_stopping, reduce_lr]) # Using callbacks
```

Epoch 1/600
236/236 4s 8ms/step - accuracy: 0.5402 - loss: 1.0779 - val_accuracy: 0.5981 - val_loss: 0.8806 - learning_rate: 0.0100
Epoch 2/600
236/236 1s 5ms/step - accuracy: 0.6440 - loss: 0.8342 - val_accuracy: 0.5705 - val_loss: 0.8656 - learning_rate: 0.0100
Epoch 3/600
236/236 1s 5ms/step - accuracy: 0.6630 - loss: 0.8064 - val_accuracy: 0.5784 - val_loss: 0.8773 - learning_rate: 0.0100
Epoch 4/600
236/236 1s 5ms/step - accuracy: 0.6790 - loss: 0.7865 - val_accuracy: 0.6143 - val_loss: 0.8465 - learning_rate: 0.0100
Epoch 5/600
236/236 1s 5ms/step - accuracy: 0.6900 - loss: 0.7705 - val_accuracy: 0.6268 - val_loss: 0.8519 - learning_rate: 0.0100
Epoch 6/600
236/236 1s 5ms/step - accuracy: 0.6984 - loss: 0.7553 - val_accuracy: 0.6385 - val_loss: 0.8216 - learning_rate: 0.0100
Epoch 7/600
236/236 1s 5ms/step - accuracy: 0.7018 - loss: 0.7498 - val_accuracy: 0.6389 - val_loss: 0.8192 - learning_rate: 0.0100
Epoch 8/600
236/236 1s 5ms/step - accuracy: 0.7047 - loss: 0.7430 - val_accuracy: 0.6349 - val_loss: 0.8356 - learning_rate: 0.0100
Epoch 9/600
236/236 1s 5ms/step - accuracy: 0.7054 - loss: 0.7393 - val_accuracy: 0.6397 - val_loss: 0.8180 - learning_rate: 0.0100
Epoch 10/600
236/236 1s 5ms/step - accuracy: 0.7051 - loss: 0.7353 - val_accuracy: 0.6396 - val_loss: 0.8077 - learning_rate: 0.0100
Epoch 11/600
236/236 1s 5ms/step - accuracy: 0.7092 - loss: 0.7273 - val_accuracy: 0.6481 - val_loss: 0.7973 - learning_rate: 0.0100
Epoch 12/600
236/236 1s 6ms/step - accuracy: 0.7120 - loss: 0.7210 - val_accuracy: 0.6424 - val_loss: 0.8163 - learning_rate: 0.0100
Epoch 13/600
236/236 1s 5ms/step - accuracy: 0.7111 - loss: 0.7233 - val_accuracy: 0.6447 - val_loss: 0.8162 - learning_rate: 0.0100
Epoch 14/600
236/236 1s 5ms/step - accuracy: 0.7094 - loss: 0.7203 - val_accuracy: 0.6502 - val_loss: 0.8040 - learning_rate: 0.0100
Epoch 15/600
236/236 1s 5ms/step - accuracy: 0.7121 - loss: 0.7169 - val_accuracy: 0.6416 - val_loss: 0.8116 - learning_rate: 0.0100
Epoch 16/600
236/236 1s 5ms/step - accuracy: 0.7153 - loss: 0.7124 - val_accuracy: 0.6487 - val_loss: 0.7946 - learning_rate: 0.0100
Epoch 17/600
236/236 1s 6ms/step - accuracy: 0.7153 - loss: 0.7085 - val_accuracy: 0.6473 - val_loss: 0.8031 - learning_rate: 0.0100
Epoch 18/600
236/236 1s 5ms/step - accuracy: 0.7154 - loss: 0.7071 - val_accuracy: 0.6539 - val_loss: 0.8004 - learning_rate: 0.0100
Epoch 19/600
236/236 1s 5ms/step - accuracy: 0.7147 - loss: 0.7065 - val_accuracy: 0.6561 - val_loss: 0.7923 - learning_rate: 0.0100
Epoch 20/600
236/236 1s 5ms/step - accuracy: 0.7167 - loss: 0.7045 - val_accuracy: 0.6513 - val_loss: 0.8002 - learning_rate: 0.0100
Epoch 21/600
236/236 1s 5ms/step - accuracy: 0.7172 - loss: 0.7006 - val_accuracy: 0.6511 - val_loss: 0.7965 - learning_rate: 0.0100
Epoch 22/600
236/236 1s 6ms/step - accuracy: 0.7164 - loss: 0.7025 - val_accuracy: 0.6529 - val_loss: 0.7925 - learning_rate: 0.0100
Epoch 23/600
236/236 1s 6ms/step - accuracy: 0.7210 - loss: 0.6943 - val_accuracy: 0.6586 - val_loss: 0.7993 - learning_rate: 0.0100
Epoch 24/600
236/236 1s 5ms/step - accuracy: 0.7163 - loss: 0.6987 - val_accuracy: 0.6593 - val_loss: 0.7843 - learning_rate: 0.0100
Epoch 25/600
236/236 1s 5ms/step - accuracy: 0.7214 - loss: 0.6885 - val_accuracy: 0.6506 - val_loss: 0.7979 - learning_rate: 0.0100
Epoch 26/600

```
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7228 - loss: 0.6897 - val_accuracy: 0.6523 - val_loss: 0.7953 - learning_rate: 0.0100
Epoch 27/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7198 - loss: 0.6914 - val_accuracy: 0.6447 - val_loss: 0.8068 - learning_rate: 0.0100
Epoch 28/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7204 - loss: 0.6883 - val_accuracy: 0.6581 - val_loss: 0.7904 - learning_rate: 0.0100
Epoch 29/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7214 - loss: 0.6887 - val_accuracy: 0.6589 - val_loss: 0.7834 - learning_rate: 0.0100
Epoch 30/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7234 - loss: 0.6802 - val_accuracy: 0.6569 - val_loss: 0.7876 - learning_rate: 0.0100
Epoch 31/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7249 - loss: 0.6781 - val_accuracy: 0.6580 - val_loss: 0.7922 - learning_rate: 0.0100
Epoch 32/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7233 - loss: 0.6805 - val_accuracy: 0.6578 - val_loss: 0.7909 - learning_rate: 0.0100
Epoch 33/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7277 - loss: 0.6772 - val_accuracy: 0.6586 - val_loss: 0.7839 - learning_rate: 0.0100
Epoch 34/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7249 - loss: 0.6756 - val_accuracy: 0.6567 - val_loss: 0.7863 - learning_rate: 0.0100
Epoch 35/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7276 - loss: 0.6743 - val_accuracy: 0.6628 - val_loss: 0.7775 - learning_rate: 0.0100
Epoch 36/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7267 - loss: 0.6719 - val_accuracy: 0.6615 - val_loss: 0.7911 - learning_rate: 0.0100
Epoch 37/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7299 - loss: 0.6686 - val_accuracy: 0.6573 - val_loss: 0.7858 - learning_rate: 0.0100
Epoch 38/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7283 - loss: 0.6718 - val_accuracy: 0.6573 - val_loss: 0.7830 - learning_rate: 0.0100
Epoch 39/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7325 - loss: 0.6658 - val_accuracy: 0.6597 - val_loss: 0.7755 - learning_rate: 0.0100
Epoch 40/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7293 - loss: 0.6656 - val_accuracy: 0.6624 - val_loss: 0.7878 - learning_rate: 0.0100
Epoch 41/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7293 - loss: 0.6679 - val_accuracy: 0.6577 - val_loss: 0.7829 - learning_rate: 0.0100
Epoch 42/600
236/236 ━━━━━━━━ 2s 7ms/step - accuracy: 0.7306 - loss: 0.6643 - val_accuracy: 0.6625 - val_loss: 0.7749 - learning_rate: 0.0100
Epoch 43/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7319 - loss: 0.6602 - val_accuracy: 0.6647 - val_loss: 0.7705 - learning_rate: 0.0100
Epoch 44/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7326 - loss: 0.6623 - val_accuracy: 0.6595 - val_loss: 0.7696 - learning_rate: 0.0100
Epoch 45/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7330 - loss: 0.6604 - val_accuracy: 0.6566 - val_loss: 0.7855 - learning_rate: 0.0100
Epoch 46/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7335 - loss: 0.6590 - val_accuracy: 0.6626 - val_loss: 0.7704 - learning_rate: 0.0100
Epoch 47/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7314 - loss: 0.6578 - val_accuracy: 0.6608 - val_loss: 0.7728 - learning_rate: 0.0100
Epoch 48/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7303 - loss: 0.6607 - val_accuracy: 0.6585 - val_loss: 0.7867 - learning_rate: 0.0100
Epoch 49/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7342 - loss: 0.6572 - val_accuracy: 0.6646 - val_loss: 0.7763 - learning_rate: 0.0100
Epoch 50/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7333 - loss: 0.6546 - val_accuracy: 0.6540 - val_loss: 0.7904 - learning_rate: 0.0100
Epoch 51/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7340 - loss: 0.6549 - val_accuracy: 0.6612 - val_loss:
```

```
s: 0.7767 - learning_rate: 0.0100
Epoch 52/600
236/236 1s 5ms/step - accuracy: 0.7381 - loss: 0.6476 - val_accuracy: 0.6653 - val_loss
s: 0.7687 - learning_rate: 0.0100
Epoch 53/600
236/236 1s 5ms/step - accuracy: 0.7353 - loss: 0.6514 - val_accuracy: 0.6619 - val_loss
s: 0.7773 - learning_rate: 0.0100
Epoch 54/600
236/236 1s 6ms/step - accuracy: 0.7350 - loss: 0.6512 - val_accuracy: 0.6617 - val_loss
s: 0.7715 - learning_rate: 0.0100
Epoch 55/600
236/236 1s 6ms/step - accuracy: 0.7363 - loss: 0.6481 - val_accuracy: 0.6627 - val_loss
s: 0.7851 - learning_rate: 0.0100
Epoch 56/600
236/236 1s 5ms/step - accuracy: 0.7359 - loss: 0.6478 - val_accuracy: 0.6635 - val_loss
s: 0.7616 - learning_rate: 0.0100
Epoch 57/600
236/236 1s 6ms/step - accuracy: 0.7408 - loss: 0.6418 - val_accuracy: 0.6672 - val_loss
s: 0.7547 - learning_rate: 0.0100
Epoch 58/600
236/236 1s 6ms/step - accuracy: 0.7389 - loss: 0.6433 - val_accuracy: 0.6593 - val_loss
s: 0.7771 - learning_rate: 0.0100
Epoch 59/600
236/236 1s 5ms/step - accuracy: 0.7368 - loss: 0.6451 - val_accuracy: 0.6677 - val_loss
s: 0.7690 - learning_rate: 0.0100
Epoch 60/600
236/236 1s 5ms/step - accuracy: 0.7373 - loss: 0.6443 - val_accuracy: 0.6607 - val_loss
s: 0.7709 - learning_rate: 0.0100
Epoch 61/600
236/236 1s 5ms/step - accuracy: 0.7395 - loss: 0.6434 - val_accuracy: 0.6642 - val_loss
s: 0.7681 - learning_rate: 0.0100
Epoch 62/600
236/236 1s 5ms/step - accuracy: 0.7382 - loss: 0.6416 - val_accuracy: 0.6640 - val_loss
s: 0.7785 - learning_rate: 0.0100
Epoch 63/600
236/236 1s 6ms/step - accuracy: 0.7386 - loss: 0.6445 - val_accuracy: 0.6649 - val_loss
s: 0.7681 - learning_rate: 0.0100
Epoch 64/600
236/236 1s 5ms/step - accuracy: 0.7414 - loss: 0.6391 - val_accuracy: 0.6654 - val_loss
s: 0.7588 - learning_rate: 0.0100
Epoch 65/600
236/236 2s 7ms/step - accuracy: 0.7391 - loss: 0.6386 - val_accuracy: 0.6688 - val_loss
s: 0.7571 - learning_rate: 0.0100
Epoch 66/600
236/236 2s 5ms/step - accuracy: 0.7391 - loss: 0.6395 - val_accuracy: 0.6652 - val_loss
s: 0.7600 - learning_rate: 0.0100
Epoch 67/600
236/236 1s 5ms/step - accuracy: 0.7414 - loss: 0.6337 - val_accuracy: 0.6638 - val_loss
s: 0.7619 - learning_rate: 0.0100
Epoch 68/600
236/236 1s 5ms/step - accuracy: 0.7420 - loss: 0.6378 - val_accuracy: 0.6656 - val_loss
s: 0.7678 - learning_rate: 0.0100
Epoch 69/600
236/236 1s 5ms/step - accuracy: 0.7428 - loss: 0.6309 - val_accuracy: 0.6617 - val_loss
s: 0.7740 - learning_rate: 0.0100
Epoch 70/600
236/236 1s 5ms/step - accuracy: 0.7439 - loss: 0.6318 - val_accuracy: 0.6610 - val_loss
s: 0.7679 - learning_rate: 0.0100
Epoch 71/600
236/236 1s 6ms/step - accuracy: 0.7419 - loss: 0.6352 - val_accuracy: 0.6718 - val_loss
s: 0.7597 - learning_rate: 0.0100
Epoch 72/600
236/236 1s 5ms/step - accuracy: 0.7434 - loss: 0.6318 - val_accuracy: 0.6730 - val_loss
s: 0.7489 - learning_rate: 0.0100
Epoch 73/600
236/236 1s 6ms/step - accuracy: 0.7440 - loss: 0.6311 - val_accuracy: 0.6647 - val_loss
s: 0.7679 - learning_rate: 0.0100
Epoch 74/600
236/236 1s 5ms/step - accuracy: 0.7453 - loss: 0.6280 - val_accuracy: 0.6717 - val_loss
s: 0.7566 - learning_rate: 0.0100
Epoch 75/600
236/236 1s 5ms/step - accuracy: 0.7450 - loss: 0.6271 - val_accuracy: 0.6690 - val_loss
s: 0.7540 - learning_rate: 0.0100
Epoch 76/600
236/236 1s 5ms/step - accuracy: 0.7467 - loss: 0.6243 - val_accuracy: 0.6697 - val_loss
s: 0.7568 - learning_rate: 0.0100
```

Epoch 77/600
236/236 2s 6ms/step - accuracy: 0.7455 - loss: 0.6260 - val_accuracy: 0.6700 - val_loss: 0.7474 - learning_rate: 0.0100
Epoch 78/600
236/236 1s 6ms/step - accuracy: 0.7465 - loss: 0.6244 - val_accuracy: 0.6683 - val_loss: 0.7568 - learning_rate: 0.0100
Epoch 79/600
236/236 1s 5ms/step - accuracy: 0.7446 - loss: 0.6257 - val_accuracy: 0.6710 - val_loss: 0.7564 - learning_rate: 0.0100
Epoch 80/600
236/236 1s 5ms/step - accuracy: 0.7480 - loss: 0.6203 - val_accuracy: 0.6715 - val_loss: 0.7490 - learning_rate: 0.0100
Epoch 81/600
236/236 1s 6ms/step - accuracy: 0.7453 - loss: 0.6257 - val_accuracy: 0.6689 - val_loss: 0.7493 - learning_rate: 0.0100
Epoch 82/600
236/236 1s 5ms/step - accuracy: 0.7494 - loss: 0.6200 - val_accuracy: 0.6734 - val_loss: 0.7595 - learning_rate: 0.0100
Epoch 83/600
236/236 1s 6ms/step - accuracy: 0.7455 - loss: 0.6259 - val_accuracy: 0.6712 - val_loss: 0.7463 - learning_rate: 0.0100
Epoch 84/600
236/236 1s 5ms/step - accuracy: 0.7475 - loss: 0.6205 - val_accuracy: 0.6750 - val_loss: 0.7434 - learning_rate: 0.0100
Epoch 85/600
236/236 1s 5ms/step - accuracy: 0.7487 - loss: 0.6197 - val_accuracy: 0.6697 - val_loss: 0.7733 - learning_rate: 0.0100
Epoch 86/600
236/236 1s 5ms/step - accuracy: 0.7469 - loss: 0.6199 - val_accuracy: 0.6736 - val_loss: 0.7529 - learning_rate: 0.0100
Epoch 87/600
236/236 1s 6ms/step - accuracy: 0.7497 - loss: 0.6173 - val_accuracy: 0.6731 - val_loss: 0.7472 - learning_rate: 0.0100
Epoch 88/600
236/236 2s 7ms/step - accuracy: 0.7515 - loss: 0.6136 - val_accuracy: 0.6632 - val_loss: 0.7699 - learning_rate: 0.0100
Epoch 89/600
236/236 2s 6ms/step - accuracy: 0.7506 - loss: 0.6164 - val_accuracy: 0.6721 - val_loss: 0.7532 - learning_rate: 0.0100
Epoch 90/600
236/236 1s 6ms/step - accuracy: 0.7493 - loss: 0.6142 - val_accuracy: 0.6719 - val_loss: 0.7463 - learning_rate: 0.0100
Epoch 91/600
236/236 1s 6ms/step - accuracy: 0.7488 - loss: 0.6180 - val_accuracy: 0.6672 - val_loss: 0.7481 - learning_rate: 0.0100
Epoch 92/600
236/236 1s 6ms/step - accuracy: 0.7504 - loss: 0.6127 - val_accuracy: 0.6688 - val_loss: 0.7541 - learning_rate: 0.0100
Epoch 93/600
236/236 1s 6ms/step - accuracy: 0.7518 - loss: 0.6105 - val_accuracy: 0.6707 - val_loss: 0.7532 - learning_rate: 0.0100
Epoch 94/600
236/236 2s 6ms/step - accuracy: 0.7527 - loss: 0.6103 - val_accuracy: 0.6750 - val_loss: 0.7395 - learning_rate: 0.0100
Epoch 95/600
236/236 1s 6ms/step - accuracy: 0.7486 - loss: 0.6130 - val_accuracy: 0.6724 - val_loss: 0.7451 - learning_rate: 0.0100
Epoch 96/600
236/236 2s 7ms/step - accuracy: 0.7528 - loss: 0.6091 - val_accuracy: 0.6723 - val_loss: 0.7536 - learning_rate: 0.0100
Epoch 97/600
236/236 2s 8ms/step - accuracy: 0.7520 - loss: 0.6125 - val_accuracy: 0.6770 - val_loss: 0.7400 - learning_rate: 0.0100
Epoch 98/600
236/236 2s 6ms/step - accuracy: 0.7535 - loss: 0.6068 - val_accuracy: 0.6726 - val_loss: 0.7422 - learning_rate: 0.0100
Epoch 99/600
236/236 2s 7ms/step - accuracy: 0.7504 - loss: 0.6103 - val_accuracy: 0.6708 - val_loss: 0.7497 - learning_rate: 0.0100
Epoch 100/600
236/236 2s 7ms/step - accuracy: 0.7509 - loss: 0.6095 - val_accuracy: 0.6751 - val_loss: 0.7417 - learning_rate: 0.0100
Epoch 101/600
236/236 1s 6ms/step - accuracy: 0.7528 - loss: 0.6080 - val_accuracy: 0.6728 - val_loss: 0.7549 - learning_rate: 0.0100
Epoch 102/600

```
236/236 ━━━━━━━━━━ 3s 6ms/step - accuracy: 0.7521 - loss: 0.6098 - val_accuracy: 0.6823 - val_loss: 0.7301 - learning_rate: 0.0100
Epoch 103/600
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7526 - loss: 0.6040 - val_accuracy: 0.6734 - val_loss: 0.7537 - learning_rate: 0.0100
Epoch 104/600
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7538 - loss: 0.6039 - val_accuracy: 0.6764 - val_loss: 0.7482 - learning_rate: 0.0100
Epoch 105/600
236/236 ━━━━━━━━━━ 1s 6ms/step - accuracy: 0.7549 - loss: 0.6031 - val_accuracy: 0.6790 - val_loss: 0.7476 - learning_rate: 0.0100
Epoch 106/600
236/236 ━━━━━━━━━━ 1s 6ms/step - accuracy: 0.7531 - loss: 0.6032 - val_accuracy: 0.6765 - val_loss: 0.7347 - learning_rate: 0.0100
Epoch 107/600
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7532 - loss: 0.6029 - val_accuracy: 0.6753 - val_loss: 0.7555 - learning_rate: 0.0100
Epoch 108/600
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7527 - loss: 0.6048 - val_accuracy: 0.6672 - val_loss: 0.7548 - learning_rate: 0.0100
Epoch 109/600
236/236 ━━━━━━━━━━ 1s 6ms/step - accuracy: 0.7531 - loss: 0.6059 - val_accuracy: 0.6720 - val_loss: 0.7492 - learning_rate: 0.0100
Epoch 110/600
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7555 - loss: 0.5976 - val_accuracy: 0.6756 - val_loss: 0.7500 - learning_rate: 0.0100
Epoch 111/600
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7533 - loss: 0.6052 - val_accuracy: 0.6782 - val_loss: 0.7462 - learning_rate: 0.0100
Epoch 112/600
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7569 - loss: 0.5987 - val_accuracy: 0.6721 - val_loss: 0.7367 - learning_rate: 0.0100
Epoch 113/600
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7572 - loss: 0.5995 - val_accuracy: 0.6693 - val_loss: 0.7503 - learning_rate: 0.0100
Epoch 114/600
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7580 - loss: 0.5975 - val_accuracy: 0.6771 - val_loss: 0.7428 - learning_rate: 0.0100
Epoch 115/600
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7541 - loss: 0.6025 - val_accuracy: 0.6728 - val_loss: 0.7505 - learning_rate: 0.0100
Epoch 116/600
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7569 - loss: 0.5959 - val_accuracy: 0.6790 - val_loss: 0.7353 - learning_rate: 0.0100
Epoch 117/600
233/236 ━━━━━━ 0s 5ms/step - accuracy: 0.7583 - loss: 0.5964
Epoch 117: ReduceLROnPlateau reducing learning rate to 0.004999999888241291.
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7582 - loss: 0.5964 - val_accuracy: 0.6799 - val_loss: 0.7336 - learning_rate: 0.0100
Epoch 118/600
236/236 ━━━━━━━━━━ 1s 6ms/step - accuracy: 0.7556 - loss: 0.6000 - val_accuracy: 0.6768 - val_loss: 0.7451 - learning_rate: 0.0050
Epoch 119/600
236/236 ━━━━━━━━━━ 1s 6ms/step - accuracy: 0.7579 - loss: 0.5924 - val_accuracy: 0.6801 - val_loss: 0.7399 - learning_rate: 0.0050
Epoch 120/600
236/236 ━━━━━━━━━━ 1s 6ms/step - accuracy: 0.7583 - loss: 0.5945 - val_accuracy: 0.6826 - val_loss: 0.7312 - learning_rate: 0.0050
Epoch 121/600
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7619 - loss: 0.5880 - val_accuracy: 0.6761 - val_loss: 0.7450 - learning_rate: 0.0050
Epoch 122/600
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7610 - loss: 0.5876 - val_accuracy: 0.6810 - val_loss: 0.7350 - learning_rate: 0.0050
Epoch 123/600
236/236 ━━━━━━━━━━ 1s 6ms/step - accuracy: 0.7597 - loss: 0.5892 - val_accuracy: 0.6796 - val_loss: 0.7461 - learning_rate: 0.0050
Epoch 124/600
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7611 - loss: 0.5881 - val_accuracy: 0.6816 - val_loss: 0.7339 - learning_rate: 0.0050
Epoch 125/600
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7624 - loss: 0.5865 - val_accuracy: 0.6848 - val_loss: 0.7271 - learning_rate: 0.0050
Epoch 126/600
236/236 ━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7615 - loss: 0.5886 - val_accuracy: 0.6844 - val_loss: 0.7301 - learning_rate: 0.0050
```

```
Epoch 127/600
236/236 1s 5ms/step - accuracy: 0.7607 - loss: 0.5858 - val_accuracy: 0.6784 - val_loss: 0.7364 - learning_rate: 0.0050
Epoch 128/600
236/236 1s 5ms/step - accuracy: 0.7605 - loss: 0.5873 - val_accuracy: 0.6820 - val_loss: 0.7400 - learning_rate: 0.0050
Epoch 129/600
236/236 1s 5ms/step - accuracy: 0.7620 - loss: 0.5849 - val_accuracy: 0.6790 - val_loss: 0.7449 - learning_rate: 0.0050
Epoch 130/600
236/236 1s 5ms/step - accuracy: 0.7616 - loss: 0.5860 - val_accuracy: 0.6778 - val_loss: 0.7416 - learning_rate: 0.0050
Epoch 131/600
236/236 1s 5ms/step - accuracy: 0.7622 - loss: 0.5851 - val_accuracy: 0.6808 - val_loss: 0.7361 - learning_rate: 0.0050
Epoch 132/600
236/236 1s 5ms/step - accuracy: 0.7628 - loss: 0.5825 - val_accuracy: 0.6825 - val_loss: 0.7359 - learning_rate: 0.0050
Epoch 133/600
236/236 1s 5ms/step - accuracy: 0.7642 - loss: 0.5828 - val_accuracy: 0.6767 - val_loss: 0.7464 - learning_rate: 0.0050
Epoch 134/600
236/236 1s 5ms/step - accuracy: 0.7623 - loss: 0.5871 - val_accuracy: 0.6772 - val_loss: 0.7411 - learning_rate: 0.0050
Epoch 135/600
236/236 1s 5ms/step - accuracy: 0.7614 - loss: 0.5858 - val_accuracy: 0.6811 - val_loss: 0.7319 - learning_rate: 0.0050
Epoch 136/600
236/236 2s 7ms/step - accuracy: 0.7642 - loss: 0.5800 - val_accuracy: 0.6774 - val_loss: 0.7406 - learning_rate: 0.0050
Epoch 137/600
236/236 1s 5ms/step - accuracy: 0.7612 - loss: 0.5885 - val_accuracy: 0.6783 - val_loss: 0.7383 - learning_rate: 0.0050
Epoch 138/600
236/236 1s 5ms/step - accuracy: 0.7613 - loss: 0.5857 - val_accuracy: 0.6805 - val_loss: 0.7355 - learning_rate: 0.0050
Epoch 139/600
236/236 1s 5ms/step - accuracy: 0.7635 - loss: 0.5819 - val_accuracy: 0.6836 - val_loss: 0.7329 - learning_rate: 0.0050
Epoch 140/600
232/236 0s 5ms/step - accuracy: 0.7637 - loss: 0.5822
Epoch 140: ReduceLROnPlateau reducing learning rate to 0.0024999999441206455.
236/236 1s 6ms/step - accuracy: 0.7637 - loss: 0.5823 - val_accuracy: 0.6783 - val_loss: 0.7318 - learning_rate: 0.0050
Epoch 141/600
236/236 1s 5ms/step - accuracy: 0.7637 - loss: 0.5801 - val_accuracy: 0.6842 - val_loss: 0.7316 - learning_rate: 0.0025
Epoch 142/600
236/236 1s 5ms/step - accuracy: 0.7656 - loss: 0.5801 - val_accuracy: 0.6816 - val_loss: 0.7347 - learning_rate: 0.0025
Epoch 143/600
236/236 1s 5ms/step - accuracy: 0.7656 - loss: 0.5781 - val_accuracy: 0.6833 - val_loss: 0.7297 - learning_rate: 0.0025
Epoch 144/600
236/236 1s 5ms/step - accuracy: 0.7642 - loss: 0.5801 - val_accuracy: 0.6808 - val_loss: 0.7342 - learning_rate: 0.0025
Epoch 145/600
236/236 1s 5ms/step - accuracy: 0.7645 - loss: 0.5791 - val_accuracy: 0.6836 - val_loss: 0.7326 - learning_rate: 0.0025
Epoch 146/600
236/236 1s 5ms/step - accuracy: 0.7666 - loss: 0.5777 - val_accuracy: 0.6795 - val_loss: 0.7366 - learning_rate: 0.0025
Epoch 147/600
236/236 1s 5ms/step - accuracy: 0.7658 - loss: 0.5744 - val_accuracy: 0.6816 - val_loss: 0.7357 - learning_rate: 0.0025
Epoch 148/600
236/236 1s 5ms/step - accuracy: 0.7637 - loss: 0.5790 - val_accuracy: 0.6840 - val_loss: 0.7312 - learning_rate: 0.0025
Epoch 149/600
236/236 1s 5ms/step - accuracy: 0.7649 - loss: 0.5780 - val_accuracy: 0.6801 - val_loss: 0.7381 - learning_rate: 0.0025
Epoch 150/600
236/236 1s 5ms/step - accuracy: 0.7682 - loss: 0.5755 - val_accuracy: 0.6813 - val_loss: 0.7367 - learning_rate: 0.0025
Epoch 151/600
236/236 1s 5ms/step - accuracy: 0.7664 - loss: 0.5749 - val_accuracy: 0.6816 - val_loss:
```

```
s: 0.7325 - learning_rate: 0.0025
Epoch 152/600
236/236 1s 5ms/step - accuracy: 0.7671 - loss: 0.5757 - val_accuracy: 0.6842 - val_loss
s: 0.7224 - learning_rate: 0.0025
Epoch 153/600
236/236 1s 5ms/step - accuracy: 0.7660 - loss: 0.5782 - val_accuracy: 0.6809 - val_loss
s: 0.7398 - learning_rate: 0.0025
Epoch 154/600
236/236 1s 5ms/step - accuracy: 0.7652 - loss: 0.5768 - val_accuracy: 0.6808 - val_loss
s: 0.7408 - learning_rate: 0.0025
Epoch 155/600
236/236 1s 5ms/step - accuracy: 0.7682 - loss: 0.5727 - val_accuracy: 0.6844 - val_loss
s: 0.7263 - learning_rate: 0.0025
Epoch 156/600
236/236 1s 5ms/step - accuracy: 0.7648 - loss: 0.5754 - val_accuracy: 0.6843 - val_loss
s: 0.7297 - learning_rate: 0.0025
Epoch 157/600
236/236 1s 5ms/step - accuracy: 0.7660 - loss: 0.5767 - val_accuracy: 0.6802 - val_loss
s: 0.7336 - learning_rate: 0.0025
Epoch 158/600
236/236 1s 5ms/step - accuracy: 0.7666 - loss: 0.5778 - val_accuracy: 0.6815 - val_loss
s: 0.7375 - learning_rate: 0.0025
Epoch 159/600
236/236 1s 5ms/step - accuracy: 0.7646 - loss: 0.5746 - val_accuracy: 0.6859 - val_loss
s: 0.7279 - learning_rate: 0.0025
Epoch 160/600
236/236 1s 5ms/step - accuracy: 0.7656 - loss: 0.5755 - val_accuracy: 0.6830 - val_loss
s: 0.7307 - learning_rate: 0.0025
Epoch 161/600
236/236 1s 5ms/step - accuracy: 0.7661 - loss: 0.5747 - val_accuracy: 0.6848 - val_loss
s: 0.7271 - learning_rate: 0.0025
Epoch 162/600
236/236 1s 5ms/step - accuracy: 0.7635 - loss: 0.5808 - val_accuracy: 0.6829 - val_loss
s: 0.7355 - learning_rate: 0.0025
Epoch 163/600
236/236 1s 5ms/step - accuracy: 0.7667 - loss: 0.5740 - val_accuracy: 0.6818 - val_loss
s: 0.7344 - learning_rate: 0.0025
Epoch 164/600
236/236 1s 5ms/step - accuracy: 0.7652 - loss: 0.5772 - val_accuracy: 0.6851 - val_loss
s: 0.7277 - learning_rate: 0.0025
Epoch 165/600
236/236 1s 5ms/step - accuracy: 0.7654 - loss: 0.5764 - val_accuracy: 0.6830 - val_loss
s: 0.7340 - learning_rate: 0.0025
Epoch 166/600
236/236 1s 5ms/step - accuracy: 0.7665 - loss: 0.5746 - val_accuracy: 0.6829 - val_loss
s: 0.7320 - learning_rate: 0.0025
Epoch 167/600
226/236 0s 5ms/step - accuracy: 0.7689 - loss: 0.5689
Epoch 167: ReduceLROnPlateau reducing learning rate to 0.0012499999720603228.
236/236 1s 5ms/step - accuracy: 0.7688 - loss: 0.5692 - val_accuracy: 0.6862 - val_loss
s: 0.7235 - learning_rate: 0.0025
Epoch 168/600
236/236 1s 5ms/step - accuracy: 0.7670 - loss: 0.5741 - val_accuracy: 0.6844 - val_loss
s: 0.7317 - learning_rate: 0.0012
Epoch 169/600
236/236 1s 5ms/step - accuracy: 0.7675 - loss: 0.5738 - val_accuracy: 0.6823 - val_loss
s: 0.7345 - learning_rate: 0.0012
Epoch 170/600
236/236 1s 5ms/step - accuracy: 0.7685 - loss: 0.5725 - val_accuracy: 0.6843 - val_loss
s: 0.7319 - learning_rate: 0.0012
Epoch 171/600
236/236 1s 5ms/step - accuracy: 0.7682 - loss: 0.5707 - val_accuracy: 0.6820 - val_loss
s: 0.7339 - learning_rate: 0.0012
Epoch 172/600
236/236 1s 5ms/step - accuracy: 0.7671 - loss: 0.5715 - val_accuracy: 0.6838 - val_loss
s: 0.7312 - learning_rate: 0.0012
Epoch 173/600
236/236 1s 5ms/step - accuracy: 0.7661 - loss: 0.5735 - val_accuracy: 0.6831 - val_loss
s: 0.7308 - learning_rate: 0.0012
Epoch 174/600
236/236 1s 6ms/step - accuracy: 0.7676 - loss: 0.5740 - val_accuracy: 0.6820 - val_loss
s: 0.7356 - learning_rate: 0.0012
Epoch 175/600
236/236 1s 5ms/step - accuracy: 0.7715 - loss: 0.5660 - val_accuracy: 0.6836 - val_loss
s: 0.7299 - learning_rate: 0.0012
Epoch 176/600
```

```
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7656 - loss: 0.5747 - val_accuracy: 0.6838 - val_loss: 0.7305 - learning_rate: 0.0012
Epoch 177/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7678 - loss: 0.5707 - val_accuracy: 0.6822 - val_loss: 0.7369 - learning_rate: 0.0012
Epoch 178/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7657 - loss: 0.5738 - val_accuracy: 0.6831 - val_loss: 0.7338 - learning_rate: 0.0012
Epoch 179/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7687 - loss: 0.5684 - val_accuracy: 0.6830 - val_loss: 0.7288 - learning_rate: 0.0012
Epoch 180/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7659 - loss: 0.5700 - val_accuracy: 0.6826 - val_loss: 0.7345 - learning_rate: 0.0012
Epoch 181/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7707 - loss: 0.5673 - val_accuracy: 0.6838 - val_loss: 0.7333 - learning_rate: 0.0012
Epoch 182/600
227/236 ━━━━━━ 0s 6ms/step - accuracy: 0.7688 - loss: 0.5661
Epoch 182: ReduceLROnPlateau reducing learning rate to 0.0006249999860301614.
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7688 - loss: 0.5663 - val_accuracy: 0.6849 - val_loss: 0.7269 - learning_rate: 0.0012
Epoch 183/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7710 - loss: 0.5691 - val_accuracy: 0.6847 - val_loss: 0.7328 - learning_rate: 6.2500e-04
Epoch 184/600
236/236 ━━━━━━━━ 2s 6ms/step - accuracy: 0.7698 - loss: 0.5680 - val_accuracy: 0.6859 - val_loss: 0.7308 - learning_rate: 6.2500e-04
Epoch 185/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7671 - loss: 0.5731 - val_accuracy: 0.6843 - val_loss: 0.7322 - learning_rate: 6.2500e-04
Epoch 186/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7672 - loss: 0.5707 - val_accuracy: 0.6858 - val_loss: 0.7309 - learning_rate: 6.2500e-04
Epoch 187/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7687 - loss: 0.5697 - val_accuracy: 0.6844 - val_loss: 0.7322 - learning_rate: 6.2500e-04
Epoch 188/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7667 - loss: 0.5703 - val_accuracy: 0.6850 - val_loss: 0.7311 - learning_rate: 6.2500e-04
Epoch 189/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7682 - loss: 0.5742 - val_accuracy: 0.6845 - val_loss: 0.7305 - learning_rate: 6.2500e-04
Epoch 190/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7699 - loss: 0.5694 - val_accuracy: 0.6849 - val_loss: 0.7299 - learning_rate: 6.2500e-04
Epoch 191/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7682 - loss: 0.5696 - val_accuracy: 0.6840 - val_loss: 0.7305 - learning_rate: 6.2500e-04
Epoch 192/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7685 - loss: 0.5698 - val_accuracy: 0.6856 - val_loss: 0.7301 - learning_rate: 6.2500e-04
Epoch 193/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7697 - loss: 0.5678 - val_accuracy: 0.6855 - val_loss: 0.7291 - learning_rate: 6.2500e-04
Epoch 194/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7728 - loss: 0.5637 - val_accuracy: 0.6846 - val_loss: 0.7294 - learning_rate: 6.2500e-04
Epoch 195/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7718 - loss: 0.5662 - val_accuracy: 0.6833 - val_loss: 0.7316 - learning_rate: 6.2500e-04
Epoch 196/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7664 - loss: 0.5705 - val_accuracy: 0.6856 - val_loss: 0.7291 - learning_rate: 6.2500e-04
Epoch 197/600
234/236 ━━━━━━ 0s 5ms/step - accuracy: 0.7685 - loss: 0.5665
Epoch 197: ReduceLROnPlateau reducing learning rate to 0.0003124999930150807.
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7684 - loss: 0.5666 - val_accuracy: 0.6838 - val_loss: 0.7306 - learning_rate: 6.2500e-04
Epoch 198/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7708 - loss: 0.5643 - val_accuracy: 0.6855 - val_loss: 0.7295 - learning_rate: 3.1250e-04
Epoch 199/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7704 - loss: 0.5658 - val_accuracy: 0.6856 - val_loss: 0.7288 - learning_rate: 3.1250e-04
Epoch 200/600
```

```
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7701 - loss: 0.5680 - val_accuracy: 0.6852 - val_loss: 0.7302 - learning_rate: 3.1250e-04
Epoch 201/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7694 - loss: 0.5668 - val_accuracy: 0.6848 - val_loss: 0.7303 - learning_rate: 3.1250e-04
Epoch 202/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7678 - loss: 0.5692 - val_accuracy: 0.6857 - val_loss: 0.7291 - learning_rate: 3.1250e-04
Epoch 203/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7692 - loss: 0.5712 - val_accuracy: 0.6838 - val_loss: 0.7304 - learning_rate: 3.1250e-04
Epoch 204/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7663 - loss: 0.5739 - val_accuracy: 0.6844 - val_loss: 0.7275 - learning_rate: 3.1250e-04
Epoch 205/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7711 - loss: 0.5662 - val_accuracy: 0.6850 - val_loss: 0.7283 - learning_rate: 3.1250e-04
Epoch 206/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7688 - loss: 0.5687 - val_accuracy: 0.6853 - val_loss: 0.7307 - learning_rate: 3.1250e-04
Epoch 207/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7699 - loss: 0.5684 - val_accuracy: 0.6843 - val_loss: 0.7303 - learning_rate: 3.1250e-04
Epoch 208/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7675 - loss: 0.5710 - val_accuracy: 0.6857 - val_loss: 0.7298 - learning_rate: 3.1250e-04
Epoch 209/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7681 - loss: 0.5696 - val_accuracy: 0.6842 - val_loss: 0.7304 - learning_rate: 3.1250e-04
Epoch 210/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7666 - loss: 0.5721 - val_accuracy: 0.6863 - val_loss: 0.7286 - learning_rate: 3.1250e-04
Epoch 211/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7697 - loss: 0.5667 - val_accuracy: 0.6846 - val_loss: 0.7306 - learning_rate: 3.1250e-04
Epoch 212/600
236/236 ━━━━ 0s 5ms/step - accuracy: 0.7694 - loss: 0.5676
Epoch 212: ReduceLROnPlateau reducing learning rate to 0.00015624999650754035.
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7694 - loss: 0.5677 - val_accuracy: 0.6849 - val_loss: 0.7294 - learning_rate: 3.1250e-04
Epoch 213/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7686 - loss: 0.5688 - val_accuracy: 0.6848 - val_loss: 0.7298 - learning_rate: 1.5625e-04
Epoch 214/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7725 - loss: 0.5630 - val_accuracy: 0.6851 - val_loss: 0.7293 - learning_rate: 1.5625e-04
Epoch 215/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7684 - loss: 0.5704 - val_accuracy: 0.6850 - val_loss: 0.7297 - learning_rate: 1.5625e-04
Epoch 216/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7681 - loss: 0.5724 - val_accuracy: 0.6848 - val_loss: 0.7300 - learning_rate: 1.5625e-04
Epoch 217/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7711 - loss: 0.5668 - val_accuracy: 0.6848 - val_loss: 0.7282 - learning_rate: 1.5625e-04
Epoch 218/600
236/236 ━━━━━━━━ 1s 6ms/step - accuracy: 0.7710 - loss: 0.5672 - val_accuracy: 0.6844 - val_loss: 0.7293 - learning_rate: 1.5625e-04
Epoch 219/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7704 - loss: 0.5676 - val_accuracy: 0.6845 - val_loss: 0.7300 - learning_rate: 1.5625e-04
Epoch 220/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7726 - loss: 0.5657 - val_accuracy: 0.6842 - val_loss: 0.7305 - learning_rate: 1.5625e-04
Epoch 221/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7697 - loss: 0.5686 - val_accuracy: 0.6844 - val_loss: 0.7297 - learning_rate: 1.5625e-04
Epoch 222/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7670 - loss: 0.5706 - val_accuracy: 0.6847 - val_loss: 0.7302 - learning_rate: 1.5625e-04
Epoch 223/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7721 - loss: 0.5609 - val_accuracy: 0.6844 - val_loss: 0.7299 - learning_rate: 1.5625e-04
Epoch 224/600
236/236 ━━━━━━━━ 1s 5ms/step - accuracy: 0.7711 - loss: 0.5615 - val_accuracy: 0.6850 - val_loss: 0.7307 - learning_rate: 1.5625e-04
```

Epoch 225/600
236/236 1s 5ms/step - accuracy: 0.7702 - loss: 0.5668 - val_accuracy: 0.6847 - val_loss: 0.7312 - learning_rate: 1.5625e-04
Epoch 226/600
236/236 1s 5ms/step - accuracy: 0.7689 - loss: 0.5721 - val_accuracy: 0.6846 - val_loss: 0.7298 - learning_rate: 1.5625e-04
Epoch 227/600
226/236 0s 5ms/step - accuracy: 0.7692 - loss: 0.5667
Epoch 227: ReduceLROnPlateau reducing learning rate to 7.812499825377017e-05.
236/236 1s 6ms/step - accuracy: 0.7692 - loss: 0.5668 - val_accuracy: 0.6846 - val_loss: 0.7308 - learning_rate: 1.5625e-04
Epoch 228/600
236/236 1s 5ms/step - accuracy: 0.7706 - loss: 0.5678 - val_accuracy: 0.6848 - val_loss: 0.7298 - learning_rate: 7.8125e-05
Epoch 229/600
236/236 1s 5ms/step - accuracy: 0.7692 - loss: 0.5674 - val_accuracy: 0.6845 - val_loss: 0.7300 - learning_rate: 7.8125e-05
Epoch 230/600
236/236 1s 5ms/step - accuracy: 0.7706 - loss: 0.5690 - val_accuracy: 0.6844 - val_loss: 0.7302 - learning_rate: 7.8125e-05
Epoch 231/600
236/236 1s 5ms/step - accuracy: 0.7718 - loss: 0.5653 - val_accuracy: 0.6847 - val_loss: 0.7308 - learning_rate: 7.8125e-05
Epoch 232/600
236/236 1s 5ms/step - accuracy: 0.7680 - loss: 0.5718 - val_accuracy: 0.6852 - val_loss: 0.7301 - learning_rate: 7.8125e-05
Epoch 233/600
236/236 1s 5ms/step - accuracy: 0.7706 - loss: 0.5663 - val_accuracy: 0.6849 - val_loss: 0.7297 - learning_rate: 7.8125e-05
Epoch 234/600
236/236 1s 5ms/step - accuracy: 0.7715 - loss: 0.5646 - val_accuracy: 0.6851 - val_loss: 0.7299 - learning_rate: 7.8125e-05
Epoch 235/600
236/236 1s 5ms/step - accuracy: 0.7728 - loss: 0.5646 - val_accuracy: 0.6853 - val_loss: 0.7287 - learning_rate: 7.8125e-05
Epoch 236/600
236/236 1s 6ms/step - accuracy: 0.7713 - loss: 0.5639 - val_accuracy: 0.6850 - val_loss: 0.7297 - learning_rate: 7.8125e-05
Epoch 237/600
236/236 1s 5ms/step - accuracy: 0.7697 - loss: 0.5665 - val_accuracy: 0.6851 - val_loss: 0.7299 - learning_rate: 7.8125e-05
Epoch 238/600
236/236 1s 5ms/step - accuracy: 0.7698 - loss: 0.5677 - val_accuracy: 0.6848 - val_loss: 0.7303 - learning_rate: 7.8125e-05
Epoch 239/600
236/236 1s 5ms/step - accuracy: 0.7709 - loss: 0.5657 - val_accuracy: 0.6852 - val_loss: 0.7296 - learning_rate: 7.8125e-05
Epoch 240/600
236/236 1s 5ms/step - accuracy: 0.7693 - loss: 0.5668 - val_accuracy: 0.6847 - val_loss: 0.7304 - learning_rate: 7.8125e-05
Epoch 241/600
236/236 1s 5ms/step - accuracy: 0.7701 - loss: 0.5666 - val_accuracy: 0.6850 - val_loss: 0.7296 - learning_rate: 7.8125e-05
Epoch 242/600
233/236 0s 5ms/step - accuracy: 0.7716 - loss: 0.5633
Epoch 242: ReduceLROnPlateau reducing learning rate to 3.9062499126885086e-05.
236/236 1s 6ms/step - accuracy: 0.7715 - loss: 0.5633 - val_accuracy: 0.6851 - val_loss: 0.7300 - learning_rate: 7.8125e-05
Epoch 243/600
236/236 1s 5ms/step - accuracy: 0.7705 - loss: 0.5673 - val_accuracy: 0.6851 - val_loss: 0.7298 - learning_rate: 3.9062e-05
Epoch 244/600
236/236 1s 5ms/step - accuracy: 0.7680 - loss: 0.5674 - val_accuracy: 0.6845 - val_loss: 0.7307 - learning_rate: 3.9062e-05
Epoch 245/600
236/236 1s 5ms/step - accuracy: 0.7729 - loss: 0.5597 - val_accuracy: 0.6849 - val_loss: 0.7303 - learning_rate: 3.9062e-05
Epoch 246/600
236/236 1s 6ms/step - accuracy: 0.7697 - loss: 0.5699 - val_accuracy: 0.6849 - val_loss: 0.7306 - learning_rate: 3.9062e-05
Epoch 247/600
236/236 1s 5ms/step - accuracy: 0.7701 - loss: 0.5671 - val_accuracy: 0.6853 - val_loss: 0.7300 - learning_rate: 3.9062e-05
Epoch 248/600
236/236 1s 6ms/step - accuracy: 0.7701 - loss: 0.5655 - val_accuracy: 0.6850 - val_loss: 0.7302 - learning_rate: 3.9062e-05

```
Epoch 249/600
236/236 1s 6ms/step - accuracy: 0.7673 - loss: 0.5663 - val_accuracy: 0.6851 - val_loss: 0.7298 - learning_rate: 3.9062e-05
Epoch 250/600
236/236 1s 5ms/step - accuracy: 0.7708 - loss: 0.5654 - val_accuracy: 0.6850 - val_loss: 0.7297 - learning_rate: 3.9062e-05
Epoch 251/600
236/236 1s 5ms/step - accuracy: 0.7703 - loss: 0.5672 - val_accuracy: 0.6849 - val_loss: 0.7302 - learning_rate: 3.9062e-05
Epoch 252/600
236/236 1s 5ms/step - accuracy: 0.7713 - loss: 0.5666 - val_accuracy: 0.6849 - val_loss: 0.7307 - learning_rate: 3.9062e-05
Epoch 253/600
236/236 1s 5ms/step - accuracy: 0.7722 - loss: 0.5628 - val_accuracy: 0.6850 - val_loss: 0.7303 - learning_rate: 3.9062e-05
Epoch 254/600
236/236 1s 5ms/step - accuracy: 0.7704 - loss: 0.5672 - val_accuracy: 0.6849 - val_loss: 0.7300 - learning_rate: 3.9062e-05
Epoch 255/600
236/236 1s 5ms/step - accuracy: 0.7735 - loss: 0.5633 - val_accuracy: 0.6850 - val_loss: 0.7299 - learning_rate: 3.9062e-05
Epoch 256/600
236/236 1s 5ms/step - accuracy: 0.7749 - loss: 0.5638 - val_accuracy: 0.6854 - val_loss: 0.7301 - learning_rate: 3.9062e-05
Epoch 257/600
229/236 0s 5ms/step - accuracy: 0.7701 - loss: 0.5685
Epoch 257: ReduceLROnPlateau reducing learning rate to 1.9531249563442543e-05.
236/236 1s 5ms/step - accuracy: 0.7701 - loss: 0.5685 - val_accuracy: 0.6849 - val_loss: 0.7299 - learning_rate: 3.9062e-05
Epoch 258/600
236/236 1s 5ms/step - accuracy: 0.7693 - loss: 0.5663 - val_accuracy: 0.6850 - val_loss: 0.7299 - learning_rate: 1.9531e-05
Epoch 259/600
236/236 1s 5ms/step - accuracy: 0.7721 - loss: 0.5618 - val_accuracy: 0.6852 - val_loss: 0.7296 - learning_rate: 1.9531e-05
Epoch 260/600
236/236 1s 6ms/step - accuracy: 0.7708 - loss: 0.5661 - val_accuracy: 0.6851 - val_loss: 0.7302 - learning_rate: 1.9531e-05
```

```
In [98]: ann8_smote.evaluate(X_train_smote, y_train_smote)
```

```
3770/3770 3s 862us/step - accuracy: 0.7390 - loss: 0.5980
```

```
Out[98]: [0.48801127076148987, 0.8024322986602783]
```

```
In [97]: ann8_smote.summary()
```

```
Model: "sequential_12"
```

Layer (type)	Output Shape	Param #
dense_68 (Dense)	(None, 256)	11,520
batch_normalization_48 (BatchNormalization)	(None, 256)	1,024
dropout_56 (Dropout)	(None, 256)	0
dense_69 (Dense)	(None, 128)	32,896
batch_normalization_49 (BatchNormalization)	(None, 128)	512
dropout_57 (Dropout)	(None, 128)	0
dense_70 (Dense)	(None, 128)	16,512
batch_normalization_50 (BatchNormalization)	(None, 128)	512
dropout_58 (Dropout)	(None, 128)	0
dense_71 (Dense)	(None, 64)	8,256
batch_normalization_51 (BatchNormalization)	(None, 64)	256
dropout_59 (Dropout)	(None, 64)	0
dense_72 (Dense)	(None, 3)	195

Total params: 142,216 (555.54 KB)

Trainable params: 70,531 (275.51 KB)

Non-trainable params: 1,152 (4.50 KB)

Optimizer params: 70,533 (275.52 KB)

```
In [96]: eval_metric(ann8_smote, X_train_smote, y_train_smote, X_val, y_val)
```

3770/3770 ━━━━━━━━ 4s 935us/step
591/591 ━━━━━━ 1s 833us/step

Test Set:

```
[[4445 651 412]
 [2270 5777 2006]
 [ 131 460 2751]]
```

	precision	recall	f1-score	support
0	0.65	0.81	0.72	5508
1	0.84	0.57	0.68	10053
2	0.53	0.82	0.65	3342
accuracy			0.69	18903
macro avg	0.67	0.73	0.68	18903
weighted avg	0.73	0.69	0.69	18903

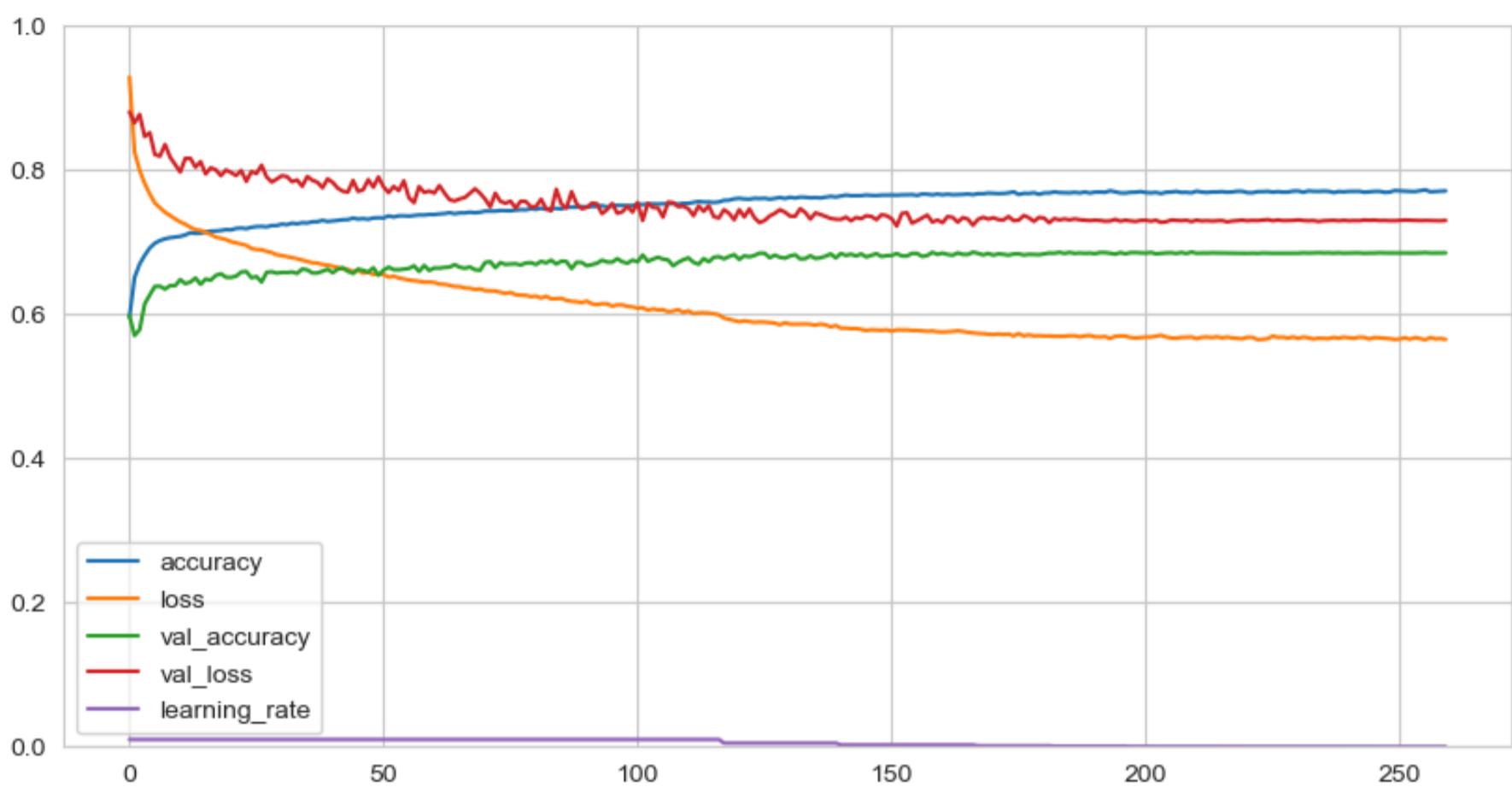
Train Set:

```
[[34626 3329 2254]
 [ 8473 24146 7590]
 [ 387 1799 38023]]
```

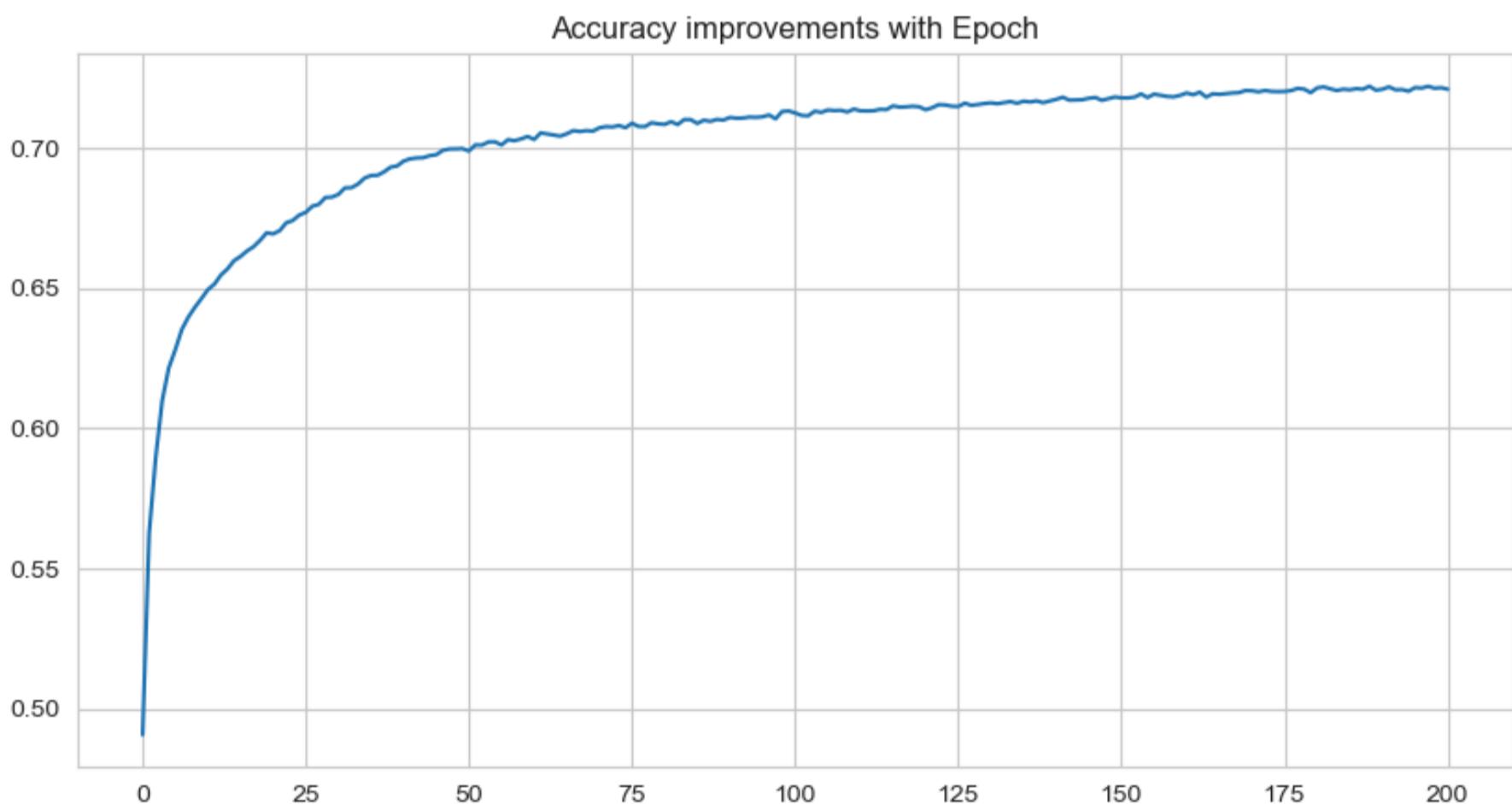
	precision	recall	f1-score	support
0	0.80	0.86	0.83	40209
1	0.82	0.60	0.70	40209
2	0.79	0.95	0.86	40209
accuracy			0.80	120627
macro avg	0.81	0.80	0.80	120627
weighted avg	0.81	0.80	0.80	120627

```
In [99]: pd.DataFrame(history.history).plot(figsize=(10,5))
plt.grid(True)
```

```
plt.gca().set_ylim(0, 1)
plt.show()
```



```
In [94]: pd.DataFrame(history.history)[“accuracy”].plot(figsize=(10, 5))
plt.title(“Accuracy improvements with Epoch”)
plt.show()
```



```
In [ ]: # Save the Model

from tensorflow.keras.models import load_model

# Save the model to 'model.h5' file
ann8_smote.save('ann8_smote_model.h5')

# To Load the model later for further use
loaded_ann8_smote = load_model('ann8_smote_model.h5')
```

ANN-8 Model with SMOTE Summary:

- **(Dense) layers:** 5 / **Neurons:** 256-128-128-64 / **Dropout:** 30-30-30-25% / **Learning Rate:** Initially 0.01 / **Batch Size:** 512 / **Epochs:** 600 / **Early Stop (val_accuracy):** 50
- **Accuracy:** 0.80 / **Val_Accuracy:** 0.69 / **Loss:** 0.5980 / **Val_Loss:** 0.6564 / **Train Recall (Class 2):** 0.95 / **Test Recall (Class 2):** 0.82

Improvements:

- The model shows a substantial improvement in handling the imbalanced class distribution with SMOTE, achieving a high recall for Class 2 in the training set.
- Dropout and batch normalization layers effectively control overfitting, evident from the convergence of training and validation loss curves.

No Improvement:

- Despite high training accuracy and recall, the validation accuracy remains lower, suggesting potential issues in generalizing to unseen data.
- The precision for Class 2 in the validation set is relatively low compared to other classes, indicating a trade-off between recall and precision.

Got Worse:

- There is a noticeable disparity between training and validation performance, particularly in precision for Class 1 and Class 2, which may indicate overfitting despite the use of dropout and other regularization techniques.

Final Model

Preprocessing

In [200...]

```
#FINAL MODEL

# Apply the column transformer to df_train and df_test datasets!
# df_train data (No target)
X_transformed = column_transformer.fit_transform(X) # Train data

# df_test data (No target)
df_test_transformed = column_transformer.transform(df_test) # Original Test Data

# Reassign the transformed features to X train data
X = pd.DataFrame(X_transformed, columns=features, index=X.index)

# Reassign the transformed features to Test Data which will be used for Submission Prediction
df_test = pd.DataFrame(df_test_transformed, columns=features, index=df_test.index)
```

In [201...]

```
#SCALE
scaler = MinMaxScaler().fit(X)

# Fit the scaler on X and transform both X and df_test data (no target)
X_scaled = scaler.transform(X)

# Apply the same transformation to df_test for Submission Prediction
df_test_scaled = scaler.transform(df_test)

# Convert scaled arrays back to DataFrame
X_scaled = pd.DataFrame(X_scaled, columns=X.columns)
df_test_scaled = pd.DataFrame(df_test_scaled, columns=df_test.columns) #for Submission Prediction
```

In [202...]

```
X.head()
```

Out[202...]

	Credit_Mix	Payment_of_Min_Amount	Payment_Behaviour	Occupation_Accountant	Occupation_Architect	Occupatio
0	3.000	0.000	3.000	0.000	0.000	
1	2.000	0.000	2.000	0.000	0.000	
2	2.000	0.000	1.000	0.000	0.000	
3	2.000	0.000	0.000	0.000	0.000	
4	2.000	0.000	4.000	0.000	0.000	

5 rows × 44 columns

In [203... df_test.head()

Out[203...]

	Credit_Mix	Payment_of_Min_Amount	Payment_Behaviour	Occupation_Accountant	Occupation_Architect	Occupatio
0	2.000	0.000	0.000	0.000	0.000	0.000
1	2.000	0.000	4.000	0.000	0.000	0.000
2	2.000	0.000	1.000	0.000	0.000	0.000
3	2.000	0.000	4.000	0.000	0.000	0.000
4	2.000	0.000	5.000	0.000	0.000	0.000

5 rows × 44 columns

In [207... from imblearn.over_sampling import SMOTE

```
smote = SMOTE()
X, y = smote.fit_resample(X,y)

# X_train_resampled and y_train_resampled are now ready for model training
print("X shape :", X.shape)
print("y_train shape :", y.shape)

y.value_counts()
```

```
X shape : (150786, 44)
y_train shape : (150786,)
```

Out[207... Credit_Score

```
2    50262
1    50262
0    50262
Name: count, dtype: int64
```

In [204... # from sklearn.utils import class_weight

```
# class_weights = class_weight.compute_class_weight('balanced',
#                                                    classes=np.unique(y),
#                                                    y=y)
# class_weights_dict = {i: class_weights[i] for i in range(len(class_weights))}

# class_weights_dict
```

Out[204... {0: 1.1439897357686675, 1: 0.6268022230180521, 2: 1.8851324397638423}

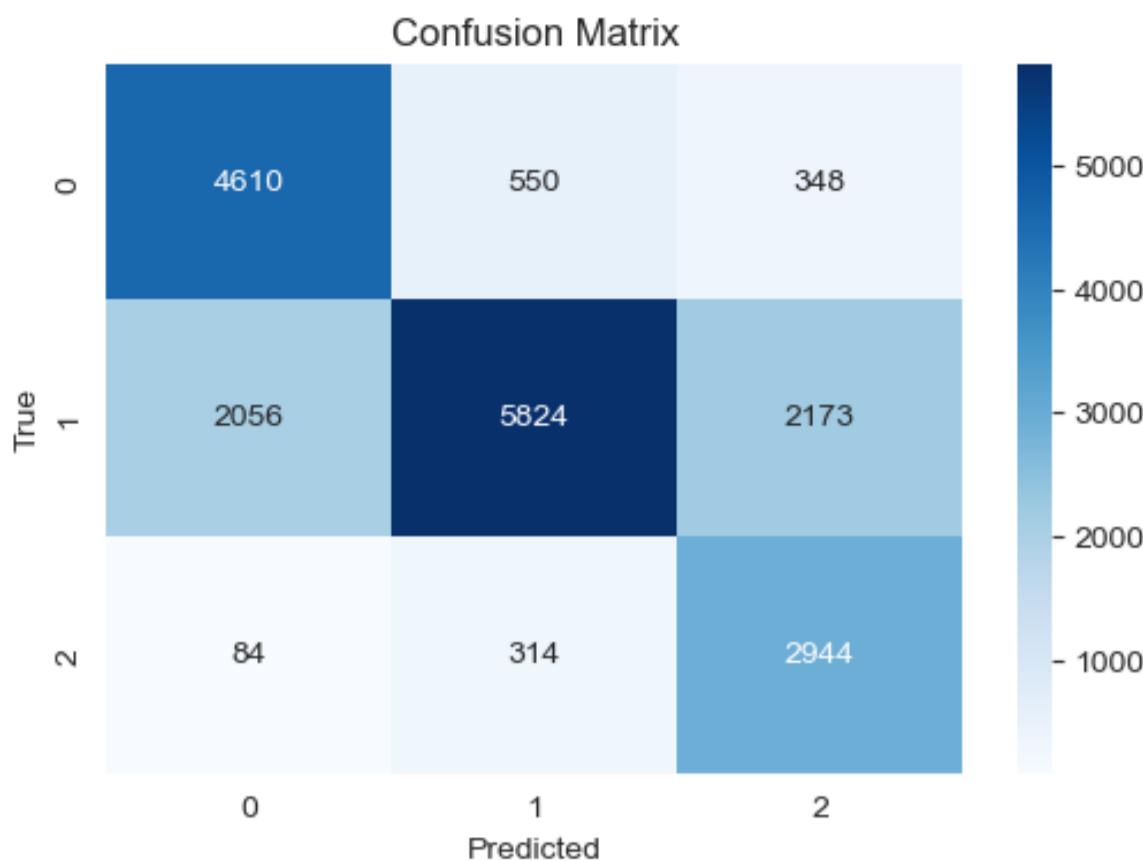
Model: ANN-7 with Smote:

```
y_pred = ann7_smote.predict(X_val)
y_pred_classes = y_pred.argmax(axis=1)

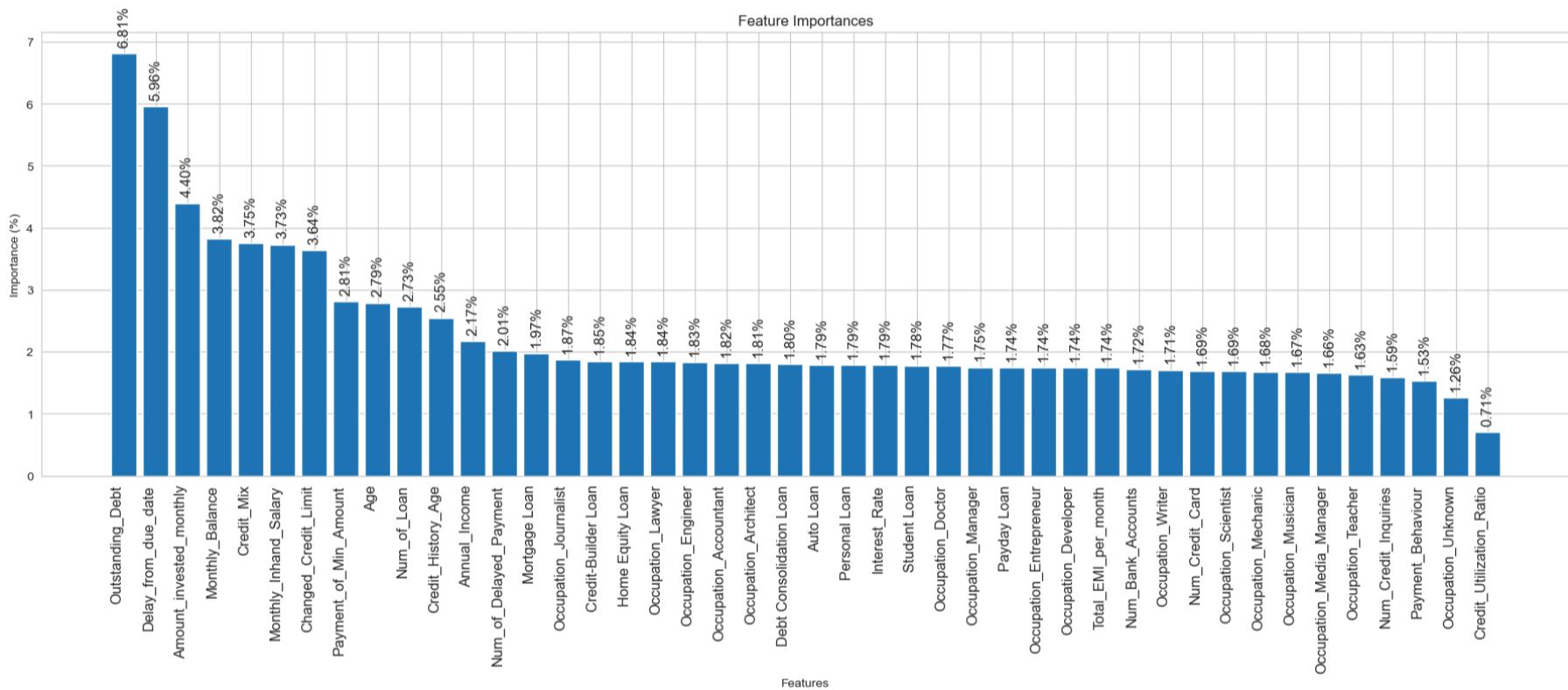
cm = confusion_matrix(y_val, y_pred_classes)

plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

591/591 ━━━━━━━━ 1s 938us/step



```
In [136]: plot_feature_importances(ann7_smote, X_train)
```



```
In [211]: # Final Model Architecture:
```

```
final_model = Sequential([
    BatchNormalization(),
    Dropout(0.3),
    Dense(256, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.25),
    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.2),
    Dense(3, activation='softmax')
])

# 2) Compiling the Model:
final_model.compile(optimizer=Adam(learning_rate=0.001),
                     loss='sparse_categorical_crossentropy',
                     metrics=['accuracy'])

# 3) Train the model
history = final_model.fit(
    x=X,
    y=y,
    batch_size=512,
```

```
epochs=500,  
verbose=1)
```

Epoch 1/500
295/295 5s 4ms/step - accuracy: 0.5394 - loss: 1.0734 - learning_rate: 0.0010
Epoch 2/500
295/295 1s 4ms/step - accuracy: 0.6393 - loss: 0.8335 - learning_rate: 0.0010
Epoch 3/500
295/295 1s 4ms/step - accuracy: 0.6647 - loss: 0.7922 - learning_rate: 0.0010
Epoch 4/500
295/295 1s 4ms/step - accuracy: 0.6731 - loss: 0.7754 - learning_rate: 0.0010
Epoch 5/500
295/295 1s 4ms/step - accuracy: 0.6773 - loss: 0.7694 - learning_rate: 0.0010
Epoch 6/500
295/295 1s 5ms/step - accuracy: 0.6784 - loss: 0.7638 - learning_rate: 0.0010
Epoch 7/500
295/295 1s 4ms/step - accuracy: 0.6812 - loss: 0.7575 - learning_rate: 0.0010
Epoch 8/500
295/295 1s 4ms/step - accuracy: 0.6842 - loss: 0.7508 - learning_rate: 0.0010
Epoch 9/500
295/295 1s 4ms/step - accuracy: 0.6844 - loss: 0.7445 - learning_rate: 0.0010
Epoch 10/500
295/295 1s 4ms/step - accuracy: 0.6851 - loss: 0.7459 - learning_rate: 0.0010
Epoch 11/500
295/295 1s 4ms/step - accuracy: 0.6894 - loss: 0.7346 - learning_rate: 0.0010
Epoch 12/500
295/295 1s 4ms/step - accuracy: 0.6880 - loss: 0.7349 - learning_rate: 0.0010
Epoch 13/500
295/295 1s 4ms/step - accuracy: 0.6896 - loss: 0.7302 - learning_rate: 0.0010
Epoch 14/500
295/295 1s 4ms/step - accuracy: 0.6915 - loss: 0.7278 - learning_rate: 0.0010
Epoch 15/500
295/295 1s 4ms/step - accuracy: 0.6941 - loss: 0.7213 - learning_rate: 0.0010
Epoch 16/500
295/295 1s 4ms/step - accuracy: 0.6948 - loss: 0.7178 - learning_rate: 0.0010
Epoch 17/500
295/295 1s 4ms/step - accuracy: 0.6961 - loss: 0.7159 - learning_rate: 0.0010
Epoch 18/500
295/295 1s 4ms/step - accuracy: 0.6992 - loss: 0.7110 - learning_rate: 0.0010
Epoch 19/500
295/295 1s 4ms/step - accuracy: 0.6985 - loss: 0.7102 - learning_rate: 0.0010
Epoch 20/500
295/295 1s 4ms/step - accuracy: 0.7002 - loss: 0.7071 - learning_rate: 0.0010
Epoch 21/500
295/295 1s 4ms/step - accuracy: 0.7006 - loss: 0.7061 - learning_rate: 0.0010
Epoch 22/500
295/295 1s 4ms/step - accuracy: 0.7014 - loss: 0.7039 - learning_rate: 0.0010
Epoch 23/500
295/295 1s 4ms/step - accuracy: 0.7007 - loss: 0.7061 - learning_rate: 0.0010
Epoch 24/500
295/295 1s 4ms/step - accuracy: 0.7043 - loss: 0.6975 - learning_rate: 0.0010
Epoch 25/500
295/295 1s 4ms/step - accuracy: 0.7051 - loss: 0.6972 - learning_rate: 0.0010
Epoch 26/500
295/295 1s 4ms/step - accuracy: 0.7049 - loss: 0.6953 - learning_rate: 0.0010
Epoch 27/500
295/295 1s 4ms/step - accuracy: 0.7053 - loss: 0.6951 - learning_rate: 0.0010
Epoch 28/500
295/295 1s 4ms/step - accuracy: 0.7043 - loss: 0.6965 - learning_rate: 0.0010
Epoch 29/500
295/295 1s 4ms/step - accuracy: 0.7060 - loss: 0.6960 - learning_rate: 0.0010
Epoch 30/500
295/295 1s 4ms/step - accuracy: 0.7058 - loss: 0.6922 - learning_rate: 0.0010
Epoch 31/500
295/295 1s 4ms/step - accuracy: 0.7052 - loss: 0.6918 - learning_rate: 0.0010
Epoch 32/500
295/295 1s 5ms/step - accuracy: 0.7070 - loss: 0.6901 - learning_rate: 0.0010
Epoch 33/500
295/295 1s 4ms/step - accuracy: 0.7095 - loss: 0.6871 - learning_rate: 0.0010
Epoch 34/500
295/295 1s 4ms/step - accuracy: 0.7099 - loss: 0.6867 - learning_rate: 0.0010
Epoch 35/500
295/295 1s 4ms/step - accuracy: 0.7105 - loss: 0.6879 - learning_rate: 0.0010
Epoch 36/500
295/295 1s 4ms/step - accuracy: 0.7072 - loss: 0.6874 - learning_rate: 0.0010
Epoch 37/500
295/295 1s 4ms/step - accuracy: 0.7064 - loss: 0.6887 - learning_rate: 0.0010
Epoch 38/500
295/295 1s 4ms/step - accuracy: 0.7090 - loss: 0.6849 - learning_rate: 0.0010

Epoch 39/500
295/295 1s 4ms/step - accuracy: 0.7081 - loss: 0.6856 - learning_rate: 0.0010
Epoch 40/500
295/295 1s 4ms/step - accuracy: 0.7073 - loss: 0.6861 - learning_rate: 0.0010
Epoch 41/500
295/295 1s 4ms/step - accuracy: 0.7060 - loss: 0.6853 - learning_rate: 0.0010
Epoch 42/500
295/295 1s 4ms/step - accuracy: 0.7104 - loss: 0.6808 - learning_rate: 0.0010
Epoch 43/500
295/295 1s 4ms/step - accuracy: 0.7106 - loss: 0.6798 - learning_rate: 0.0010
Epoch 44/500
295/295 1s 5ms/step - accuracy: 0.7076 - loss: 0.6816 - learning_rate: 0.0010
Epoch 45/500
295/295 1s 4ms/step - accuracy: 0.7084 - loss: 0.6842 - learning_rate: 0.0010
Epoch 46/500
295/295 1s 4ms/step - accuracy: 0.7134 - loss: 0.6757 - learning_rate: 0.0010
Epoch 47/500
295/295 1s 4ms/step - accuracy: 0.7145 - loss: 0.6722 - learning_rate: 0.0010
Epoch 48/500
295/295 1s 4ms/step - accuracy: 0.7108 - loss: 0.6778 - learning_rate: 0.0010
Epoch 49/500
295/295 1s 4ms/step - accuracy: 0.7088 - loss: 0.6807 - learning_rate: 0.0010
Epoch 50/500
295/295 1s 4ms/step - accuracy: 0.7122 - loss: 0.6782 - learning_rate: 0.0010
Epoch 51/500
295/295 1s 4ms/step - accuracy: 0.7122 - loss: 0.6747 - learning_rate: 0.0010
Epoch 52/500
295/295 1s 4ms/step - accuracy: 0.7118 - loss: 0.6749 - learning_rate: 0.0010
Epoch 53/500
295/295 1s 4ms/step - accuracy: 0.7125 - loss: 0.6767 - learning_rate: 0.0010
Epoch 54/500
295/295 1s 4ms/step - accuracy: 0.7139 - loss: 0.6706 - learning_rate: 0.0010
Epoch 55/500
295/295 1s 5ms/step - accuracy: 0.7133 - loss: 0.6729 - learning_rate: 0.0010
Epoch 56/500
295/295 1s 5ms/step - accuracy: 0.7142 - loss: 0.6738 - learning_rate: 0.0010
Epoch 57/500
295/295 1s 4ms/step - accuracy: 0.7127 - loss: 0.6740 - learning_rate: 0.0010
Epoch 58/500
295/295 1s 4ms/step - accuracy: 0.7139 - loss: 0.6729 - learning_rate: 0.0010
Epoch 59/500
295/295 1s 4ms/step - accuracy: 0.7147 - loss: 0.6695 - learning_rate: 0.0010
Epoch 60/500
295/295 1s 4ms/step - accuracy: 0.7138 - loss: 0.6712 - learning_rate: 0.0010
Epoch 61/500
295/295 1s 4ms/step - accuracy: 0.7127 - loss: 0.6739 - learning_rate: 0.0010
Epoch 62/500
295/295 1s 4ms/step - accuracy: 0.7156 - loss: 0.6694 - learning_rate: 0.0010
Epoch 63/500
295/295 1s 4ms/step - accuracy: 0.7151 - loss: 0.6706 - learning_rate: 0.0010
Epoch 64/500
295/295 1s 4ms/step - accuracy: 0.7162 - loss: 0.6677 - learning_rate: 0.0010
Epoch 65/500
295/295 1s 4ms/step - accuracy: 0.7128 - loss: 0.6711 - learning_rate: 0.0010
Epoch 66/500
295/295 1s 4ms/step - accuracy: 0.7176 - loss: 0.6647 - learning_rate: 0.0010
Epoch 67/500
295/295 1s 4ms/step - accuracy: 0.7136 - loss: 0.6708 - learning_rate: 0.0010
Epoch 68/500
295/295 1s 5ms/step - accuracy: 0.7155 - loss: 0.6665 - learning_rate: 0.0010
Epoch 69/500
295/295 1s 4ms/step - accuracy: 0.7157 - loss: 0.6668 - learning_rate: 0.0010
Epoch 70/500
295/295 1s 5ms/step - accuracy: 0.7147 - loss: 0.6683 - learning_rate: 0.0010
Epoch 71/500
295/295 1s 5ms/step - accuracy: 0.7162 - loss: 0.6644 - learning_rate: 0.0010
Epoch 72/500
295/295 1s 4ms/step - accuracy: 0.7167 - loss: 0.6666 - learning_rate: 0.0010
Epoch 73/500
295/295 2s 5ms/step - accuracy: 0.7164 - loss: 0.6641 - learning_rate: 0.0010
Epoch 74/500
295/295 1s 5ms/step - accuracy: 0.7141 - loss: 0.6652 - learning_rate: 0.0010
Epoch 75/500
295/295 2s 5ms/step - accuracy: 0.7160 - loss: 0.6635 - learning_rate: 0.0010
Epoch 76/500
295/295 1s 4ms/step - accuracy: 0.7151 - loss: 0.6658 - learning_rate: 0.0010

Epoch 77/500
295/295 1s 4ms/step - accuracy: 0.7154 - loss: 0.6650 - learning_rate: 0.0010
Epoch 78/500
295/295 1s 4ms/step - accuracy: 0.7158 - loss: 0.6663 - learning_rate: 0.0010
Epoch 79/500
295/295 1s 4ms/step - accuracy: 0.7155 - loss: 0.6639 - learning_rate: 0.0010
Epoch 80/500
295/295 1s 4ms/step - accuracy: 0.7148 - loss: 0.6650 - learning_rate: 0.0010
Epoch 81/500
295/295 1s 5ms/step - accuracy: 0.7198 - loss: 0.6578 - learning_rate: 0.0010
Epoch 82/500
295/295 1s 4ms/step - accuracy: 0.7171 - loss: 0.6614 - learning_rate: 0.0010
Epoch 83/500
295/295 2s 5ms/step - accuracy: 0.7145 - loss: 0.6644 - learning_rate: 0.0010
Epoch 84/500
295/295 2s 5ms/step - accuracy: 0.7214 - loss: 0.6571 - learning_rate: 0.0010
Epoch 85/500
295/295 1s 4ms/step - accuracy: 0.7135 - loss: 0.6646 - learning_rate: 0.0010
Epoch 86/500
295/295 1s 4ms/step - accuracy: 0.7175 - loss: 0.6610 - learning_rate: 0.0010
Epoch 87/500
295/295 1s 4ms/step - accuracy: 0.7182 - loss: 0.6584 - learning_rate: 0.0010
Epoch 88/500
295/295 1s 4ms/step - accuracy: 0.7182 - loss: 0.6617 - learning_rate: 0.0010
Epoch 89/500
295/295 1s 4ms/step - accuracy: 0.7193 - loss: 0.6587 - learning_rate: 0.0010
Epoch 90/500
295/295 1s 4ms/step - accuracy: 0.7167 - loss: 0.6588 - learning_rate: 0.0010
Epoch 91/500
295/295 1s 5ms/step - accuracy: 0.7186 - loss: 0.6582 - learning_rate: 0.0010
Epoch 92/500
295/295 1s 4ms/step - accuracy: 0.7177 - loss: 0.6618 - learning_rate: 0.0010
Epoch 93/500
295/295 1s 4ms/step - accuracy: 0.7175 - loss: 0.6601 - learning_rate: 0.0010
Epoch 94/500
295/295 1s 4ms/step - accuracy: 0.7188 - loss: 0.6582 - learning_rate: 0.0010
Epoch 95/500
295/295 2s 6ms/step - accuracy: 0.7175 - loss: 0.6626 - learning_rate: 0.0010
Epoch 96/500
295/295 1s 5ms/step - accuracy: 0.7182 - loss: 0.6626 - learning_rate: 0.0010
Epoch 97/500
295/295 1s 4ms/step - accuracy: 0.7159 - loss: 0.6603 - learning_rate: 0.0010
Epoch 98/500
295/295 1s 4ms/step - accuracy: 0.7201 - loss: 0.6584 - learning_rate: 0.0010
Epoch 99/500
295/295 1s 4ms/step - accuracy: 0.7185 - loss: 0.6598 - learning_rate: 0.0010
Epoch 100/500
295/295 1s 4ms/step - accuracy: 0.7179 - loss: 0.6574 - learning_rate: 0.0010
Epoch 101/500
295/295 1s 4ms/step - accuracy: 0.7194 - loss: 0.6588 - learning_rate: 0.0010
Epoch 102/500
295/295 1s 4ms/step - accuracy: 0.7195 - loss: 0.6620 - learning_rate: 0.0010
Epoch 103/500
295/295 1s 4ms/step - accuracy: 0.7206 - loss: 0.6559 - learning_rate: 0.0010
Epoch 104/500
295/295 1s 4ms/step - accuracy: 0.7181 - loss: 0.6586 - learning_rate: 0.0010
Epoch 105/500
295/295 1s 4ms/step - accuracy: 0.7196 - loss: 0.6573 - learning_rate: 0.0010
Epoch 106/500
295/295 1s 4ms/step - accuracy: 0.7193 - loss: 0.6579 - learning_rate: 0.0010
Epoch 107/500
295/295 1s 4ms/step - accuracy: 0.7202 - loss: 0.6535 - learning_rate: 0.0010
Epoch 108/500
295/295 1s 4ms/step - accuracy: 0.7190 - loss: 0.6579 - learning_rate: 0.0010
Epoch 109/500
295/295 1s 4ms/step - accuracy: 0.7194 - loss: 0.6574 - learning_rate: 0.0010
Epoch 110/500
295/295 1s 4ms/step - accuracy: 0.7188 - loss: 0.6571 - learning_rate: 0.0010
Epoch 111/500
295/295 1s 4ms/step - accuracy: 0.7211 - loss: 0.6545 - learning_rate: 0.0010
Epoch 112/500
295/295 1s 4ms/step - accuracy: 0.7225 - loss: 0.6527 - learning_rate: 0.0010
Epoch 113/500
295/295 2s 5ms/step - accuracy: 0.7198 - loss: 0.6569 - learning_rate: 0.0010
Epoch 114/500
295/295 1s 5ms/step - accuracy: 0.7202 - loss: 0.6557 - learning_rate: 0.0010

Epoch 115/500
295/295 1s 5ms/step - accuracy: 0.7187 - loss: 0.6555 - learning_rate: 0.0010
Epoch 116/500
295/295 1s 5ms/step - accuracy: 0.7206 - loss: 0.6571 - learning_rate: 0.0010
Epoch 117/500
295/295 1s 4ms/step - accuracy: 0.7213 - loss: 0.6563 - learning_rate: 0.0010
Epoch 118/500
295/295 1s 4ms/step - accuracy: 0.7191 - loss: 0.6567 - learning_rate: 0.0010
Epoch 119/500
295/295 1s 4ms/step - accuracy: 0.7215 - loss: 0.6530 - learning_rate: 0.0010
Epoch 120/500
295/295 1s 4ms/step - accuracy: 0.7202 - loss: 0.6530 - learning_rate: 0.0010
Epoch 121/500
295/295 1s 4ms/step - accuracy: 0.7211 - loss: 0.6534 - learning_rate: 0.0010
Epoch 122/500
295/295 1s 4ms/step - accuracy: 0.7208 - loss: 0.6533 - learning_rate: 0.0010
Epoch 123/500
295/295 2s 5ms/step - accuracy: 0.7247 - loss: 0.6496 - learning_rate: 0.0010
Epoch 124/500
295/295 2s 4ms/step - accuracy: 0.7217 - loss: 0.6523 - learning_rate: 0.0010
Epoch 125/500
295/295 1s 4ms/step - accuracy: 0.7219 - loss: 0.6516 - learning_rate: 0.0010
Epoch 126/500
295/295 1s 4ms/step - accuracy: 0.7235 - loss: 0.6496 - learning_rate: 0.0010
Epoch 127/500
295/295 1s 4ms/step - accuracy: 0.7206 - loss: 0.6556 - learning_rate: 0.0010
Epoch 128/500
295/295 1s 4ms/step - accuracy: 0.7222 - loss: 0.6496 - learning_rate: 0.0010
Epoch 129/500
295/295 1s 4ms/step - accuracy: 0.7215 - loss: 0.6527 - learning_rate: 0.0010
Epoch 130/500
295/295 1s 4ms/step - accuracy: 0.7193 - loss: 0.6557 - learning_rate: 0.0010
Epoch 131/500
295/295 1s 4ms/step - accuracy: 0.7197 - loss: 0.6540 - learning_rate: 0.0010
Epoch 132/500
295/295 1s 4ms/step - accuracy: 0.7206 - loss: 0.6535 - learning_rate: 0.0010
Epoch 133/500
295/295 2s 5ms/step - accuracy: 0.7215 - loss: 0.6514 - learning_rate: 0.0010
Epoch 134/500
295/295 1s 5ms/step - accuracy: 0.7221 - loss: 0.6535 - learning_rate: 0.0010
Epoch 135/500
295/295 1s 4ms/step - accuracy: 0.7225 - loss: 0.6508 - learning_rate: 0.0010
Epoch 136/500
295/295 1s 4ms/step - accuracy: 0.7203 - loss: 0.6550 - learning_rate: 0.0010
Epoch 137/500
295/295 1s 5ms/step - accuracy: 0.7229 - loss: 0.6502 - learning_rate: 0.0010
Epoch 138/500
295/295 1s 4ms/step - accuracy: 0.7203 - loss: 0.6516 - learning_rate: 0.0010
Epoch 139/500
295/295 1s 5ms/step - accuracy: 0.7221 - loss: 0.6507 - learning_rate: 0.0010
Epoch 140/500
295/295 1s 5ms/step - accuracy: 0.7208 - loss: 0.6514 - learning_rate: 0.0010
Epoch 141/500
295/295 1s 4ms/step - accuracy: 0.7211 - loss: 0.6528 - learning_rate: 0.0010
Epoch 142/500
295/295 1s 4ms/step - accuracy: 0.7198 - loss: 0.6544 - learning_rate: 0.0010
Epoch 143/500
295/295 1s 4ms/step - accuracy: 0.7216 - loss: 0.6501 - learning_rate: 0.0010
Epoch 144/500
295/295 1s 4ms/step - accuracy: 0.7223 - loss: 0.6512 - learning_rate: 0.0010
Epoch 145/500
295/295 1s 5ms/step - accuracy: 0.7232 - loss: 0.6471 - learning_rate: 0.0010
Epoch 146/500
295/295 1s 5ms/step - accuracy: 0.7225 - loss: 0.6481 - learning_rate: 0.0010
Epoch 147/500
295/295 1s 5ms/step - accuracy: 0.7215 - loss: 0.6504 - learning_rate: 0.0010
Epoch 148/500
295/295 1s 4ms/step - accuracy: 0.7229 - loss: 0.6490 - learning_rate: 0.0010
Epoch 149/500
295/295 1s 5ms/step - accuracy: 0.7254 - loss: 0.6452 - learning_rate: 0.0010
Epoch 150/500
295/295 1s 4ms/step - accuracy: 0.7235 - loss: 0.6486 - learning_rate: 0.0010
Epoch 151/500
295/295 1s 4ms/step - accuracy: 0.7223 - loss: 0.6511 - learning_rate: 0.0010
Epoch 152/500
295/295 1s 4ms/step - accuracy: 0.7217 - loss: 0.6515 - learning_rate: 0.0010

Epoch 153/500
295/295 1s 4ms/step - accuracy: 0.7227 - loss: 0.6512 - learning_rate: 0.0010
Epoch 154/500
295/295 1s 4ms/step - accuracy: 0.7203 - loss: 0.6487 - learning_rate: 0.0010
Epoch 155/500
295/295 1s 4ms/step - accuracy: 0.7236 - loss: 0.6473 - learning_rate: 0.0010
Epoch 156/500
295/295 1s 4ms/step - accuracy: 0.7212 - loss: 0.6493 - learning_rate: 0.0010
Epoch 157/500
295/295 1s 4ms/step - accuracy: 0.7219 - loss: 0.6498 - learning_rate: 0.0010
Epoch 158/500
295/295 1s 4ms/step - accuracy: 0.7225 - loss: 0.6489 - learning_rate: 0.0010
Epoch 159/500
295/295 1s 5ms/step - accuracy: 0.7233 - loss: 0.6482 - learning_rate: 0.0010
Epoch 160/500
295/295 1s 4ms/step - accuracy: 0.7239 - loss: 0.6456 - learning_rate: 0.0010
Epoch 161/500
295/295 1s 4ms/step - accuracy: 0.7224 - loss: 0.6506 - learning_rate: 0.0010
Epoch 162/500
295/295 1s 4ms/step - accuracy: 0.7225 - loss: 0.6492 - learning_rate: 0.0010
Epoch 163/500
295/295 1s 4ms/step - accuracy: 0.7245 - loss: 0.6462 - learning_rate: 0.0010
Epoch 164/500
295/295 1s 4ms/step - accuracy: 0.7210 - loss: 0.6510 - learning_rate: 0.0010
Epoch 165/500
295/295 1s 4ms/step - accuracy: 0.7224 - loss: 0.6498 - learning_rate: 0.0010
Epoch 166/500
295/295 1s 4ms/step - accuracy: 0.7209 - loss: 0.6466 - learning_rate: 0.0010
Epoch 167/500
295/295 1s 4ms/step - accuracy: 0.7239 - loss: 0.6452 - learning_rate: 0.0010
Epoch 168/500
295/295 1s 4ms/step - accuracy: 0.7225 - loss: 0.6487 - learning_rate: 0.0010
Epoch 169/500
295/295 1s 4ms/step - accuracy: 0.7235 - loss: 0.6459 - learning_rate: 0.0010
Epoch 170/500
295/295 1s 4ms/step - accuracy: 0.7220 - loss: 0.6485 - learning_rate: 0.0010
Epoch 171/500
295/295 1s 5ms/step - accuracy: 0.7224 - loss: 0.6482 - learning_rate: 0.0010
Epoch 172/500
295/295 1s 4ms/step - accuracy: 0.7221 - loss: 0.6504 - learning_rate: 0.0010
Epoch 173/500
295/295 1s 4ms/step - accuracy: 0.7216 - loss: 0.6483 - learning_rate: 0.0010
Epoch 174/500
295/295 1s 4ms/step - accuracy: 0.7222 - loss: 0.6500 - learning_rate: 0.0010
Epoch 175/500
295/295 1s 4ms/step - accuracy: 0.7214 - loss: 0.6478 - learning_rate: 0.0010
Epoch 176/500
295/295 1s 4ms/step - accuracy: 0.7229 - loss: 0.6466 - learning_rate: 0.0010
Epoch 177/500
295/295 1s 4ms/step - accuracy: 0.7254 - loss: 0.6452 - learning_rate: 0.0010
Epoch 178/500
295/295 1s 4ms/step - accuracy: 0.7241 - loss: 0.6444 - learning_rate: 0.0010
Epoch 179/500
295/295 1s 4ms/step - accuracy: 0.7218 - loss: 0.6498 - learning_rate: 0.0010
Epoch 180/500
295/295 1s 4ms/step - accuracy: 0.7218 - loss: 0.6503 - learning_rate: 0.0010
Epoch 181/500
295/295 1s 4ms/step - accuracy: 0.7219 - loss: 0.6481 - learning_rate: 0.0010
Epoch 182/500
295/295 1s 4ms/step - accuracy: 0.7224 - loss: 0.6464 - learning_rate: 0.0010
Epoch 183/500
295/295 1s 4ms/step - accuracy: 0.7237 - loss: 0.6449 - learning_rate: 0.0010
Epoch 184/500
295/295 1s 4ms/step - accuracy: 0.7226 - loss: 0.6470 - learning_rate: 0.0010
Epoch 185/500
295/295 1s 4ms/step - accuracy: 0.7221 - loss: 0.6485 - learning_rate: 0.0010
Epoch 186/500
295/295 1s 4ms/step - accuracy: 0.7232 - loss: 0.6477 - learning_rate: 0.0010
Epoch 187/500
295/295 1s 4ms/step - accuracy: 0.7223 - loss: 0.6482 - learning_rate: 0.0010
Epoch 188/500
295/295 1s 4ms/step - accuracy: 0.7247 - loss: 0.6465 - learning_rate: 0.0010
Epoch 189/500
295/295 1s 4ms/step - accuracy: 0.7227 - loss: 0.6467 - learning_rate: 0.0010
Epoch 190/500
295/295 1s 5ms/step - accuracy: 0.7227 - loss: 0.6480 - learning_rate: 0.0010

Epoch 191/500
295/295 1s 4ms/step - accuracy: 0.7239 - loss: 0.6475 - learning_rate: 0.0010
Epoch 192/500
295/295 1s 4ms/step - accuracy: 0.7245 - loss: 0.6417 - learning_rate: 0.0010
Epoch 193/500
295/295 1s 4ms/step - accuracy: 0.7255 - loss: 0.6456 - learning_rate: 0.0010
Epoch 194/500
295/295 1s 4ms/step - accuracy: 0.7252 - loss: 0.6432 - learning_rate: 0.0010
Epoch 195/500
295/295 1s 4ms/step - accuracy: 0.7210 - loss: 0.6491 - learning_rate: 0.0010
Epoch 196/500
295/295 1s 4ms/step - accuracy: 0.7235 - loss: 0.6463 - learning_rate: 0.0010
Epoch 197/500
295/295 1s 4ms/step - accuracy: 0.7245 - loss: 0.6446 - learning_rate: 0.0010
Epoch 198/500
295/295 1s 4ms/step - accuracy: 0.7255 - loss: 0.6452 - learning_rate: 0.0010
Epoch 199/500
295/295 1s 4ms/step - accuracy: 0.7260 - loss: 0.6444 - learning_rate: 0.0010
Epoch 200/500
295/295 1s 4ms/step - accuracy: 0.7240 - loss: 0.6472 - learning_rate: 0.0010
Epoch 201/500
295/295 1s 4ms/step - accuracy: 0.7229 - loss: 0.6476 - learning_rate: 0.0010
Epoch 202/500
295/295 1s 4ms/step - accuracy: 0.7245 - loss: 0.6457 - learning_rate: 0.0010
Epoch 203/500
295/295 1s 5ms/step - accuracy: 0.7245 - loss: 0.6461 - learning_rate: 0.0010
Epoch 204/500
295/295 2s 5ms/step - accuracy: 0.7244 - loss: 0.6418 - learning_rate: 0.0010
Epoch 205/500
295/295 1s 4ms/step - accuracy: 0.7237 - loss: 0.6454 - learning_rate: 0.0010
Epoch 206/500
295/295 1s 5ms/step - accuracy: 0.7247 - loss: 0.6428 - learning_rate: 0.0010
Epoch 207/500
295/295 1s 5ms/step - accuracy: 0.7249 - loss: 0.6432 - learning_rate: 0.0010
Epoch 208/500
295/295 1s 4ms/step - accuracy: 0.7239 - loss: 0.6469 - learning_rate: 0.0010
Epoch 209/500
295/295 1s 4ms/step - accuracy: 0.7252 - loss: 0.6436 - learning_rate: 0.0010
Epoch 210/500
295/295 1s 4ms/step - accuracy: 0.7251 - loss: 0.6458 - learning_rate: 0.0010
Epoch 211/500
295/295 1s 4ms/step - accuracy: 0.7253 - loss: 0.6405 - learning_rate: 0.0010
Epoch 212/500
295/295 1s 4ms/step - accuracy: 0.7239 - loss: 0.6446 - learning_rate: 0.0010
Epoch 213/500
295/295 1s 4ms/step - accuracy: 0.7244 - loss: 0.6472 - learning_rate: 0.0010
Epoch 214/500
295/295 1s 5ms/step - accuracy: 0.7227 - loss: 0.6450 - learning_rate: 0.0010
Epoch 215/500
295/295 1s 4ms/step - accuracy: 0.7247 - loss: 0.6444 - learning_rate: 0.0010
Epoch 216/500
295/295 1s 4ms/step - accuracy: 0.7233 - loss: 0.6469 - learning_rate: 0.0010
Epoch 217/500
295/295 1s 4ms/step - accuracy: 0.7239 - loss: 0.6450 - learning_rate: 0.0010
Epoch 218/500
295/295 1s 5ms/step - accuracy: 0.7224 - loss: 0.6476 - learning_rate: 0.0010
Epoch 219/500
295/295 1s 4ms/step - accuracy: 0.7269 - loss: 0.6409 - learning_rate: 0.0010
Epoch 220/500
295/295 1s 4ms/step - accuracy: 0.7227 - loss: 0.6433 - learning_rate: 0.0010
Epoch 221/500
295/295 1s 5ms/step - accuracy: 0.7243 - loss: 0.6453 - learning_rate: 0.0010
Epoch 222/500
295/295 1s 4ms/step - accuracy: 0.7259 - loss: 0.6412 - learning_rate: 0.0010
Epoch 223/500
295/295 1s 4ms/step - accuracy: 0.7248 - loss: 0.6441 - learning_rate: 0.0010
Epoch 224/500
295/295 1s 4ms/step - accuracy: 0.7218 - loss: 0.6473 - learning_rate: 0.0010
Epoch 225/500
295/295 1s 4ms/step - accuracy: 0.7215 - loss: 0.6466 - learning_rate: 0.0010
Epoch 226/500
295/295 1s 5ms/step - accuracy: 0.7253 - loss: 0.6437 - learning_rate: 0.0010
Epoch 227/500
295/295 1s 4ms/step - accuracy: 0.7248 - loss: 0.6426 - learning_rate: 0.0010
Epoch 228/500
295/295 1s 4ms/step - accuracy: 0.7255 - loss: 0.6435 - learning_rate: 0.0010

Epoch 229/500
295/295 1s 4ms/step - accuracy: 0.7249 - loss: 0.6446 - learning_rate: 0.0010
Epoch 230/500
295/295 1s 4ms/step - accuracy: 0.7268 - loss: 0.6402 - learning_rate: 0.0010
Epoch 231/500
295/295 1s 4ms/step - accuracy: 0.7239 - loss: 0.6439 - learning_rate: 0.0010
Epoch 232/500
295/295 1s 4ms/step - accuracy: 0.7239 - loss: 0.6455 - learning_rate: 0.0010
Epoch 233/500
295/295 1s 4ms/step - accuracy: 0.7259 - loss: 0.6419 - learning_rate: 0.0010
Epoch 234/500
295/295 1s 4ms/step - accuracy: 0.7256 - loss: 0.6443 - learning_rate: 0.0010
Epoch 235/500
295/295 1s 4ms/step - accuracy: 0.7244 - loss: 0.6411 - learning_rate: 0.0010
Epoch 236/500
295/295 1s 4ms/step - accuracy: 0.7238 - loss: 0.6431 - learning_rate: 0.0010
Epoch 237/500
295/295 1s 4ms/step - accuracy: 0.7260 - loss: 0.6443 - learning_rate: 0.0010
Epoch 238/500
295/295 1s 4ms/step - accuracy: 0.7231 - loss: 0.6490 - learning_rate: 0.0010
Epoch 239/500
295/295 1s 4ms/step - accuracy: 0.7240 - loss: 0.6457 - learning_rate: 0.0010
Epoch 240/500
295/295 1s 4ms/step - accuracy: 0.7244 - loss: 0.6424 - learning_rate: 0.0010
Epoch 241/500
295/295 1s 4ms/step - accuracy: 0.7245 - loss: 0.6425 - learning_rate: 0.0010
Epoch 242/500
295/295 1s 4ms/step - accuracy: 0.7257 - loss: 0.6413 - learning_rate: 0.0010
Epoch 243/500
295/295 1s 4ms/step - accuracy: 0.7251 - loss: 0.6415 - learning_rate: 0.0010
Epoch 244/500
295/295 1s 4ms/step - accuracy: 0.7242 - loss: 0.6406 - learning_rate: 0.0010
Epoch 245/500
295/295 1s 4ms/step - accuracy: 0.7270 - loss: 0.6439 - learning_rate: 0.0010
Epoch 246/500
295/295 2s 6ms/step - accuracy: 0.7288 - loss: 0.6401 - learning_rate: 0.0010
Epoch 247/500
295/295 2s 5ms/step - accuracy: 0.7232 - loss: 0.6443 - learning_rate: 0.0010
Epoch 248/500
295/295 1s 5ms/step - accuracy: 0.7253 - loss: 0.6431 - learning_rate: 0.0010
Epoch 249/500
295/295 1s 5ms/step - accuracy: 0.7246 - loss: 0.6435 - learning_rate: 0.0010
Epoch 250/500
295/295 2s 6ms/step - accuracy: 0.7253 - loss: 0.6412 - learning_rate: 0.0010
Epoch 251/500
295/295 1s 5ms/step - accuracy: 0.7243 - loss: 0.6450 - learning_rate: 0.0010
Epoch 252/500
295/295 1s 5ms/step - accuracy: 0.7246 - loss: 0.6438 - learning_rate: 0.0010
Epoch 253/500
295/295 1s 4ms/step - accuracy: 0.7233 - loss: 0.6458 - learning_rate: 0.0010
Epoch 254/500
295/295 1s 4ms/step - accuracy: 0.7267 - loss: 0.6406 - learning_rate: 0.0010
Epoch 255/500
295/295 1s 4ms/step - accuracy: 0.7268 - loss: 0.6396 - learning_rate: 0.0010
Epoch 256/500
295/295 1s 4ms/step - accuracy: 0.7243 - loss: 0.6425 - learning_rate: 0.0010
Epoch 257/500
295/295 1s 4ms/step - accuracy: 0.7247 - loss: 0.6420 - learning_rate: 0.0010
Epoch 258/500
295/295 1s 4ms/step - accuracy: 0.7250 - loss: 0.6424 - learning_rate: 0.0010
Epoch 259/500
295/295 1s 4ms/step - accuracy: 0.7240 - loss: 0.6445 - learning_rate: 0.0010
Epoch 260/500
295/295 1s 4ms/step - accuracy: 0.7252 - loss: 0.6402 - learning_rate: 0.0010
Epoch 261/500
295/295 1s 4ms/step - accuracy: 0.7241 - loss: 0.6422 - learning_rate: 0.0010
Epoch 262/500
295/295 1s 4ms/step - accuracy: 0.7265 - loss: 0.6401 - learning_rate: 0.0010
Epoch 263/500
295/295 1s 4ms/step - accuracy: 0.7271 - loss: 0.6403 - learning_rate: 0.0010
Epoch 264/500
295/295 1s 4ms/step - accuracy: 0.7264 - loss: 0.6386 - learning_rate: 0.0010
Epoch 265/500
295/295 1s 4ms/step - accuracy: 0.7259 - loss: 0.6414 - learning_rate: 0.0010
Epoch 266/500
295/295 1s 4ms/step - accuracy: 0.7252 - loss: 0.6419 - learning_rate: 0.0010

Epoch 267/500
295/295 1s 4ms/step - accuracy: 0.7242 - loss: 0.6405 - learning_rate: 0.0010
Epoch 268/500
295/295 1s 4ms/step - accuracy: 0.7258 - loss: 0.6398 - learning_rate: 0.0010
Epoch 269/500
295/295 1s 4ms/step - accuracy: 0.7282 - loss: 0.6368 - learning_rate: 0.0010
Epoch 270/500
295/295 1s 4ms/step - accuracy: 0.7268 - loss: 0.6396 - learning_rate: 0.0010
Epoch 271/500
295/295 1s 4ms/step - accuracy: 0.7245 - loss: 0.6421 - learning_rate: 0.0010
Epoch 272/500
295/295 2s 5ms/step - accuracy: 0.7261 - loss: 0.6429 - learning_rate: 0.0010
Epoch 273/500
295/295 1s 4ms/step - accuracy: 0.7239 - loss: 0.6455 - learning_rate: 0.0010
Epoch 274/500
295/295 1s 4ms/step - accuracy: 0.7282 - loss: 0.6368 - learning_rate: 0.0010
Epoch 275/500
295/295 1s 4ms/step - accuracy: 0.7250 - loss: 0.6404 - learning_rate: 0.0010
Epoch 276/500
295/295 1s 4ms/step - accuracy: 0.7233 - loss: 0.6412 - learning_rate: 0.0010
Epoch 277/500
295/295 1s 4ms/step - accuracy: 0.7243 - loss: 0.6415 - learning_rate: 0.0010
Epoch 278/500
295/295 1s 4ms/step - accuracy: 0.7284 - loss: 0.6359 - learning_rate: 0.0010
Epoch 279/500
295/295 1s 4ms/step - accuracy: 0.7263 - loss: 0.6415 - learning_rate: 0.0010
Epoch 280/500
295/295 1s 4ms/step - accuracy: 0.7267 - loss: 0.6376 - learning_rate: 0.0010
Epoch 281/500
295/295 1s 4ms/step - accuracy: 0.7254 - loss: 0.6426 - learning_rate: 0.0010
Epoch 282/500
295/295 1s 4ms/step - accuracy: 0.7257 - loss: 0.6421 - learning_rate: 0.0010
Epoch 283/500
295/295 1s 4ms/step - accuracy: 0.7251 - loss: 0.6422 - learning_rate: 0.0010
Epoch 284/500
295/295 1s 4ms/step - accuracy: 0.7275 - loss: 0.6402 - learning_rate: 0.0010
Epoch 285/500
295/295 1s 4ms/step - accuracy: 0.7266 - loss: 0.6401 - learning_rate: 0.0010
Epoch 286/500
295/295 1s 4ms/step - accuracy: 0.7258 - loss: 0.6409 - learning_rate: 0.0010
Epoch 287/500
295/295 1s 4ms/step - accuracy: 0.7257 - loss: 0.6404 - learning_rate: 0.0010
Epoch 288/500
295/295 1s 4ms/step - accuracy: 0.7239 - loss: 0.6417 - learning_rate: 0.0010
Epoch 289/500
295/295 1s 4ms/step - accuracy: 0.7249 - loss: 0.6432 - learning_rate: 0.0010
Epoch 290/500
295/295 1s 4ms/step - accuracy: 0.7264 - loss: 0.6377 - learning_rate: 0.0010
Epoch 291/500
295/295 1s 4ms/step - accuracy: 0.7260 - loss: 0.6417 - learning_rate: 0.0010
Epoch 292/500
295/295 1s 4ms/step - accuracy: 0.7247 - loss: 0.6436 - learning_rate: 0.0010
Epoch 293/500
295/295 1s 4ms/step - accuracy: 0.7264 - loss: 0.6382 - learning_rate: 0.0010
Epoch 294/500
295/295 1s 4ms/step - accuracy: 0.7254 - loss: 0.6408 - learning_rate: 0.0010
Epoch 295/500
295/295 1s 4ms/step - accuracy: 0.7244 - loss: 0.6426 - learning_rate: 0.0010
Epoch 296/500
295/295 1s 4ms/step - accuracy: 0.7247 - loss: 0.6417 - learning_rate: 0.0010
Epoch 297/500
295/295 1s 4ms/step - accuracy: 0.7253 - loss: 0.6405 - learning_rate: 0.0010
Epoch 298/500
295/295 1s 4ms/step - accuracy: 0.7277 - loss: 0.6378 - learning_rate: 0.0010
Epoch 299/500
295/295 1s 4ms/step - accuracy: 0.7252 - loss: 0.6389 - learning_rate: 0.0010
Epoch 300/500
295/295 1s 4ms/step - accuracy: 0.7259 - loss: 0.6410 - learning_rate: 0.0010
Epoch 301/500
295/295 1s 4ms/step - accuracy: 0.7231 - loss: 0.6434 - learning_rate: 0.0010
Epoch 302/500
295/295 2s 5ms/step - accuracy: 0.7241 - loss: 0.6452 - learning_rate: 0.0010
Epoch 303/500
295/295 1s 4ms/step - accuracy: 0.7269 - loss: 0.6381 - learning_rate: 0.0010
Epoch 304/500
295/295 1s 4ms/step - accuracy: 0.7281 - loss: 0.6372 - learning_rate: 0.0010

Epoch 305/500
295/295 1s 4ms/step - accuracy: 0.7246 - loss: 0.6413 - learning_rate: 0.0010
Epoch 306/500
295/295 1s 4ms/step - accuracy: 0.7245 - loss: 0.6390 - learning_rate: 0.0010
Epoch 307/500
295/295 1s 4ms/step - accuracy: 0.7257 - loss: 0.6412 - learning_rate: 0.0010
Epoch 308/500
295/295 1s 4ms/step - accuracy: 0.7239 - loss: 0.6425 - learning_rate: 0.0010
Epoch 309/500
295/295 1s 4ms/step - accuracy: 0.7244 - loss: 0.6415 - learning_rate: 0.0010
Epoch 310/500
295/295 1s 4ms/step - accuracy: 0.7254 - loss: 0.6396 - learning_rate: 0.0010
Epoch 311/500
295/295 1s 4ms/step - accuracy: 0.7284 - loss: 0.6386 - learning_rate: 0.0010
Epoch 312/500
295/295 1s 4ms/step - accuracy: 0.7263 - loss: 0.6399 - learning_rate: 0.0010
Epoch 313/500
295/295 1s 4ms/step - accuracy: 0.7265 - loss: 0.6405 - learning_rate: 0.0010
Epoch 314/500
295/295 1s 4ms/step - accuracy: 0.7253 - loss: 0.6399 - learning_rate: 0.0010
Epoch 315/500
295/295 1s 4ms/step - accuracy: 0.7271 - loss: 0.6405 - learning_rate: 0.0010
Epoch 316/500
295/295 1s 4ms/step - accuracy: 0.7279 - loss: 0.6357 - learning_rate: 0.0010
Epoch 317/500
295/295 1s 4ms/step - accuracy: 0.7258 - loss: 0.6396 - learning_rate: 0.0010
Epoch 318/500
295/295 1s 4ms/step - accuracy: 0.7241 - loss: 0.6427 - learning_rate: 0.0010
Epoch 319/500
295/295 1s 4ms/step - accuracy: 0.7222 - loss: 0.6435 - learning_rate: 0.0010
Epoch 320/500
295/295 1s 4ms/step - accuracy: 0.7257 - loss: 0.6407 - learning_rate: 0.0010
Epoch 321/500
295/295 1s 4ms/step - accuracy: 0.7267 - loss: 0.6374 - learning_rate: 0.0010
Epoch 322/500
295/295 1s 4ms/step - accuracy: 0.7256 - loss: 0.6423 - learning_rate: 0.0010
Epoch 323/500
295/295 1s 4ms/step - accuracy: 0.7253 - loss: 0.6402 - learning_rate: 0.0010
Epoch 324/500
295/295 1s 4ms/step - accuracy: 0.7276 - loss: 0.6362 - learning_rate: 0.0010
Epoch 325/500
295/295 1s 4ms/step - accuracy: 0.7285 - loss: 0.6391 - learning_rate: 0.0010
Epoch 326/500
295/295 1s 4ms/step - accuracy: 0.7284 - loss: 0.6356 - learning_rate: 0.0010
Epoch 327/500
295/295 1s 4ms/step - accuracy: 0.7265 - loss: 0.6373 - learning_rate: 0.0010
Epoch 328/500
295/295 1s 4ms/step - accuracy: 0.7243 - loss: 0.6450 - learning_rate: 0.0010
Epoch 329/500
295/295 1s 4ms/step - accuracy: 0.7257 - loss: 0.6400 - learning_rate: 0.0010
Epoch 330/500
295/295 1s 4ms/step - accuracy: 0.7294 - loss: 0.6374 - learning_rate: 0.0010
Epoch 331/500
295/295 1s 4ms/step - accuracy: 0.7254 - loss: 0.6385 - learning_rate: 0.0010
Epoch 332/500
295/295 1s 4ms/step - accuracy: 0.7261 - loss: 0.6402 - learning_rate: 0.0010
Epoch 333/500
295/295 1s 4ms/step - accuracy: 0.7256 - loss: 0.6409 - learning_rate: 0.0010
Epoch 334/500
295/295 1s 4ms/step - accuracy: 0.7244 - loss: 0.6422 - learning_rate: 0.0010
Epoch 335/500
295/295 1s 4ms/step - accuracy: 0.7265 - loss: 0.6396 - learning_rate: 0.0010
Epoch 336/500
295/295 1s 4ms/step - accuracy: 0.7255 - loss: 0.6393 - learning_rate: 0.0010
Epoch 337/500
295/295 1s 4ms/step - accuracy: 0.7272 - loss: 0.6380 - learning_rate: 0.0010
Epoch 338/500
295/295 1s 4ms/step - accuracy: 0.7283 - loss: 0.6370 - learning_rate: 0.0010
Epoch 339/500
295/295 1s 4ms/step - accuracy: 0.7285 - loss: 0.6373 - learning_rate: 0.0010
Epoch 340/500
295/295 1s 4ms/step - accuracy: 0.7259 - loss: 0.6413 - learning_rate: 0.0010
Epoch 341/500
295/295 1s 4ms/step - accuracy: 0.7255 - loss: 0.6403 - learning_rate: 0.0010
Epoch 342/500
295/295 1s 4ms/step - accuracy: 0.7273 - loss: 0.6373 - learning_rate: 0.0010

Epoch 343/500
295/295 1s 4ms/step - accuracy: 0.7247 - loss: 0.6385 - learning_rate: 0.0010
Epoch 344/500
295/295 1s 4ms/step - accuracy: 0.7264 - loss: 0.6373 - learning_rate: 0.0010
Epoch 345/500
295/295 1s 4ms/step - accuracy: 0.7258 - loss: 0.6379 - learning_rate: 0.0010
Epoch 346/500
295/295 1s 5ms/step - accuracy: 0.7270 - loss: 0.6379 - learning_rate: 0.0010
Epoch 347/500
295/295 1s 4ms/step - accuracy: 0.7264 - loss: 0.6383 - learning_rate: 0.0010
Epoch 348/500
295/295 1s 4ms/step - accuracy: 0.7265 - loss: 0.6373 - learning_rate: 0.0010
Epoch 349/500
295/295 1s 4ms/step - accuracy: 0.7276 - loss: 0.6371 - learning_rate: 0.0010
Epoch 350/500
295/295 1s 4ms/step - accuracy: 0.7272 - loss: 0.6365 - learning_rate: 0.0010
Epoch 351/500
295/295 1s 4ms/step - accuracy: 0.7250 - loss: 0.6415 - learning_rate: 0.0010
Epoch 352/500
295/295 1s 4ms/step - accuracy: 0.7278 - loss: 0.6358 - learning_rate: 0.0010
Epoch 353/500
295/295 1s 4ms/step - accuracy: 0.7238 - loss: 0.6450 - learning_rate: 0.0010
Epoch 354/500
295/295 1s 4ms/step - accuracy: 0.7267 - loss: 0.6389 - learning_rate: 0.0010
Epoch 355/500
295/295 1s 4ms/step - accuracy: 0.7269 - loss: 0.6383 - learning_rate: 0.0010
Epoch 356/500
295/295 1s 4ms/step - accuracy: 0.7257 - loss: 0.6392 - learning_rate: 0.0010
Epoch 357/500
295/295 1s 4ms/step - accuracy: 0.7271 - loss: 0.6377 - learning_rate: 0.0010
Epoch 358/500
295/295 1s 4ms/step - accuracy: 0.7247 - loss: 0.6376 - learning_rate: 0.0010
Epoch 359/500
295/295 1s 4ms/step - accuracy: 0.7280 - loss: 0.6369 - learning_rate: 0.0010
Epoch 360/500
295/295 1s 4ms/step - accuracy: 0.7265 - loss: 0.6386 - learning_rate: 0.0010
Epoch 361/500
295/295 1s 4ms/step - accuracy: 0.7262 - loss: 0.6377 - learning_rate: 0.0010
Epoch 362/500
295/295 1s 4ms/step - accuracy: 0.7266 - loss: 0.6388 - learning_rate: 0.0010
Epoch 363/500
295/295 1s 4ms/step - accuracy: 0.7281 - loss: 0.6395 - learning_rate: 0.0010
Epoch 364/500
295/295 1s 4ms/step - accuracy: 0.7256 - loss: 0.6403 - learning_rate: 0.0010
Epoch 365/500
295/295 1s 4ms/step - accuracy: 0.7259 - loss: 0.6360 - learning_rate: 0.0010
Epoch 366/500
295/295 3s 4ms/step - accuracy: 0.7262 - loss: 0.6381 - learning_rate: 0.0010
Epoch 367/500
295/295 1s 4ms/step - accuracy: 0.7271 - loss: 0.6389 - learning_rate: 0.0010
Epoch 368/500
295/295 1s 4ms/step - accuracy: 0.7271 - loss: 0.6396 - learning_rate: 0.0010
Epoch 369/500
295/295 1s 4ms/step - accuracy: 0.7280 - loss: 0.6369 - learning_rate: 0.0010
Epoch 370/500
295/295 1s 4ms/step - accuracy: 0.7273 - loss: 0.6378 - learning_rate: 0.0010
Epoch 371/500
295/295 1s 4ms/step - accuracy: 0.7249 - loss: 0.6393 - learning_rate: 0.0010
Epoch 372/500
295/295 1s 4ms/step - accuracy: 0.7269 - loss: 0.6372 - learning_rate: 0.0010
Epoch 373/500
295/295 1s 4ms/step - accuracy: 0.7248 - loss: 0.6406 - learning_rate: 0.0010
Epoch 374/500
295/295 1s 4ms/step - accuracy: 0.7269 - loss: 0.6391 - learning_rate: 0.0010
Epoch 375/500
295/295 1s 4ms/step - accuracy: 0.7253 - loss: 0.6410 - learning_rate: 0.0010
Epoch 376/500
295/295 1s 4ms/step - accuracy: 0.7254 - loss: 0.6400 - learning_rate: 0.0010
Epoch 377/500
295/295 1s 5ms/step - accuracy: 0.7262 - loss: 0.6413 - learning_rate: 0.0010
Epoch 378/500
295/295 1s 4ms/step - accuracy: 0.7262 - loss: 0.6383 - learning_rate: 0.0010
Epoch 379/500
295/295 1s 4ms/step - accuracy: 0.7273 - loss: 0.6384 - learning_rate: 0.0010
Epoch 380/500
295/295 1s 4ms/step - accuracy: 0.7252 - loss: 0.6388 - learning_rate: 0.0010

Epoch 381/500
295/295 1s 4ms/step - accuracy: 0.7262 - loss: 0.6392 - learning_rate: 0.0010
Epoch 382/500
295/295 1s 4ms/step - accuracy: 0.7279 - loss: 0.6380 - learning_rate: 0.0010
Epoch 383/500
295/295 1s 4ms/step - accuracy: 0.7261 - loss: 0.6372 - learning_rate: 0.0010
Epoch 384/500
295/295 1s 4ms/step - accuracy: 0.7257 - loss: 0.6383 - learning_rate: 0.0010
Epoch 385/500
295/295 1s 4ms/step - accuracy: 0.7285 - loss: 0.6369 - learning_rate: 0.0010
Epoch 386/500
295/295 1s 4ms/step - accuracy: 0.7255 - loss: 0.6404 - learning_rate: 0.0010
Epoch 387/500
295/295 1s 4ms/step - accuracy: 0.7274 - loss: 0.6352 - learning_rate: 0.0010
Epoch 388/500
295/295 1s 4ms/step - accuracy: 0.7274 - loss: 0.6373 - learning_rate: 0.0010
Epoch 389/500
295/295 1s 4ms/step - accuracy: 0.7270 - loss: 0.6366 - learning_rate: 0.0010
Epoch 390/500
295/295 1s 4ms/step - accuracy: 0.7261 - loss: 0.6388 - learning_rate: 0.0010
Epoch 391/500
295/295 1s 4ms/step - accuracy: 0.7279 - loss: 0.6340 - learning_rate: 0.0010
Epoch 392/500
295/295 1s 4ms/step - accuracy: 0.7251 - loss: 0.6410 - learning_rate: 0.0010
Epoch 393/500
295/295 1s 4ms/step - accuracy: 0.7265 - loss: 0.6397 - learning_rate: 0.0010
Epoch 394/500
295/295 1s 4ms/step - accuracy: 0.7237 - loss: 0.6438 - learning_rate: 0.0010
Epoch 395/500
295/295 1s 5ms/step - accuracy: 0.7263 - loss: 0.6379 - learning_rate: 0.0010
Epoch 396/500
295/295 1s 4ms/step - accuracy: 0.7251 - loss: 0.6376 - learning_rate: 0.0010
Epoch 397/500
295/295 1s 4ms/step - accuracy: 0.7275 - loss: 0.6361 - learning_rate: 0.0010
Epoch 398/500
295/295 1s 4ms/step - accuracy: 0.7269 - loss: 0.6394 - learning_rate: 0.0010
Epoch 399/500
295/295 1s 4ms/step - accuracy: 0.7249 - loss: 0.6390 - learning_rate: 0.0010
Epoch 400/500
295/295 1s 4ms/step - accuracy: 0.7280 - loss: 0.6385 - learning_rate: 0.0010
Epoch 401/500
295/295 1s 4ms/step - accuracy: 0.7271 - loss: 0.6370 - learning_rate: 0.0010
Epoch 402/500
295/295 1s 4ms/step - accuracy: 0.7280 - loss: 0.6365 - learning_rate: 0.0010
Epoch 403/500
295/295 1s 4ms/step - accuracy: 0.7282 - loss: 0.6359 - learning_rate: 0.0010
Epoch 404/500
295/295 1s 4ms/step - accuracy: 0.7264 - loss: 0.6384 - learning_rate: 0.0010
Epoch 405/500
295/295 1s 4ms/step - accuracy: 0.7294 - loss: 0.6336 - learning_rate: 0.0010
Epoch 406/500
295/295 1s 4ms/step - accuracy: 0.7271 - loss: 0.6360 - learning_rate: 0.0010
Epoch 407/500
295/295 1s 4ms/step - accuracy: 0.7270 - loss: 0.6357 - learning_rate: 0.0010
Epoch 408/500
295/295 1s 4ms/step - accuracy: 0.7265 - loss: 0.6373 - learning_rate: 0.0010
Epoch 409/500
295/295 1s 4ms/step - accuracy: 0.7264 - loss: 0.6378 - learning_rate: 0.0010
Epoch 410/500
295/295 1s 4ms/step - accuracy: 0.7282 - loss: 0.6369 - learning_rate: 0.0010
Epoch 411/500
295/295 1s 4ms/step - accuracy: 0.7259 - loss: 0.6383 - learning_rate: 0.0010
Epoch 412/500
295/295 1s 5ms/step - accuracy: 0.7264 - loss: 0.6390 - learning_rate: 0.0010
Epoch 413/500
295/295 1s 4ms/step - accuracy: 0.7270 - loss: 0.6375 - learning_rate: 0.0010
Epoch 414/500
295/295 1s 4ms/step - accuracy: 0.7262 - loss: 0.6401 - learning_rate: 0.0010
Epoch 415/500
295/295 1s 5ms/step - accuracy: 0.7287 - loss: 0.6364 - learning_rate: 0.0010
Epoch 416/500
295/295 1s 5ms/step - accuracy: 0.7287 - loss: 0.6364 - learning_rate: 0.0010
Epoch 417/500
295/295 1s 5ms/step - accuracy: 0.7268 - loss: 0.6395 - learning_rate: 0.0010
Epoch 418/500
295/295 1s 4ms/step - accuracy: 0.7286 - loss: 0.6328 - learning_rate: 0.0010

Epoch 419/500
295/295 1s 4ms/step - accuracy: 0.7273 - loss: 0.6367 - learning_rate: 0.0010
Epoch 420/500
295/295 1s 4ms/step - accuracy: 0.7280 - loss: 0.6380 - learning_rate: 0.0010
Epoch 421/500
295/295 1s 4ms/step - accuracy: 0.7279 - loss: 0.6346 - learning_rate: 0.0010
Epoch 422/500
295/295 1s 4ms/step - accuracy: 0.7261 - loss: 0.6396 - learning_rate: 0.0010
Epoch 423/500
295/295 1s 4ms/step - accuracy: 0.7263 - loss: 0.6356 - learning_rate: 0.0010
Epoch 424/500
295/295 1s 4ms/step - accuracy: 0.7257 - loss: 0.6378 - learning_rate: 0.0010
Epoch 425/500
295/295 1s 4ms/step - accuracy: 0.7277 - loss: 0.6361 - learning_rate: 0.0010
Epoch 426/500
295/295 1s 4ms/step - accuracy: 0.7280 - loss: 0.6370 - learning_rate: 0.0010
Epoch 427/500
295/295 1s 4ms/step - accuracy: 0.7267 - loss: 0.6390 - learning_rate: 0.0010
Epoch 428/500
295/295 1s 4ms/step - accuracy: 0.7282 - loss: 0.6346 - learning_rate: 0.0010
Epoch 429/500
295/295 1s 4ms/step - accuracy: 0.7284 - loss: 0.6348 - learning_rate: 0.0010
Epoch 430/500
295/295 1s 5ms/step - accuracy: 0.7258 - loss: 0.6383 - learning_rate: 0.0010
Epoch 431/500
295/295 1s 4ms/step - accuracy: 0.7276 - loss: 0.6355 - learning_rate: 0.0010
Epoch 432/500
295/295 1s 4ms/step - accuracy: 0.7273 - loss: 0.6364 - learning_rate: 0.0010
Epoch 433/500
295/295 1s 4ms/step - accuracy: 0.7281 - loss: 0.6336 - learning_rate: 0.0010
Epoch 434/500
295/295 1s 4ms/step - accuracy: 0.7301 - loss: 0.6325 - learning_rate: 0.0010
Epoch 435/500
295/295 1s 4ms/step - accuracy: 0.7255 - loss: 0.6377 - learning_rate: 0.0010
Epoch 436/500
295/295 1s 4ms/step - accuracy: 0.7275 - loss: 0.6388 - learning_rate: 0.0010
Epoch 437/500
295/295 1s 4ms/step - accuracy: 0.7283 - loss: 0.6351 - learning_rate: 0.0010
Epoch 438/500
295/295 1s 4ms/step - accuracy: 0.7269 - loss: 0.6389 - learning_rate: 0.0010
Epoch 439/500
295/295 1s 4ms/step - accuracy: 0.7271 - loss: 0.6353 - learning_rate: 0.0010
Epoch 440/500
295/295 1s 4ms/step - accuracy: 0.7291 - loss: 0.6358 - learning_rate: 0.0010
Epoch 441/500
295/295 1s 4ms/step - accuracy: 0.7269 - loss: 0.6373 - learning_rate: 0.0010
Epoch 442/500
295/295 1s 4ms/step - accuracy: 0.7267 - loss: 0.6383 - learning_rate: 0.0010
Epoch 443/500
295/295 1s 4ms/step - accuracy: 0.7277 - loss: 0.6343 - learning_rate: 0.0010
Epoch 444/500
295/295 1s 4ms/step - accuracy: 0.7281 - loss: 0.6350 - learning_rate: 0.0010
Epoch 445/500
295/295 1s 4ms/step - accuracy: 0.7266 - loss: 0.6364 - learning_rate: 0.0010
Epoch 446/500
295/295 1s 4ms/step - accuracy: 0.7252 - loss: 0.6398 - learning_rate: 0.0010
Epoch 447/500
295/295 1s 4ms/step - accuracy: 0.7274 - loss: 0.6364 - learning_rate: 0.0010
Epoch 448/500
295/295 1s 4ms/step - accuracy: 0.7251 - loss: 0.6396 - learning_rate: 0.0010
Epoch 449/500
295/295 1s 4ms/step - accuracy: 0.7284 - loss: 0.6351 - learning_rate: 0.0010
Epoch 450/500
295/295 1s 4ms/step - accuracy: 0.7270 - loss: 0.6360 - learning_rate: 0.0010
Epoch 451/500
295/295 1s 4ms/step - accuracy: 0.7289 - loss: 0.6326 - learning_rate: 0.0010
Epoch 452/500
295/295 1s 4ms/step - accuracy: 0.7273 - loss: 0.6368 - learning_rate: 0.0010
Epoch 453/500
295/295 1s 4ms/step - accuracy: 0.7266 - loss: 0.6366 - learning_rate: 0.0010
Epoch 454/500
295/295 1s 4ms/step - accuracy: 0.7270 - loss: 0.6366 - learning_rate: 0.0010
Epoch 455/500
295/295 1s 4ms/step - accuracy: 0.7289 - loss: 0.6343 - learning_rate: 0.0010
Epoch 456/500
295/295 1s 4ms/step - accuracy: 0.7300 - loss: 0.6358 - learning_rate: 0.0010

Epoch 457/500
295/295 1s 4ms/step - accuracy: 0.7271 - loss: 0.6368 - learning_rate: 0.0010
Epoch 458/500
295/295 1s 4ms/step - accuracy: 0.7251 - loss: 0.6386 - learning_rate: 0.0010
Epoch 459/500
295/295 1s 4ms/step - accuracy: 0.7258 - loss: 0.6379 - learning_rate: 0.0010
Epoch 460/500
295/295 1s 4ms/step - accuracy: 0.7271 - loss: 0.6368 - learning_rate: 0.0010
Epoch 461/500
295/295 1s 4ms/step - accuracy: 0.7275 - loss: 0.6366 - learning_rate: 0.0010
Epoch 462/500
295/295 1s 4ms/step - accuracy: 0.7278 - loss: 0.6360 - learning_rate: 0.0010
Epoch 463/500
295/295 1s 4ms/step - accuracy: 0.7298 - loss: 0.6328 - learning_rate: 0.0010
Epoch 464/500
295/295 1s 4ms/step - accuracy: 0.7285 - loss: 0.6336 - learning_rate: 0.0010
Epoch 465/500
295/295 1s 4ms/step - accuracy: 0.7276 - loss: 0.6345 - learning_rate: 0.0010
Epoch 466/500
295/295 1s 4ms/step - accuracy: 0.7313 - loss: 0.6314 - learning_rate: 0.0010
Epoch 467/500
295/295 1s 4ms/step - accuracy: 0.7278 - loss: 0.6346 - learning_rate: 0.0010
Epoch 468/500
295/295 1s 4ms/step - accuracy: 0.7295 - loss: 0.6337 - learning_rate: 0.0010
Epoch 469/500
295/295 1s 4ms/step - accuracy: 0.7273 - loss: 0.6358 - learning_rate: 0.0010
Epoch 470/500
295/295 1s 4ms/step - accuracy: 0.7259 - loss: 0.6382 - learning_rate: 0.0010
Epoch 471/500
295/295 1s 4ms/step - accuracy: 0.7263 - loss: 0.6355 - learning_rate: 0.0010
Epoch 472/500
295/295 1s 4ms/step - accuracy: 0.7279 - loss: 0.6377 - learning_rate: 0.0010
Epoch 473/500
295/295 1s 4ms/step - accuracy: 0.7280 - loss: 0.6366 - learning_rate: 0.0010
Epoch 474/500
295/295 1s 4ms/step - accuracy: 0.7271 - loss: 0.6343 - learning_rate: 0.0010
Epoch 475/500
295/295 1s 4ms/step - accuracy: 0.7308 - loss: 0.6343 - learning_rate: 0.0010
Epoch 476/500
295/295 1s 4ms/step - accuracy: 0.7293 - loss: 0.6312 - learning_rate: 0.0010
Epoch 477/500
295/295 1s 4ms/step - accuracy: 0.7267 - loss: 0.6392 - learning_rate: 0.0010
Epoch 478/500
295/295 1s 4ms/step - accuracy: 0.7276 - loss: 0.6352 - learning_rate: 0.0010
Epoch 479/500
295/295 1s 4ms/step - accuracy: 0.7287 - loss: 0.6369 - learning_rate: 0.0010
Epoch 480/500
295/295 1s 4ms/step - accuracy: 0.7276 - loss: 0.6327 - learning_rate: 0.0010
Epoch 481/500
295/295 1s 4ms/step - accuracy: 0.7273 - loss: 0.6383 - learning_rate: 0.0010
Epoch 482/500
295/295 1s 4ms/step - accuracy: 0.7287 - loss: 0.6347 - learning_rate: 0.0010
Epoch 483/500
295/295 1s 4ms/step - accuracy: 0.7275 - loss: 0.6349 - learning_rate: 0.0010
Epoch 484/500
295/295 1s 4ms/step - accuracy: 0.7280 - loss: 0.6341 - learning_rate: 0.0010
Epoch 485/500
295/295 1s 4ms/step - accuracy: 0.7259 - loss: 0.6360 - learning_rate: 0.0010
Epoch 486/500
295/295 1s 4ms/step - accuracy: 0.7273 - loss: 0.6352 - learning_rate: 0.0010
Epoch 487/500
295/295 1s 4ms/step - accuracy: 0.7270 - loss: 0.6343 - learning_rate: 0.0010
Epoch 488/500
295/295 1s 4ms/step - accuracy: 0.7277 - loss: 0.6327 - learning_rate: 0.0010
Epoch 489/500
295/295 1s 4ms/step - accuracy: 0.7272 - loss: 0.6329 - learning_rate: 0.0010
Epoch 490/500
295/295 1s 4ms/step - accuracy: 0.7284 - loss: 0.6343 - learning_rate: 0.0010
Epoch 491/500
295/295 1s 4ms/step - accuracy: 0.7292 - loss: 0.6334 - learning_rate: 0.0010
Epoch 492/500
295/295 1s 5ms/step - accuracy: 0.7272 - loss: 0.6336 - learning_rate: 0.0010
Epoch 493/500
295/295 1s 4ms/step - accuracy: 0.7302 - loss: 0.6325 - learning_rate: 0.0010
Epoch 494/500
295/295 1s 4ms/step - accuracy: 0.7281 - loss: 0.6338 - learning_rate: 0.0010

```
Epoch 495/500
295/295 1s 4ms/step - accuracy: 0.7280 - loss: 0.6364 - learning_rate: 0.0010
Epoch 496/500
295/295 1s 4ms/step - accuracy: 0.7288 - loss: 0.6362 - learning_rate: 0.0010
Epoch 497/500
295/295 1s 4ms/step - accuracy: 0.7259 - loss: 0.6380 - learning_rate: 0.0010
Epoch 498/500
295/295 1s 4ms/step - accuracy: 0.7287 - loss: 0.6367 - learning_rate: 0.0010
Epoch 499/500
295/295 1s 4ms/step - accuracy: 0.7261 - loss: 0.6383 - learning_rate: 0.0010
Epoch 500/500
295/295 1s 4ms/step - accuracy: 0.7262 - loss: 0.6365 - learning_rate: 0.0010
```

In [213]: `final_model.evaluate(X, y)`

4713/4713 4s 740us/step - accuracy: 0.6068 - loss: 1.3670

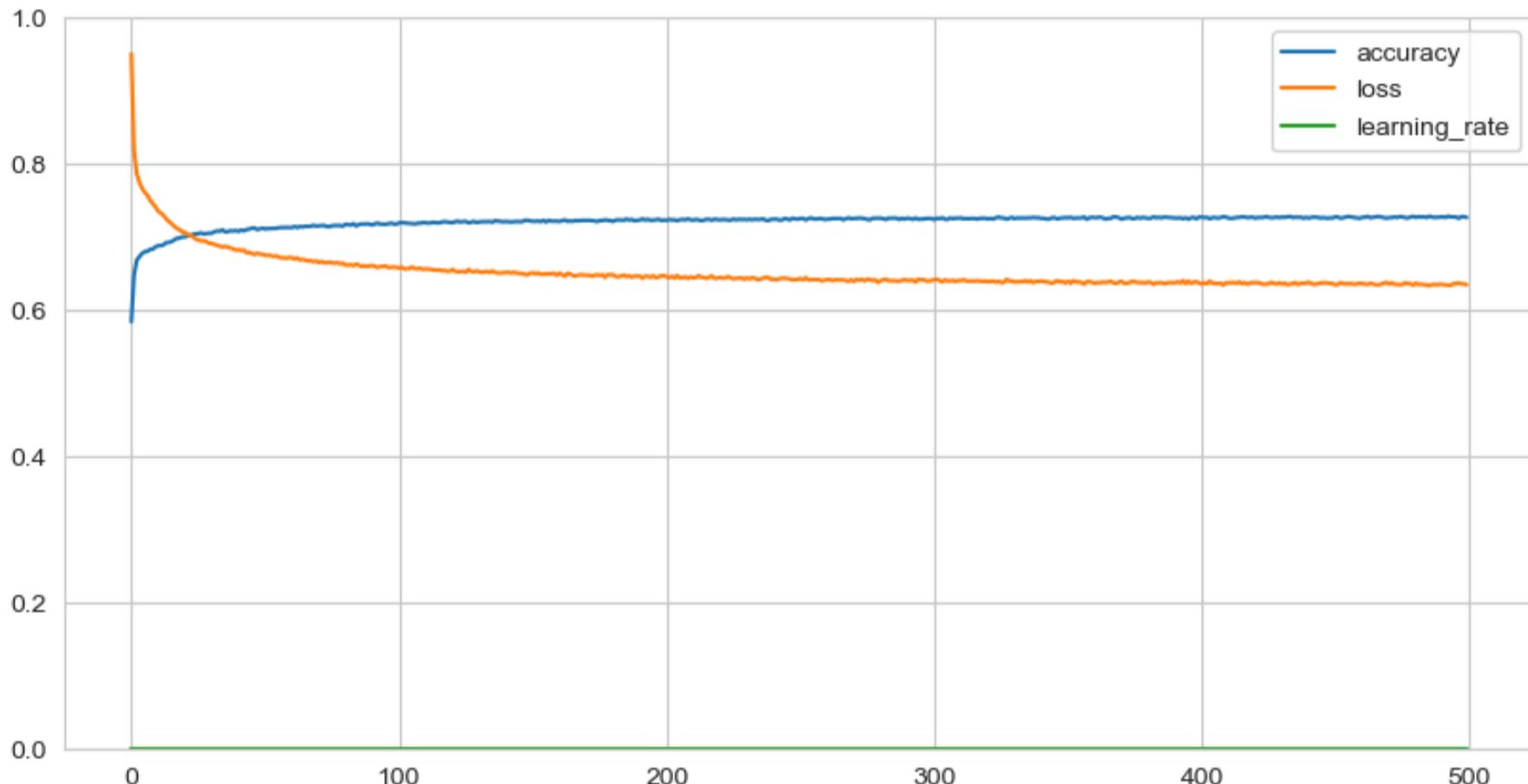
Out[213]: [1.0154026746749878, 0.7046675682067871]

In [217]: `pd.DataFrame(history.history).plot(figsize=(10,5))`

`plt.grid(True)`

`plt.gca().set_ylim(0, 1)`

`plt.show()`



In [118]: `# Save the Model`

`from tensorflow.keras.models import load_model`

`# Save the model`

`final_model.save('final_model_ANN_creditscore.h5')`

`# Load the model`

`ann_final_model = load_model('final_model_ANN_creditscore.h5')`

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`.
` . This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.

Prediction with the Test Data

In [119]: `df_test.head()`

Out[119...]

	Credit_Mix	Payment_of_Min_Amount	Payment_Behaviour	Occupation_Accountant	Occupation_Architect	Occupatio
0	2.000	0.000	0.000	0.000	0.000	
1	2.000	0.000	4.000	0.000	0.000	
2	2.000	0.000	1.000	0.000	0.000	
3	2.000	0.000	4.000	0.000	0.000	
4	2.000	0.000	5.000	0.000	0.000	

5 rows × 44 columns

In [121...]

```
# Predictions
probabilities = ann_final_model.predict(df_test)
probabilities
```

1457/1457 ━━━━━━━━ 2s 1ms/step

Out[121...]

```
array([[0.14058124, 0.33113644, 0.52828234],
       [0.09729843, 0.36774498, 0.5349566 ],
       [0.12302919, 0.35304   , 0.5239308 ],
       ...,
       [0.10455655, 0.71971667, 0.17572676],
       [0.13746727, 0.6255915 , 0.23694122],
       [0.17775083, 0.5120907 , 0.3101585 ]], dtype=float32)
```

In [122...]

```
proba_df = pd.DataFrame(probabilities, columns=['Probability_Class_0', 'Probability_Class_1', 'Probability_'])
proba_df
```

Out[122...]

	Probability_Class_0	Probability_Class_1	Probability_Class_2
0	0.141	0.331	0.528
1	0.097	0.368	0.535
2	0.123	0.353	0.524
3	0.105	0.310	0.586
4	0.099	0.188	0.713
...
46604	0.625	0.351	0.025
46605	0.810	0.179	0.012
46606	0.105	0.720	0.176
46607	0.137	0.626	0.237
46608	0.178	0.512	0.310

46609 rows × 3 columns

In [123...]

```
pred_labels = np.argmax(probabilities, axis=1)
pred_labels
```

Out[123...]

```
array([2, 2, 2, ..., 1, 1, 1], dtype=int64)
```

In [125...]

```
pred_df = pd.DataFrame(pred_labels, columns=['Predicted_Label'])
pred_df
```

Out[125...]

Predicted_Label

	Predicted_Label
0	2
1	2
2	2
3	2
4	2
...	...
46604	0
46605	0
46606	1
46607	1
46608	1

46609 rows × 1 columns

In [126...]

```
# Concatenate predictions and probabilities DataFrames along the columns axis
pred_proba_df = pd.concat([proba_df, pred_df], axis=1)
pred_proba_df
```

Out[126...]

Probability_Class_0 **Probability_Class_1** **Probability_Class_2** **Predicted_Label**

	Probability_Class_0	Probability_Class_1	Probability_Class_2	Predicted_Label
0	0.141	0.331	0.528	2
1	0.097	0.368	0.535	2
2	0.123	0.353	0.524	2
3	0.105	0.310	0.586	2
4	0.099	0.188	0.713	2
...
46604	0.625	0.351	0.025	0
46605	0.810	0.179	0.012	0
46606	0.105	0.720	0.176	1
46607	0.137	0.626	0.237	1
46608	0.178	0.512	0.310	1

46609 rows × 4 columns

In [127...]

```
pred_proba_df.describe().T
```

Out[127...]

	count	mean	std	min	25%	50%	75%	max
Probability_Class_0	46609.000	0.343	0.305	0.000	0.114	0.148	0.645	0.990
Probability_Class_1	46609.000	0.421	0.221	0.010	0.234	0.375	0.625	0.899
Probability_Class_2	46609.000	0.235	0.219	0.000	0.029	0.165	0.422	0.786
Predicted_Label	46609.000	0.907	0.761	0.000	0.000	1.000	1.000	2.000

In [128...]

```
pred_proba_df['Predicted_Label'].value_counts()
```

Out[128...]

```
Predicted_Label
1    19230
0    15860
2    11519
Name: count, dtype: int64
```

Project Summary:

This project undertook a comprehensive exploratory data analysis (EDA), addressing missing and unusual values while retaining outliers to preserve the data's inherent characteristics.

Preprocessing included meticulous encoding of categorical columns and employing pipelines to prevent data leakage, ensuring data integrity.

Data scaling was executed appropriately, and to address the imbalanced classes, both SMOTE and class weighting techniques were applied. SMOTE demonstrated superior efficacy, leading to the iterative optimization of the ANN model architecture in eight distinct configurations. The ANN-7 model with SMOTE was selected as the final model due to its robust performance, and predictions were subsequently made on the test dataset.

For additional details about each model tested in this project, please visit the repository on my [GitHub profile](#).

Importance of the ANN Model in Credit Score Prediction:

The ANN-7 model's ability to effectively manage imbalanced data makes it especially significant for credit score prediction. This model excels in accurately distinguishing between varying degrees of creditworthiness, which is critical in the financial industry. By achieving a high recall for Class 2—the class representing potential defaulters—the model minimizes the risk associated with extending credit to high-risk individuals. This precision in predicting default risks not only reduces potential financial losses but also reinforces the reliability and stability of credit lending practices. The model's robust generalization capabilities ensure that credit assessments are based on accurate and dependable evaluations, supporting responsible lending decisions and maintaining financial stability. This strategic application underscores the project's contribution to enhancing predictive accuracy and operational efficiency in financial decision-making processes.

If you find this work helpful, don't forget to give it an  UPVOTE! and join the discussion! 

Thank you...

Duygu Jones | Data Scientist | 2024

Follow me: duygujones.com | [Linkedin](#) | [GitHub](#) | [Kaggle](#) | [Medium](#) | [Tableau](#)