Import Libraries¶

In [1]:

```
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 [2]:

```
# All relevant libraries
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.saving import save_model
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import roc_auc_score, roc_curve, precision_recall_curve, average_pre
cision_score
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.model_selection import GridSearchCV

from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

from keras.layers import BatchNormalization
from keras.optimizers import Adam
from keras.regularizers import l2
```

Loading Files

In [3]:

```
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
train.head()
```

Out[3]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Acc
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	

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Acc
3	0x1605	CUS_0xd40	April	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	NaN	
4	0x1606	CUS_0xd40	May	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843	

5 rows × 28 columns

In [4]:

```
train.head()
```

Out[4]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Acc
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	0x1605	CUS_0xd40	April	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	NaN	
4	0x1606	CUS_0xd40	May	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843	

5 rows × 28 columns

In [5]:

```
test.head()
```

Out[5]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_
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	0x160d	CUS_0xd40	December	Aaron Maashoh	24_	821-00-0265	Scientist	19114.12	NaN	
4	0x1616	CUS_0x21b1	September	Rick Rothackerj	28	004-07-5839	_______	34847.84	3037.987	

5 rows × 27 columns

In [6]:

```
print(train.shape)
```

```
print(train.shape)
print(test.shape)
```

```
(100000, 28)
(50000, 27)
```

Exploratory Data Analysis and Visualization

In [7]:

```
df = train
df.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 [8]:

```
df.shape
```

Out[8]:

```
(100000, 28)
```

In [9]:

```
df.duplicated().sum()
```

Out[9]:

```
0
```

In [10]:

```
df.describe().T
```

Out[10]:

	count	mean	std	min	25%	50%	75%	max
Monthly_Inhand_Salary	84998.000	4194.171	3183.686	303.645	1625.568	3093.745	5957.448	15204.633
Num_Bank_Accounts	100000.000	17.091	117.405	-1.000	3.000	6.000	7.000	1798.000
Num_Credit_Card	100000.000	22.474	129.057	0.000	4.000	5.000	7.000	1499.000
Interest_Rate	100000.000	72.466	466.423	1.000	8.000	13.000	20.000	5797.000
Delay_from_due_date	100000.000	21.069	14.860	-5.000	10.000	18.000	28.000	67.000
Num_Credit_Inquiries	98035.000	27.754	193.177	0.000	3.000	6.000	9.000	2597.000
Credit_Utilization_Ratio	100000.000	32.285	5.117	20.000	28.053	32.306	36.497	50.000
Total_EMI_per_month	100000.000	1403.118	8306.041	0.000	30.307	69.249	161.224	82331.000

In [11]:

```
df.describe(include='object').T
```

Out[11]:

	count	unique	top	freq
ID	100000	100000	0x1602	1
Customer_ID	100000	12500	CUS_0xd40	8
Month	100000	8	January	12500
Name	90015	10139	Langep	44
Age	100000	1788	38	2833
SSN	100000	12501	#F%$D@*&8	5572
Occupation	100000	16	_______	7062
Annual_Income	100000	18940	36585.12	16
Num_of_Loan	100000	434	3	14386
Type_of_Loan	88592	6260	Not Specified	1408
Num_of_Delayed_Payment	92998	749	19	5327
Changed_Credit_Limit	100000	4384	_	2091
Credit_Mix	100000	4	Standard	36479
Outstanding_Debt	100000	13178	1360.45	24
Credit_History_Age	90970	404	15 Years and 11 Months	446
Payment_of_Min_Amount	100000	3	Yes	52326
Amount_invested_monthly	95521	91049	__10000__	4305
Payment_Behaviour	100000	7	Low_spent_Small_value_payments	25513
Monthly_Balance	98800	98792	__-333333333333333333333333333__	9
Credit_Score	100000	3	Standard	53174

Features and Data Cleaning

In [12]:

```
df.isnull().sum()
```

Out[12]:

```
ID                             0
Customer_ID                    0
Month                          0
Name                        9985
Age                            0
SSN                            0
Occupation                     0
```

```
Annual_Income                 0
Monthly_Inhand_Salary     15002
Num_Bank_Accounts             0
Num_Credit_Card               0
Interest_Rate                 0
Num_of_Loan                   0
Type_of_Loan              11408
Delay_from_due_date           0
Num_of_Delayed_Payment     7002
Changed_Credit_Limit          0
Num_Credit_Inquiries       1965
Credit_Mix                    0
Outstanding_Debt              0
Credit_Utilization_Ratio      0
Credit_History_Age         9030
Payment_of_Min_Amount         0
Total_EMI_per_month           0
Amount_invested_monthly    4479
Payment_Behaviour             0
Monthly_Balance            1200
Credit_Score                  0
dtype: int64
```

In [13]:

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

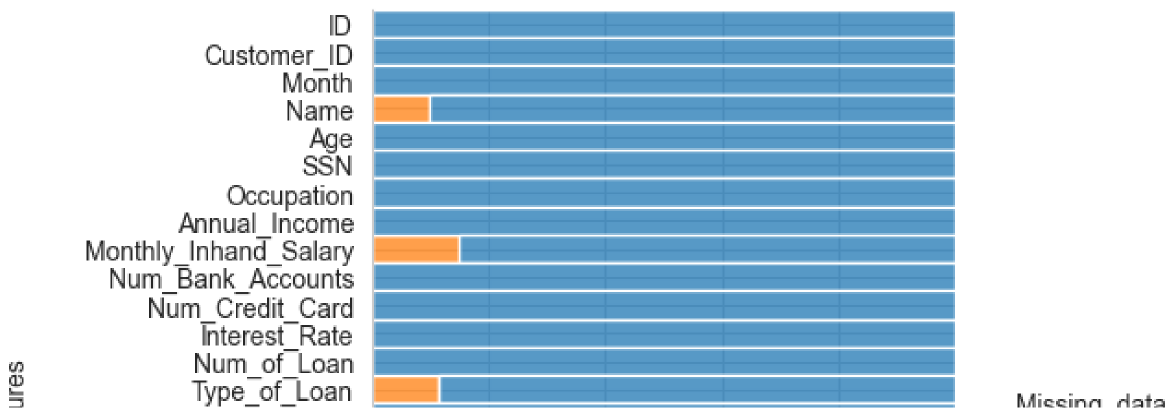
Out[13]:

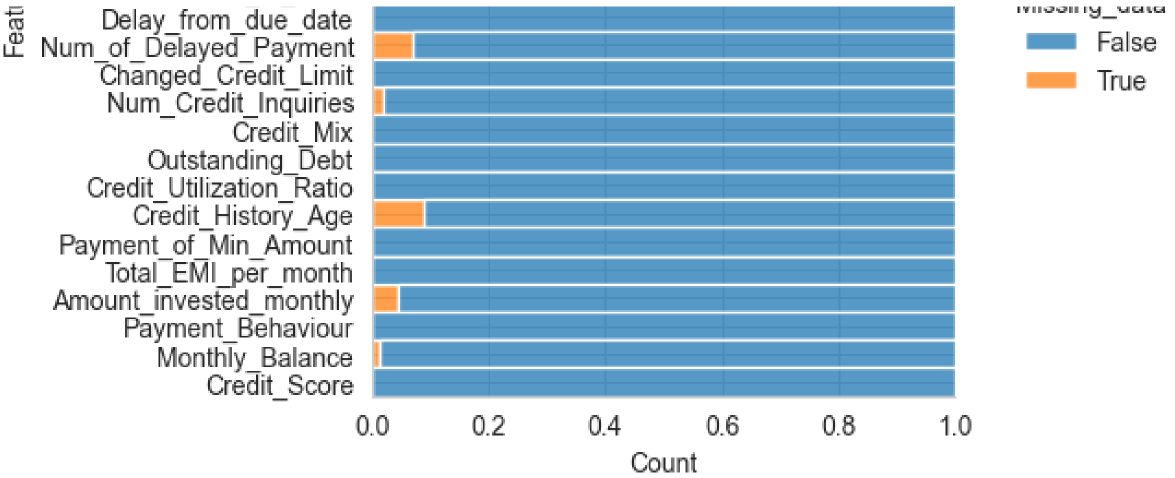
```
Name                        9985
Monthly_Inhand_Salary      15002
Type_of_Loan               11408
Num_of_Delayed_Payment      7002
Num_Credit_Inquiries        1965
Credit_History_Age          9030
Amount_invested_monthly     4479
Monthly_Balance             1200
dtype: int64
```

In [14]:

```
def na_ratio_plot(df=df):

    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])
na_ratio_plot()
```

```
Name                        9985
Monthly_Inhand_Salary      15002
Type_of_Loan               11408
Num_of_Delayed_Payment      7002
Num_Credit_Inquiries        1965
Credit_History_Age          9030
Amount_invested_monthly     4479
Monthly_Balance             1200
dtype: int64
```





Data Cleaning

Month

In [15]:

```
df.Month.value_counts()
```

Out[15]:

```
Month
January      12500
February     12500
March        12500
April        12500
May          12500
June         12500
July         12500
August       12500
Name: count, dtype: int64
```

In [16]:

```
df.Month.unique()
```

Out[16]:

```
array(['January', 'February', 'March', 'April', 'May', 'June', 'July',
       'August'], dtype=object)
```

In [17]:

```
dict = {"January" : 1,"February" : 2,"March" : 3,"April" : 4,"May" : 5,"June" : 6,"July"
: 7,"August" : 8}
df["Month"] = df["Month"].map(dict)
```

Age

In [18]:

```
# 'Age'
def clean_age(age):
    try:
        return int(age)
    except ValueError:
        return None

df['Age'] = df['Age'].str.replace('_', '').str.replace('-', '')
df['Age'] = df['Age'].apply(clean_age)
```

In [19]:

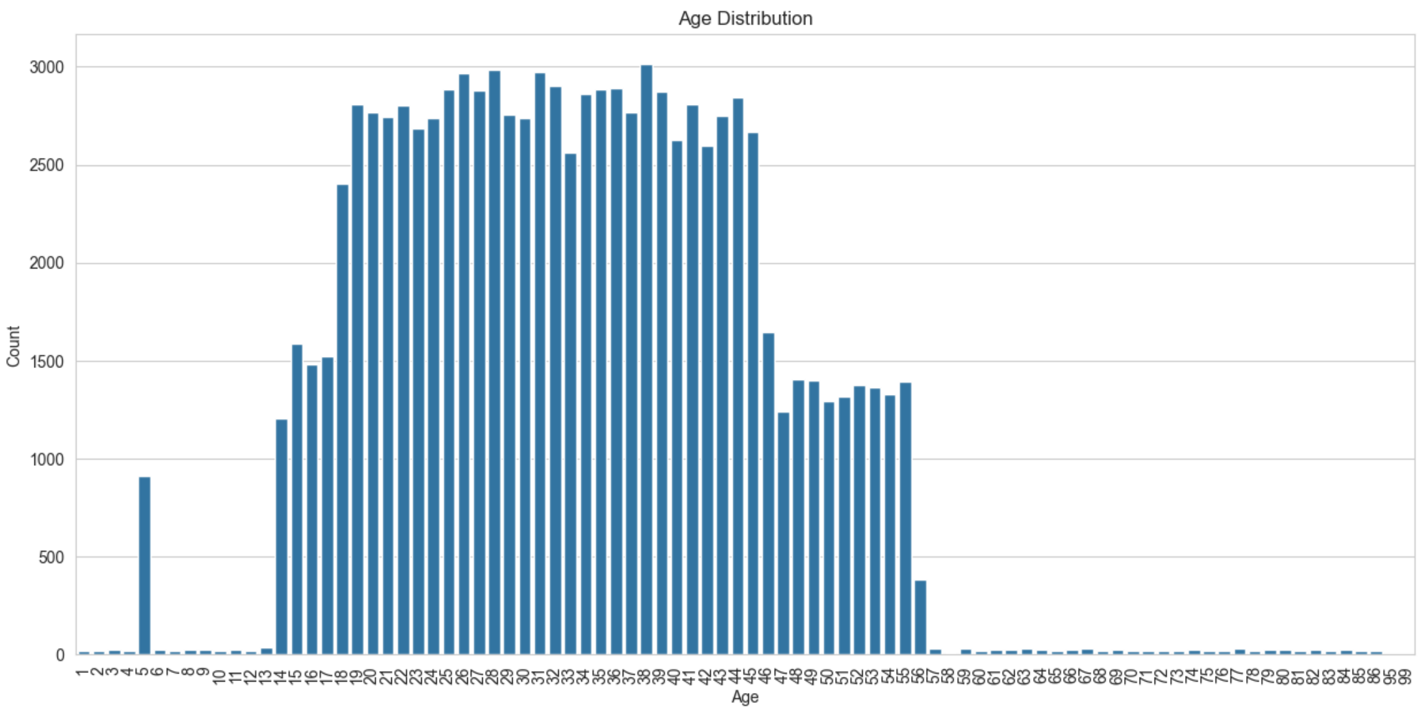
```
def truncate_last_two_digits(age):
    if age > 99:
        return age // 100
    else:
        return age


df['Age'] = df['Age'].apply(truncate_last_two_digits)


df.Age
```

Out[19]:

```
0        23
1        23
2         5
3        23
4        23
         ..
99995    25
99996    25
99997    25
99998    25
99999    25
Name: Age, Length: 100000, dtype: int64
```

In [20]:

```
plt.figure(figsize=(15, 7))
sns.countplot(x='Age', data=df)
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.xticks(rotation=90)
plt.show()
```

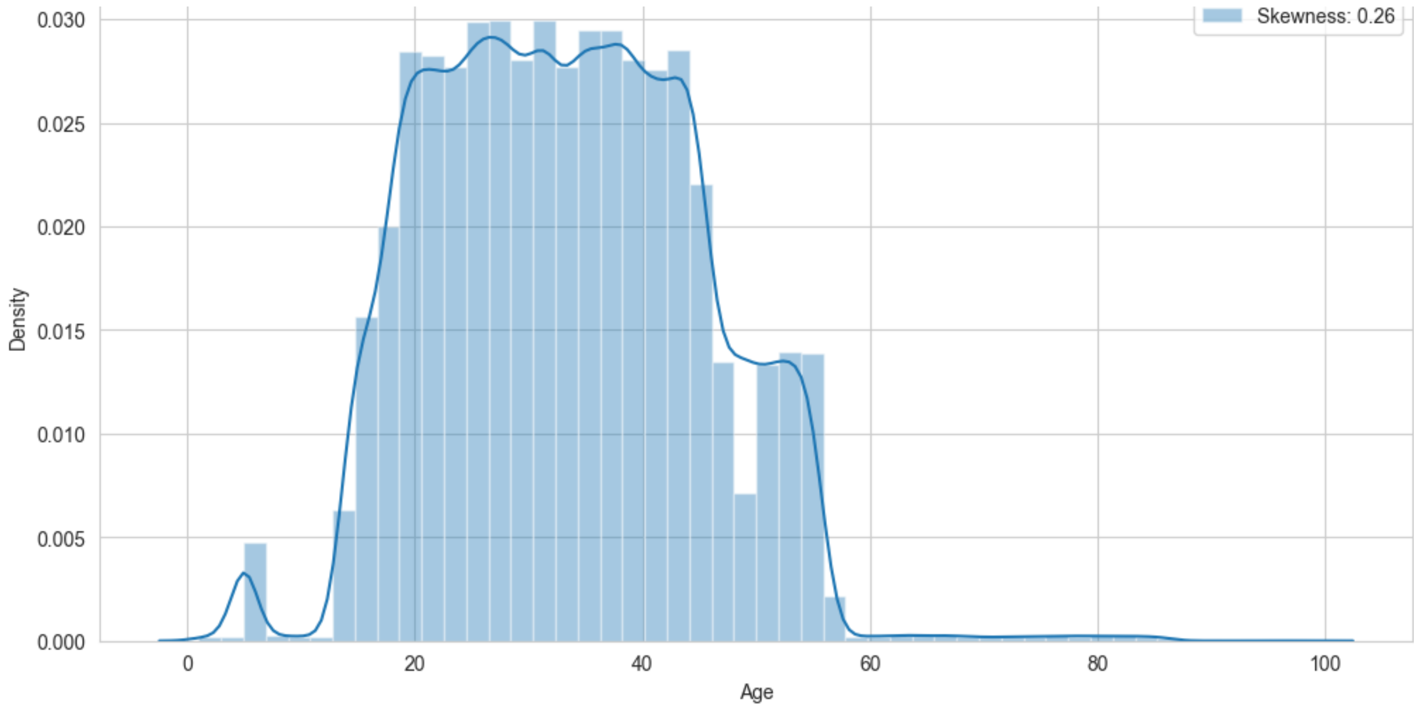


In [21]:

```
sns.distplot(df['Age'], label = 'Skewness: %.2f'%(df['Age'].skew()))
plt.legend(loc = 'best')
plt.title('Customer Age Distribution')
```

Out[21]:

```
Text(0.5, 1.0, 'Customer Age Distribution')
```



Annual_Incame

In [22]:

```
# To remove the tire at the end
def remove_trailing_dash(value):
    if isinstance(value, str) and value.endswith('_'):
        return value[:-1]  # Son karakteri (tireyi) kaldır
    else:
        return value


df['Annual_Income'] = df['Annual_Income'].apply(remove_trailing_dash)
```

In [23]:

```
df['Annual_Income'] = df['Annual_Income'].astype(float)
```

In [24]:

```
df['Annual_Income'].unique()
```

Out[24]:

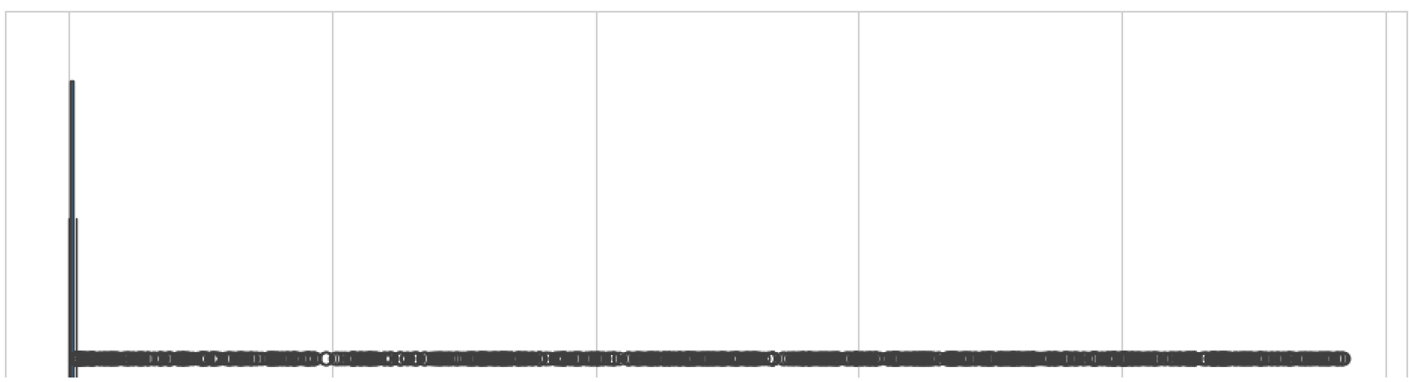
```
array([ 19114.12,  34847.84, 143162.64, ...,  37188.1 ,  20002.88,
        39628.99])
```

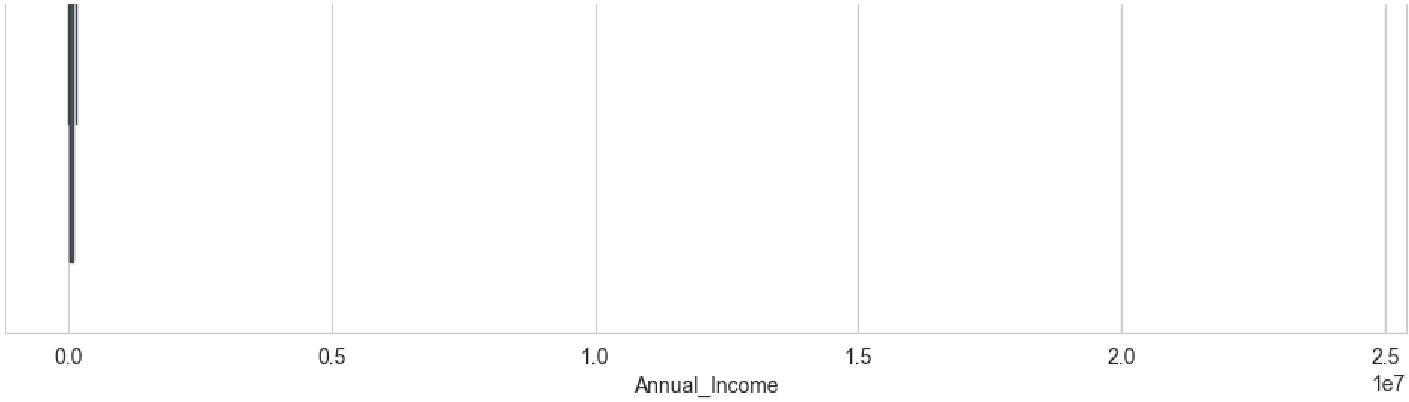
In [25]:

```
sns.boxplot(x = 'Annual_Income', data = df)
```

Out[25]:

```
<Axes: xlabel='Annual_Income'>
```





Monthly_Inhand_Salary

In [26]:

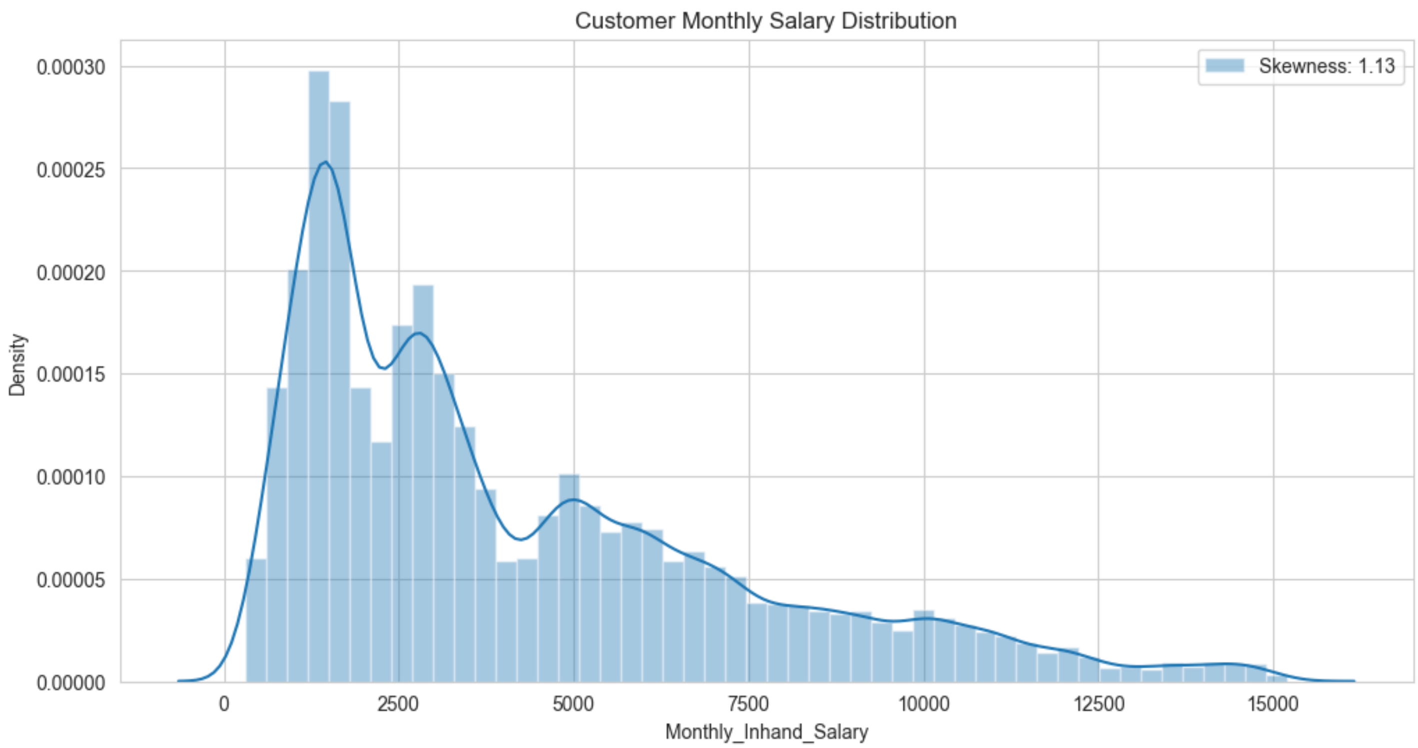
```
Customer_Mode_Salary = df.groupby('Customer_ID')['Monthly_Inhand_Salary'].transform(lambd
a x : x.mode().iloc[0])
df['Monthly_Inhand_Salary'] = np.where(df['Monthly_Inhand_Salary'].isnull(), Customer_Mo
de_Salary, df['Monthly_Inhand_Salary'])
```

In [27]:

```
sns.distplot(df['Monthly_Inhand_Salary'], label = 'Skewness: %.2f'%(df['Monthly_Inhand_Sa
lary'].skew()))
plt.legend(loc = 'best')
plt.title('Customer Monthly Salary Distribution')
```

Out[27]:

```
Text(0.5, 1.0, 'Customer Monthly Salary Distribution')
```



Occupation'

In [28]:

```
def fill_occupation_by_ssn(df):
    # Replace '_______' values in 'Occupation' column with NaN (empty) values
    df['Occupation'] = df['Occupation'].replace('_______', np.nan)

    # Find the most recurring 'Occupation' values for each SNN number
    most_common_occupation_by_ssn = df.groupby('SSN')['Occupation'].apply(lambda x: x.mo
```

```
de().iloc[0])

    # 'Populating '_______' values in 'Occupation' column
    for index, row in df.iterrows():
        if pd.isnull(row['Occupation']) and row['SSN'] in most_common_occupation_by_ssn:
            df.at[index, 'Occupation'] = most_common_occupation_by_ssn[row['SSN']]


fill_occupation_by_ssn(df)
```

In [29]:

```
occupation_count = df['Occupation'].value_counts()
occupation_count
```

Out[29]:

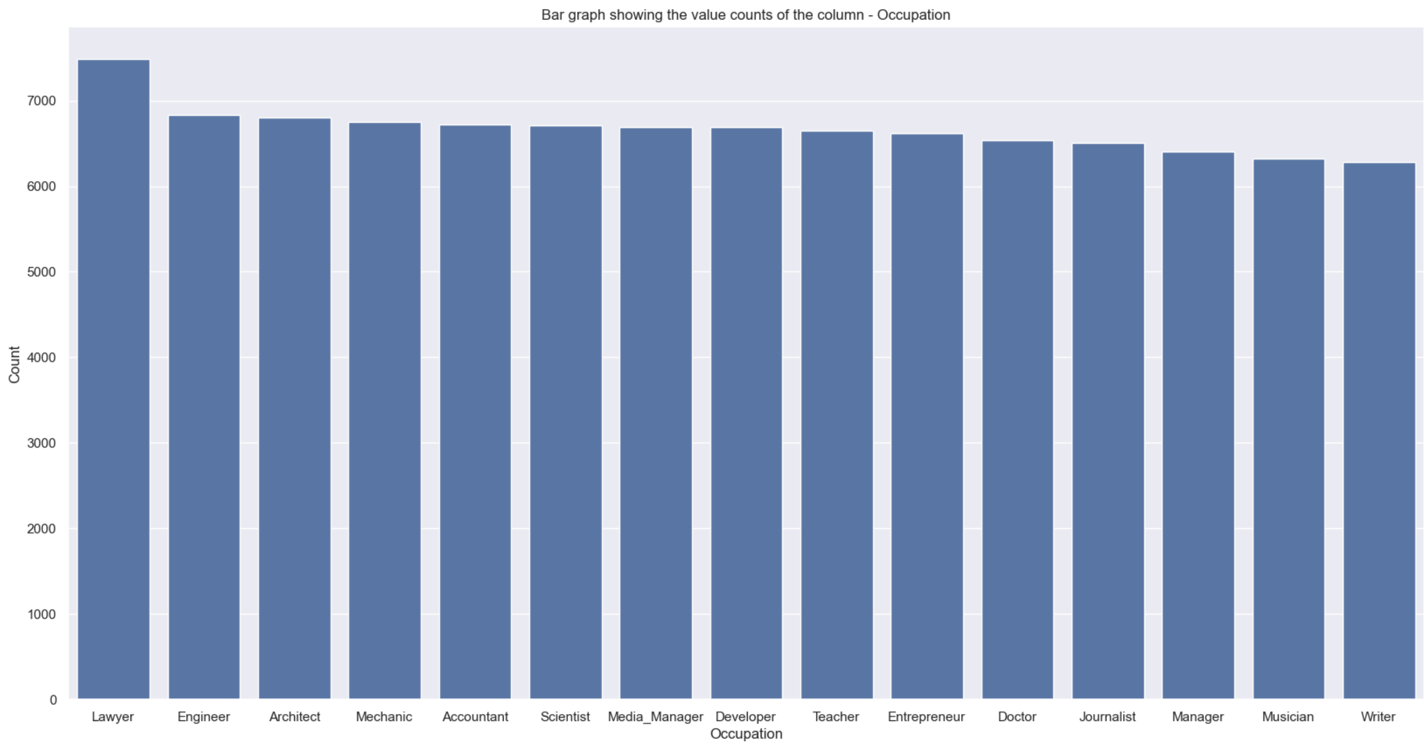
```
Occupation
Lawyer           7489
Engineer         6837
Architect        6806
Mechanic         6752
Accountant       6717
Scientist        6713
Media_Manager    6689
Developer        6687
Teacher          6646
Entrepreneur     6621
Doctor           6537
Journalist       6502
Manager          6402
Musician         6322
Writer           6280
Name: count, dtype: int64
```

In [30]:

```
# occupation_count, chart with the number of occupations in the Occupation column
sns.set(rc={'figure.figsize': (20, 10)})
sns.barplot(x=occupation_count.index, y=occupation_count.values)
plt.title('Bar graph showing the value counts of the column - Occupation')
plt.ylabel('Count', fontsize=12)
plt.xlabel('Occupation', fontsize=12)
```

Out[30]:

```
Text(0.5, 0, 'Occupation')
```

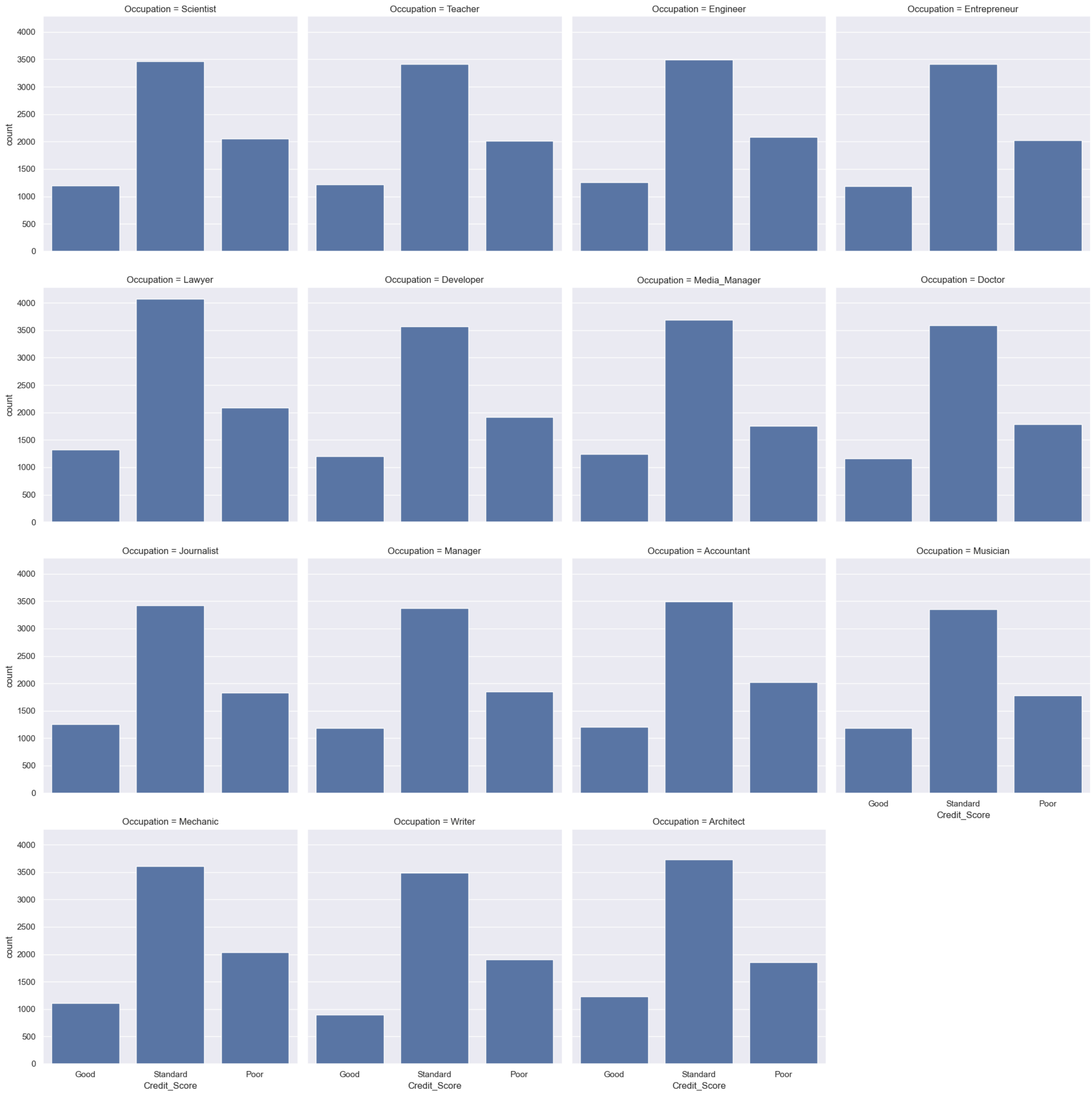


In [31]:

```
sns.catplot(x='Credit_Score', col='Occupation', data=df, kind='count', col_wrap=4)
```

Out[31]:

```
&lt;seaborn.axisgrid.FacetGrid at 0x2b2d72d3860&gt;
```



Num_of_Loan

In [32]:

```
df['Num_of_Loan'].unique()
```

Out[32]:

```
array(['4', '1', '3', '967', '-100', '0', '0_', '2', '3_', '2_', '7', '5',
       '5_', '6', '8', '8_', '9', '9_', '4_', '7_', '1_', '1464', '6_',
       '622', '352', '472', '1017', '945', '146', '563', '341', '444',
       '720', '1485', '49', '737', '1106', '466', '728', '313', '843',
       '597_', '617', '119', '663', '640', '92_', '1019', '501', '1302',
       '39', '716', '848', '931', '1214', '186', '424', '1001', '1110',
       '1152', '457', '1433', '1187', '52', '1480', '1047', '1035',
       '1347_', '33', '193', '699', '329', '1451', '484', '132', '649',
       '995', '545', '684', '1135', '1094', '1204', '654', '58', '348',
```

```
       '614', '1363', '323', '1406', '1348', '430', '153', '1461', '905',
       '1312', '1424', '1154', '95', '1353', '1228', '819', '1006', '795',
       '359', '1209', '590', '696', '1185_', '1465', '911', '1181', '70',
       '816', '1369', '143', '1416', '455', '55', '1096', '1474', '420',
       '1131', '904', '89', '1259', '527', '1241', '449', '983', '418',
       '319', '23', '238', '638', '138', '235_', '280', '1070', '1484',
       '274', '494', '1459_', '404', '1354', '1495', '1391', '601',
       '1313', '1319', '898', '231', '752', '174', '961', '1046', '834',
       '284', '438', '288', '1463', '1151', '719', '198', '1015', '855',
       '841', '392', '1444', '103', '1320_', '745', '172', '252', '630_',
       '241', '31', '405', '1217', '1030', '1257', '137', '157', '164',
       '1088', '1236', '777', '1048', '613', '330', '1439', '321', '661',
       '952', '939', '562', '1202', '302', '943', '394', '955', '1318',
       '936', '781', '100', '1329', '1365', '860', '217', '191', '32',
       '282', '351', '1387', '757', '416', '833', '359_', '292', '1225_',
       '1227', '639', '859', '243', '267', '510', '332', '996', '597',
       '311', '492', '820', '336', '123', '540', '131_', '1311_', '1441',
       '895', '891', '50', '940', '935', '596', '29', '1182', '1129_',
       '1014', '251', '365', '291', '1447', '742', '1085', '148', '462',
       '832', '881', '1225', '1412', '785_', '1127', '910', '538', '999',
       '733', '101', '237', '87', '659', '633', '387', '447', '629',
       '831', '1384', '773', '621', '1419', '289', '143_', '285', '1393',
       '1131_', '27_', '1359', '1482', '1189', '1294', '201', '579',
       '814', '141', '1320', '581', '1171_', '295', '290', '433', '679',
       '1040', '1054', '1430', '1023', '1077', '1457', '1150', '701',
       '1382', '889', '437', '372', '1222', '126', '1159', '868', '19',
       '1297', '227_', '190', '809', '1216', '1074', '571', '520', '1274',
       '1340', '991', '316', '697', '926', '873', '1002', '378_', '65',
       '875', '867', '548', '652', '1372', '606', '1036', '1300', '17',
       '1178', '802', '1219_', '1271', '1137', '1496', '439', '196',
       '636', '192', '228', '1053', '229', '753', '1296', '1371', '254',
       '863', '464', '515', '838', '1160', '1289', '1298', '799', '182',
       '574', '527_', '242', '415', '869', '958', '54', '1265', '656',
       '275', '778', '208', '147', '350', '507', '463', '497', '1129',
       '927', '653', '662', '529', '635', '1027_', '897', '1039', '227',
       '1345', '924', '696_', '1279', '546', '1112', '1210', '526', '300',
       '1103', '504', '136', '1400', '78', '686', '1091', '344', '215',
       '84', '628', '1470', '968', '1478', '83', '1196', '1307', '1132_',
       '1008', '917', '657', '56', '18', '41', '801', '978', '216', '349',
       '966'], dtype=object)
```

In [33]:

```
# Function to remove "-" and "_" characters
def clean_num(num):
    num = num.strip("-_")
    if num == "100":
        return np.nan
    elif len(num) > 1:
        return num[0]
    else:
        return num


df["Num_of_Loan"] = df["Num_of_Loan"].apply(clean_num)


most_common_value = df["Num_of_Loan"].mode()[0]
df["Num_of_Loan"] = df["Num_of_Loan"].fillna(most_common_value)
```

In [34]:

```
df.Num_of_Loan.value_counts()
```

Out[34]:

```
Num_of_Loan
3    19016
2    15076
4    14776
0    10930
1    10800
```

```
6     7839
7     7368
5     7231
9     3736
8     3228
Name: count, dtype: int64
```

In [35]:

```
df.Num_of_Loan.isnull().sum()
```

Out[35]:

```
0
```

Type_of_Loan

In [36]:

```
df['Type_of_Loan'].fillna('Unknown', inplace=True)
```

In [37]:

```
loan_type_groups = df.groupby('Type_of_Loan').size()
print(loan_type_groups)
```

```
Type_of_Loan
Auto Loan
1152
Auto Loan, Auto Loan, Auto Loan, Auto Loan, Credit-Builder Loan, Credit-Builder Loan, Mor
tgage Loan, and Personal Loan                                   8
Auto Loan, Auto Loan, Auto Loan, Auto Loan, Student Loan, and Student Loan
8
Auto Loan, Auto Loan, Auto Loan, Credit-Builder Loan, Payday Loan, Not Specified, Payday
Loan, Student Loan, and Debt Consolidation Loan         8
Auto Loan, Auto Loan, Auto Loan, Not Specified, Debt Consolidation Loan, and Credit-Build
er Loan                                                         8

...
Student Loan, and Not Specified
160
Student Loan, and Payday Loan
256
Student Loan, and Personal Loan
176
Student Loan, and Student Loan
216
Unknown
11408
Length: 6261, dtype: int64
```

In [38]:

```
df.Type_of_Loan.value_counts()
```

Out[38]:

```
Type_of_Loan
Unknown
11408
Not Specified
1408
Credit-Builder Loan
1280
Personal Loan
1272
Debt Consolidation Loan
1264

...
Not Specified, Mortgage Loan, Auto Loan, and Payday Loan
```

```
8
Payday Loan, Mortgage Loan, Debt Consolidation Loan, and Student Loan
8
Debt Consolidation Loan, Auto Loan, Personal Loan, Debt Consolidation Loan, Student Loan,
and Credit-Builder Loan                              8
Student Loan, Auto Loan, Student Loan, Credit-Builder Loan, Home Equity Loan, Debt Consol
idation Loan, and Debt Consolidation Loan         8
Personal Loan, Auto Loan, Mortgage Loan, Student Loan, and Student Loan
8
Name: count, Length: 6261, dtype: int64
```

Num_of_Delayed_Payment

In [39]:

```
df['Num_of_Delayed_Payment'] = df['Num_of_Delayed_Payment'].fillna('0')
```

In [40]:

```
def remove_special_characters(value):
    if isinstance(value, str):

        value = value.strip('_').strip('-')
    return value
```

In [41]:

```
df['Num_of_Delayed_Payment'] = df['Num_of_Delayed_Payment'].apply(
    remove_special_characters)
```

Changed_Credit_Limit

In [42]:

```
df['Changed_Credit_Limit'] = df['Changed_Credit_Limit'].replace('-', np.nan)
df['Changed_Credit_Limit'] = df['Changed_Credit_Limit'].replace('_', np.nan)


df['Changed_Credit_Limit'] = pd.to_numeric(df['Changed_Credit_Limit'], errors='coerce')


mean_value = df['Changed_Credit_Limit'].mean()


df['Changed_Credit_Limit'].fillna(mean_value, inplace=True)
```

Num_Credit_Inquiries

In [43]:

```
df['Num_Credit_Inquiries'].unique()
```

Out[43]:

```
array([   4.,     2.,     3., ..., 1361.,   310.,    74.])
```

In [44]:

```
df['Num_Credit_Inquiries'].fillna(0, inplace=True)
```

Credit_Mix

In [45]:

```
credit_mix_count = df['Credit_Mix'].value_counts()
credit_mix_count
```

Out[45]:

```
Credit_Mix
Standard    36479
Good        24337
_           20195
Bad         18989
Name: count, dtype: int64
```

In [46]:

```
df['Credit_Mix'] = df['Credit_Mix'].replace('_', np.nan)

def fill_na_cat(data, val):

    for col in data.select_dtypes(include='object').columns:
        mode_by_customer = data.groupby('Customer_ID')[col].transform(lambda x: x.mode()
[0] if not x.mode().empty else np.nan)
        mode_global = data[col].mode()[0]
        data[col] = data[col].fillna(mode_by_customer.fillna(mode_global))
    return data

df = fill_na_cat(data=df, val="Credit_Mix")
```

In [47]:

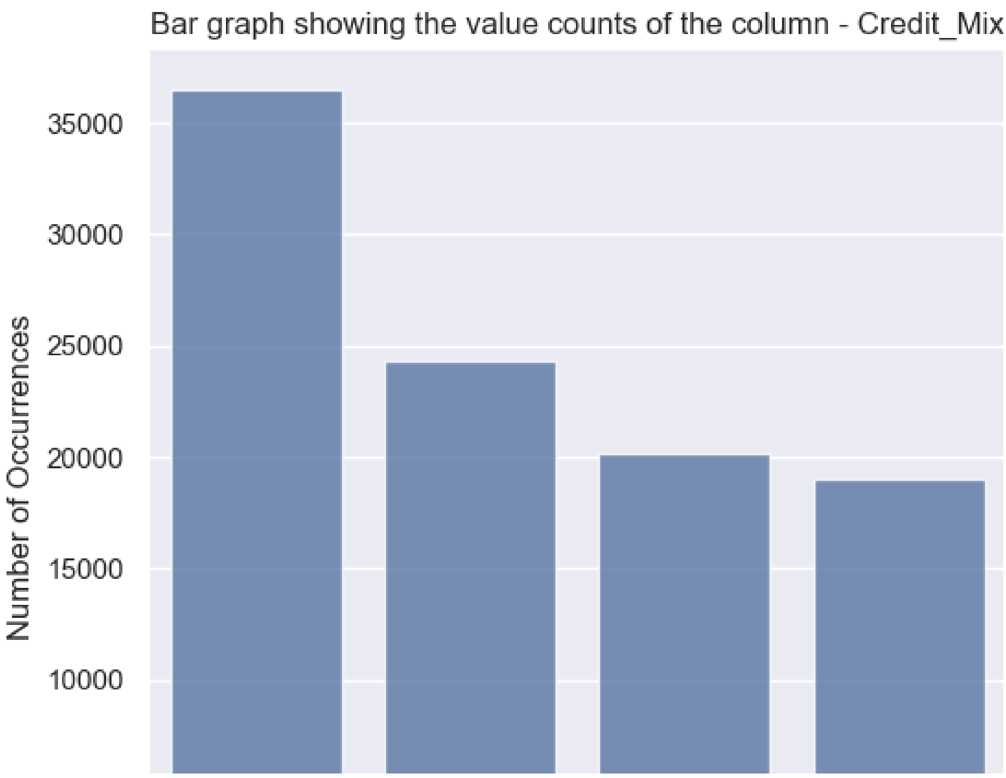
```
df['Credit_Mix'].value_counts()
```

Out[47]:

```
Credit_Mix
Standard    45848
Good        30384
Bad         23768
Name: count, dtype: int64
```

In [48]:

```
# Bar graph showing the value counts of the column - Credit_Mix

sns.set(rc={'figure.figsize': (6, 6)})
sns.barplot(x=credit_mix_count.index, y=credit_mix_count.values, alpha=0.8)
plt.title('Bar graph showing the value counts of the column - Credit_Mix')
plt.ylabel('Number of Occurrences', fontsize=12)
plt.xlabel('Credit Mix', fontsize=12)
plt.show()
```



In [49]:

```
credit_mix_count = df['Credit_Mix'].value_counts()
credit_mix_count
```

Out[49]:

```
Credit_Mix
Standard     45848
Good         30384
Bad          23768
Name: count, dtype: int64
```

In [50]:

```
df.sample(15)
```

Out[50]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Banl
26012	0xae6a	CUS_0x5ebe	5	Angelai	46	889-60-8908	Mechanic	19731.160	1887.263	
31261	0xcd2b	CUS_0xbdd9	6	Chiangy	15	491-09-0120	Scientist	31785.100	2410.758	
97019	0x24e79	CUS_0x45e2	4	Lewis Krauskopfd	25	264-39-4694	Teacher	59683.100	5198.592	
10617	0x5437	CUS_0x2c99	2	Qing Xiaoyig	19	180-82-7743	Teacher	9648.220	931.018	
89793	0x22423	CUS_0xa4e7	2	Lynchj	41	625-58-3829	Musician	17481.785	1736.815	
21778	0x959c	CUS_0x4409	3	Elzio Barreton	16	024-84-1689	Writer	82899.520	7069.293	
80041	0x1eaff	CUS_0x131f	2	Silke Koltrowitzl	45	006-11-7238	Developer	68105.600	5636.467	
39540	0xfdae	CUS_0x14d9	5	Subhedarm	36	225-59-9039	Writer	76082.250	6082.188	
90924	0x22ac2	CUS_0x2048	5	Dmitryo	40	212-55-0432	Developer	20934.490	1619.541	
97650	0x2522c	CUS_0x95d0	3	Ingramz	24	412-45-1812	Manager	105449.340	9012.445	
95790	0x24744	CUS_0xbfa6	7	Espanaa	37	006-73-8209	Teacher	74036.960	6384.747	
13719	0x6661	CUS_0x45f1	8	Charleso	49	900-25-3681	Writer	41235.330	3384.278	
				Gertrude		631-				

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank
17290	0x7b50	CUS_0x812c	3	Chavez-Dreyfusse	35	45-4165	Engineer	65644.400	5275.367	
42727	0x11059	CUS_0xafa3	8	Alwyn Scotts	26	484-09-6362	Engineer	28347.680	2091.307	
89614	0x22314	CUS_0x7500	7	Tim Hephero	33	841-44-2527	Scientist	166621.840	13719.153	

15 rows × 28 columns

Credit_History_Age

In [51]:

```
# Covert Credit_History_Age to month
def parse_years_and_months_to_months(age):
    if isinstance(age, str):
        age_parts = age.split(' Years and ')
        years = int(age_parts[0]) if 'Years' in age else 0
        months_str = age_parts[1].split(' Months')[0] if 'Months' in age_parts[1] else '
0'
        months = int(months_str)
        total_months = years * 12 + months
        return total_months
    else:
        return 0


df['Credit_History_Age_Months'] = df['Credit_History_Age'].apply(parse_years_and_months_
to_months)
```

In [52]:

```
df.drop(columns=['Credit_History_Age'], inplace=True)
```

Amount_invested_monthly

In [53]:

```
df['Amount_invested_monthly'] = df['Amount_invested_monthly'].replace(
    '__10000__', np.nan)

df['Amount_invested_monthly'] = df.groupby(
    'Customer_ID')['Amount_invested_monthly'].transform(
        lambda x: x.mode()[0] if not x.mode().empty else np.NaN)
```

In [54]:

```
df['Amount_invested_monthly'].isnull().sum()
```

Out[54]:

```
0
```

Monthly_Balance

In [55]:

```
df['Monthly_Balance'].value_counts()
```

Out[55]:

```
Monthly_Balance
 __-333333333333333333333333333__     9
 239.59620527100063                   6
 212.44492522535438                   5
```

```
 212.44492522555458                5
 128.54140433011784                5
 244.99107777431962                5
                                  ..
 658.9412372133265                 1
 1072.867912409617                 1
 740.0418087099729                 1
 1279.6106996658787                1
393.674                            1
Name: count, Length: 98792, dtype: int64
```

In [56]:

```
df['Monthly_Balance'] = df['Monthly_Balance'].astype(str)

df['Monthly_Balance'] = df['Monthly_Balance'].str.replace(r'[^0-9.-]+', '').str.replace(
'_', '').str.replace('-', '')
```

In [57]:

```
df['Monthly_Balance'] = df['Monthly_Balance'].astype(float)

mean_value = df['Monthly_Balance'].mean()

df['Monthly_Balance'].fillna(mean_value, inplace=True)
```

In [58]:

```
df['Monthly_Balance'].value_counts()
```

Out[58]:

```
Monthly_Balance
3333333333333333314856026112.000    9
239.596                             6
212.445                             5
128.541                             5
244.991                             5
                                   ..
658.941                             1
1072.868                            1
740.042                             1
1279.611                            1
393.674                             1
Name: count, Length: 98792, dtype: int64
```

Payment_Behaviour

In [59]:

```
df['Payment_Behaviour'] = df['Payment_Behaviour'].replace('!@9#%8', np.nan)
```

In [60]:

```
df['Payment_Behaviour'].isna().sum()
```

Out[60]:

```
7600
```

In [61]:

```
# Group data by 'Payment Behaviour' column
grouped_df = df.groupby('Payment_Behaviour').size()

print(grouped_df)
```

```
Payment_Behaviour
High_spent_Large_value_payments     13721
High_spent_Medium_value_payments    17540
High_spent_Small_value_payments     11340
Low_spent_Large_value_payments      10425
```

```
Low_spent_Large_value_payments      10425
Low_spent_Medium_value_payments     13861
Low_spent_Small_value_payments      25513
dtype: int64
```

In [62]:

```python
df['Payment_Behaviour'] = df['Payment_Behaviour'].fillna(method='ffill')
```

In [63]:

```python
df['Payment_Behaviour'].isna().sum()
```

Out[63]:

```
0
```

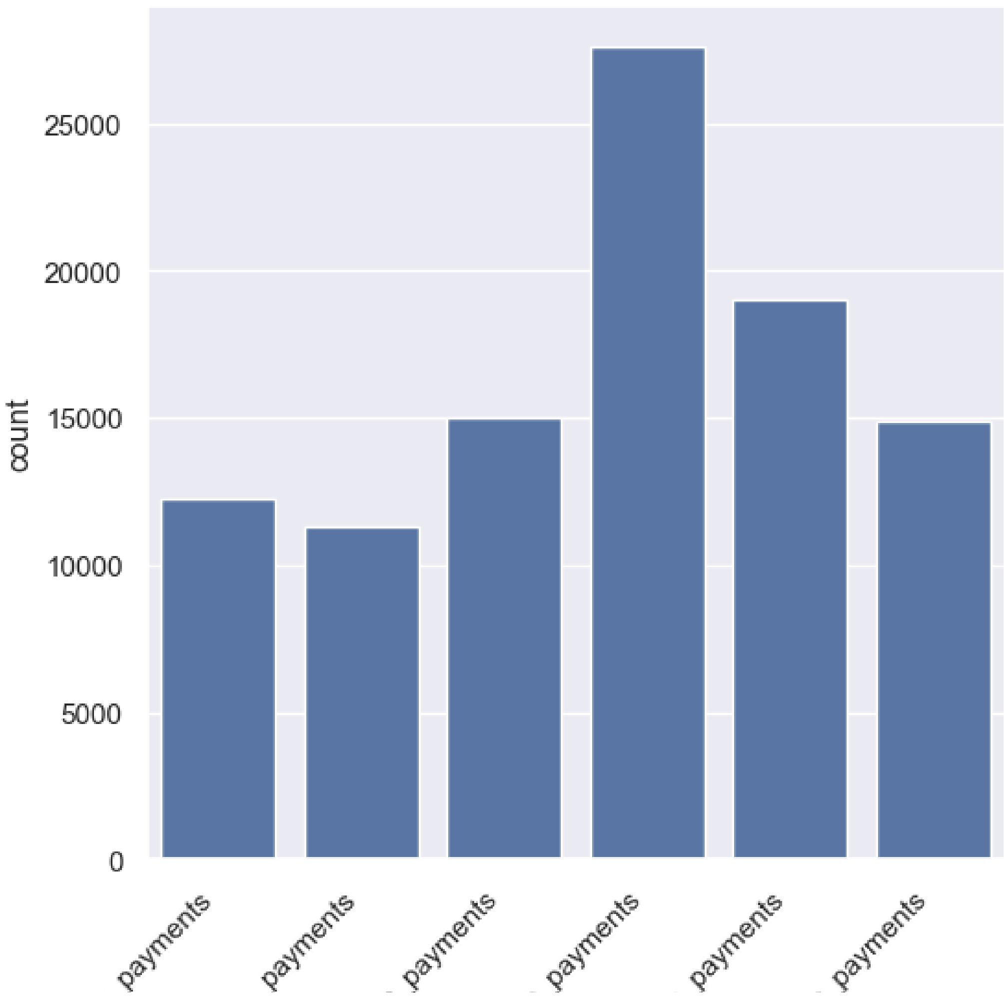
In [64]:

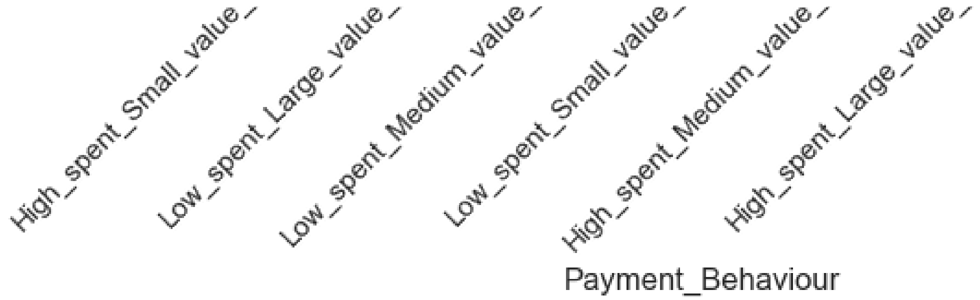
```python
grouped_df = df.groupby('Payment_Behaviour').size()

print(grouped_df)
```

```
Payment_Behaviour
High_spent_Large_value_payments     14863
High_spent_Medium_value_payments    19010
High_spent_Small_value_payments     12250
Low_spent_Large_value_payments      11297
Low_spent_Medium_value_payments     14987
Low_spent_Small_value_payments      27593
dtype: int64
```

In [65]:

```python
plot = sns.countplot(x='Payment_Behaviour', data=df)

plot.set_xticklabels(plot.get_xticklabels(), rotation=45, ha='right')

plt.show()
```





Payment_of_Min_Amount

In [66]:

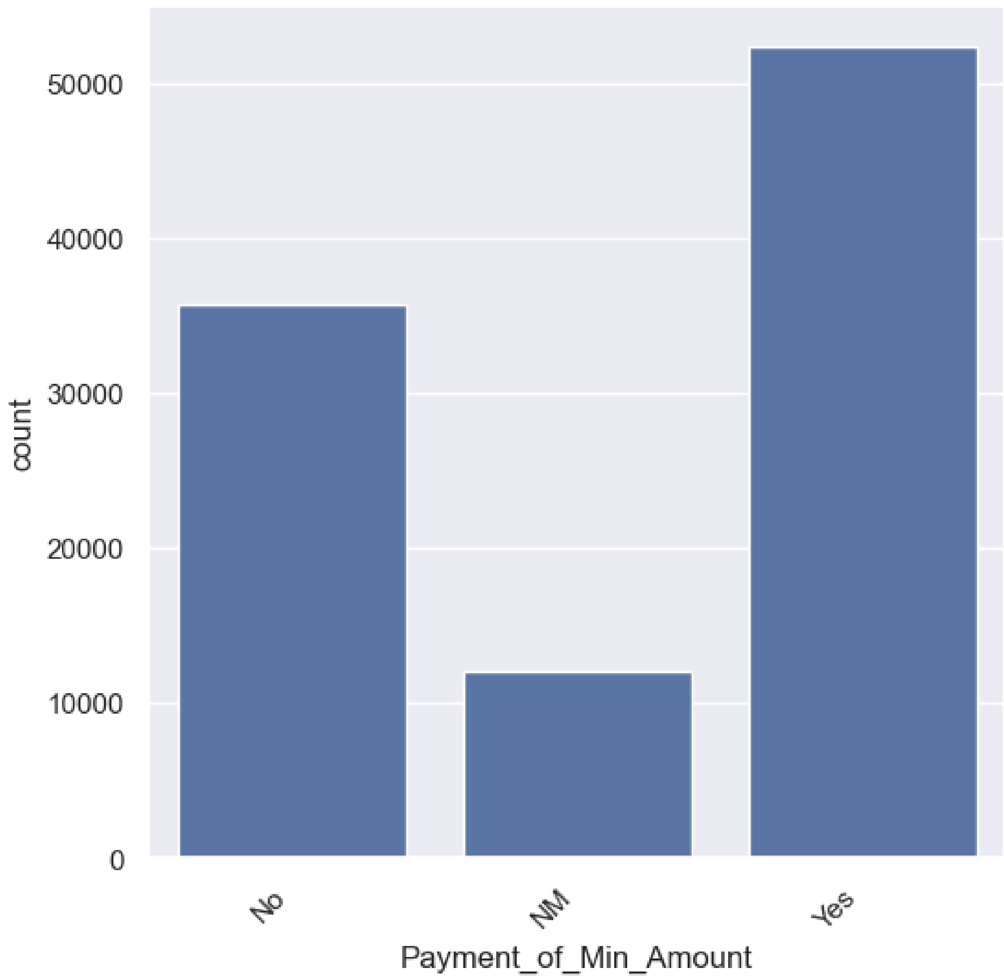
```
min_amount_count = df['Payment_of_Min_Amount'].value_counts()
min_amount_count
```

Out[66]:

```
Payment_of_Min_Amount
Yes     52326
No      35667
NM      12007
Name: count, dtype: int64
```

In [67]:

```
plot = sns.countplot(x='Payment_of_Min_Amount', data=df)

plot.set_xticklabels(plot.get_xticklabels(), rotation=45, ha='right')   # ha='right' ile
etiketlerin hizalanması sağlanır

plt.show()
```



Interest_Rate¶

In [68]:

```
df['Interest_Rate'] = df['Interest_Rate'].astype(float)
```

In [69]:

```
df.Interest_Rate.value_counts()
```

Out[69]:

```
Interest_Rate
8.000       5012
5.000       4979
6.000       4721
12.000      4540
10.000      4540
            ...
4995.000       1
1899.000       1
2120.000       1
5762.000       1
5729.000       1
Name: count, Length: 1750, dtype: int64
```

Drop unnecessary columns

In [70]:

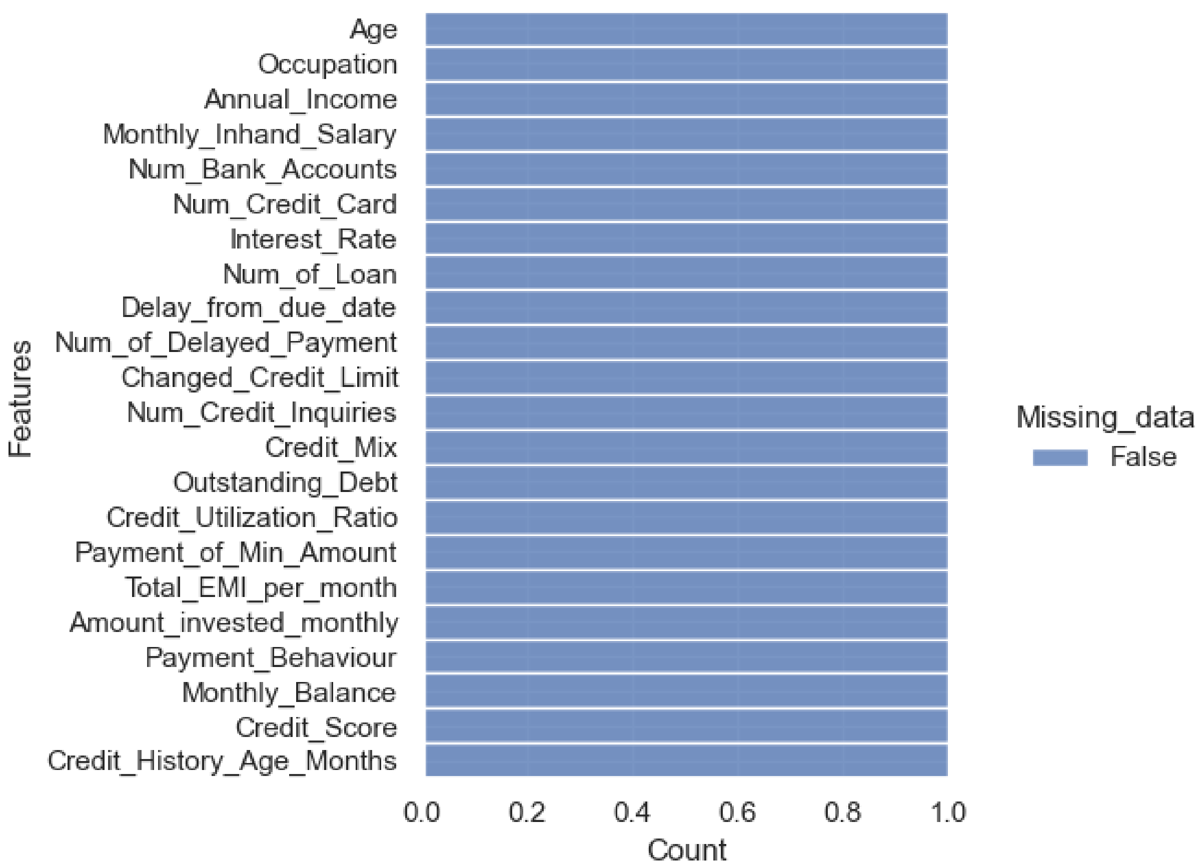
```
df.drop(['ID','Customer_ID', 'Month', 'Name','SSN', 'Type_of_Loan'], axis = 1, inplace =
True)
```

Convert numeric columns dtypes

In [71]:

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

```
Series([], dtype: int64)
```



In [72]:

```
df.dtypes
```

Out[72]:

```
Age                           int64
Occupation                   object
Annual_Income               float64
Monthly_Inhand_Salary       float64
Num_Bank_Accounts             int64
Num_Credit_Card               int64
Interest_Rate               float64
Num_of_Loan                  object
Delay_from_due_date           int64
Num_of_Delayed_Payment       object
Changed_Credit_Limit        float64
Num_Credit_Inquiries        float64
Credit_Mix                   object
Outstanding_Debt             object
Credit_Utilization_Ratio    float64
Payment_of_Min_Amount        object
Total_EMI_per_month         float64
Amount_invested_monthly      object
Payment_Behaviour            object
Monthly_Balance             float64
Credit_Score                 object
Credit_History_Age_Months     int64
dtype: object
```

In [73]:

```
columns_to_convert = ['Num_of_Delayed_Payment', 'Outstanding_Debt', 'Amount_invested_mont
hly','Num_of_Loan']
for col in columns_to_convert:
    df[col] = df[col].str.replace('_', '').astype(float)
```

In [74]:

```
df.dtypes
```

Out[74]:

```
Age                           int64
Occupation                   object
Annual_Income               float64
Monthly_Inhand_Salary       float64
Num_Bank_Accounts             int64
Num_Credit_Card               int64
Interest_Rate               float64
Num_of_Loan                 float64
Delay_from_due_date           int64
Num_of_Delayed_Payment      float64
Changed_Credit_Limit        float64
Num_Credit_Inquiries        float64
Credit_Mix                   object
Outstanding_Debt            float64
Credit_Utilization_Ratio    float64
Payment_of_Min_Amount        object
Total_EMI_per_month         float64
Amount_invested_monthly     float64
Payment_Behaviour            object
Monthly_Balance             float64
Credit_Score                 object
Credit_History_Age_Months     int64
dtype: object
```

In [75]:

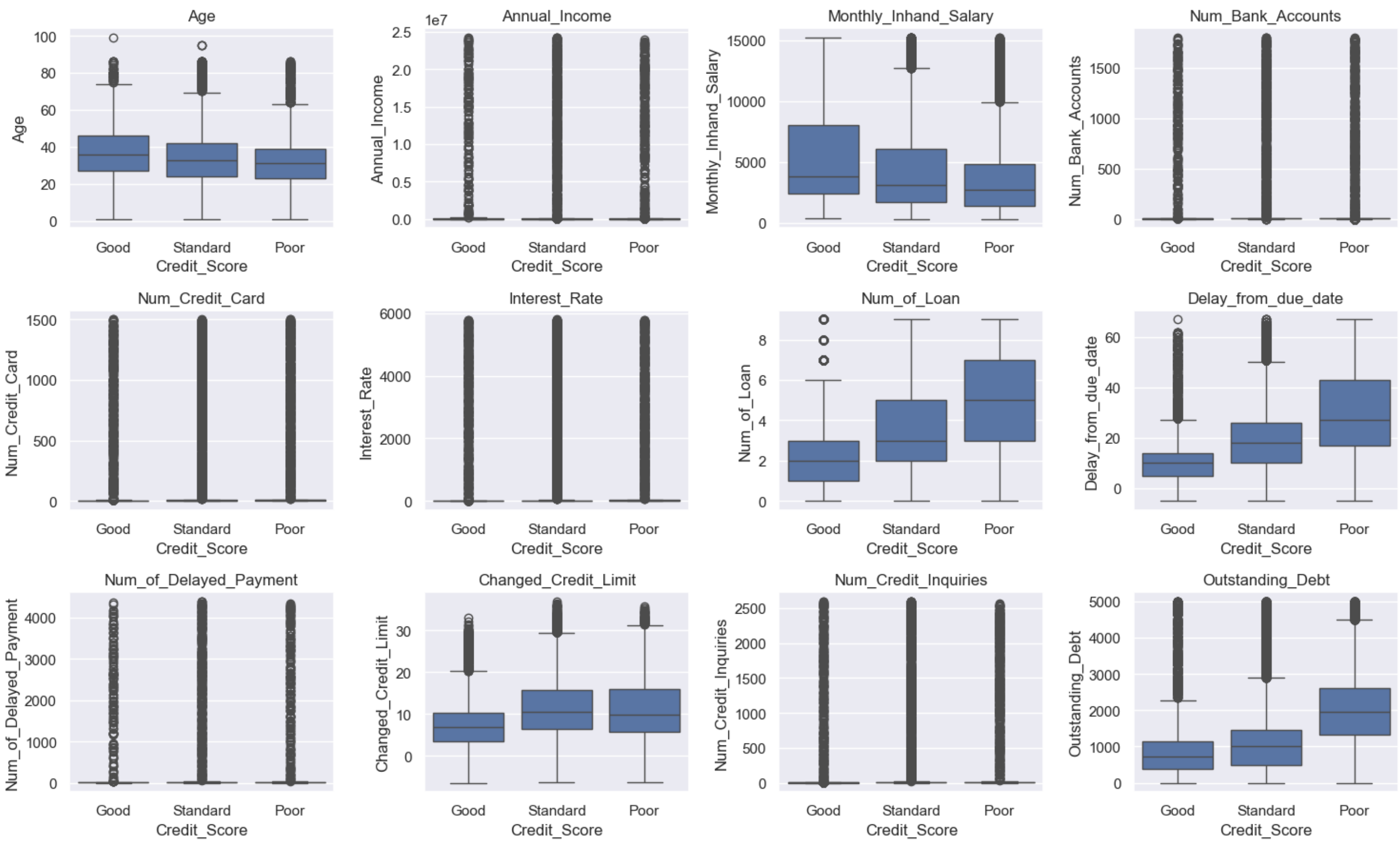
```
df.describe().T
```

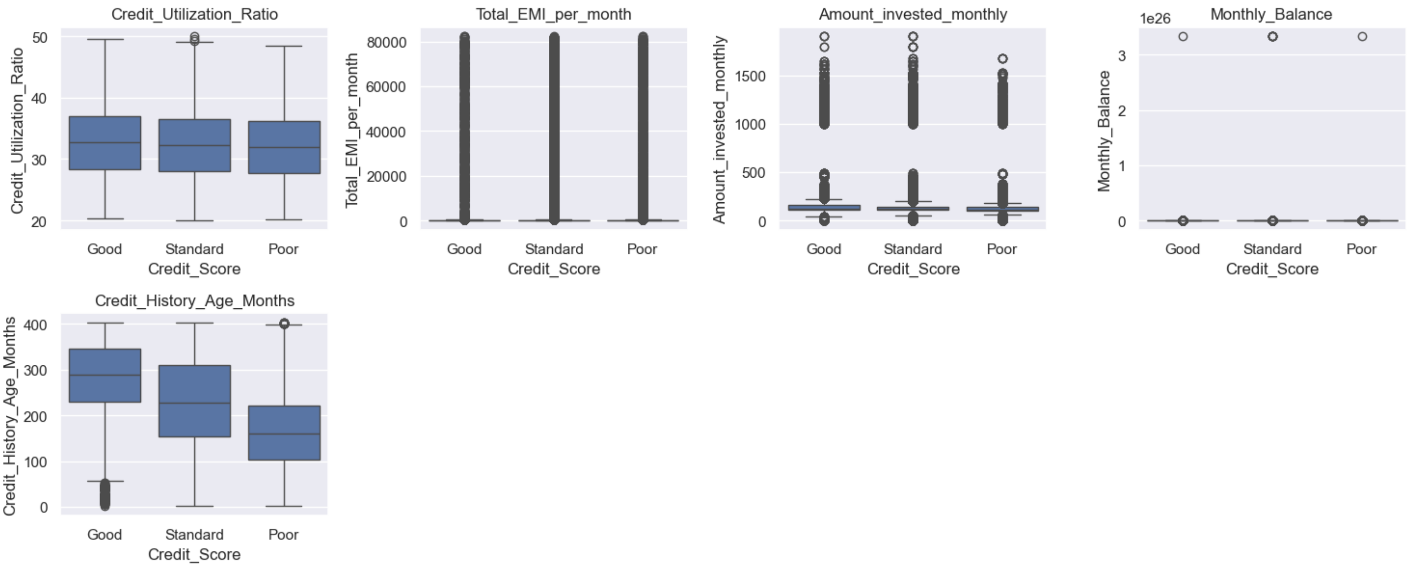
Out[75]:

	count	mean	std	min	25%
Age	100000.000	33.259	11.563	1.000	24.000
Annual_Income	100000.000	176415.701	1429618.051	7005.930	19457.500
Monthly_Inhand_Salary	100000.000	4198.351	3187.402	303.645	1626.762
Num_Bank_Accounts	100000.000	17.091	117.405	-1.000	3.000
Num_Credit_Card	100000.000	22.474	129.057	0.000	4.000
Interest_Rate	100000.000	72.466	466.423	1.000	8.000
Num_of_Loan	100000.000	3.513	2.403	0.000	2.000
Delay_from_due_date	100000.000	21.069	14.860	-5.000	10.000
Num_of_Delayed_Payment	100000.000	28.779	218.115	0.000	8.000
Changed_Credit_Limit	100000.000	10.389	6.718	-6.490	5.420
Num_Credit_Inquiries	100000.000	27.209	191.309	0.000	3.000
Outstanding_Debt	100000.000	1426.220	1155.129	0.230	566.072
Credit_Utilization_Ratio	100000.000	32.285	5.117	20.000	28.053
Total_EMI_per_month	100000.000	1403.118	8306.041	0.000	30.307
Amount_invested_monthly	100000.000	166.268	215.994	0.000	106.106
Monthly_Balance	100000.000	2999999999999999995805696.000	316215116526707522351923 2.000	0.008	268.945
Credit_History_Age_Months	100000.000	221.112	99.669	1.000	144.000

In [76]:

```
df_numeric_cols = [col for col in df.columns if df[col].dtype in ['int64', 'float64']]

plt.figure(figsize=(15,15))
for i, col in enumerate(df_numeric_cols):
    plt.subplot(5, 4, i+1)
    sns.boxplot(x='Credit_Score', y=col, data=df)
    plt.title(col)
plt.tight_layout()
plt.show()
```





Outliers

In [77]:

```python
# outlier deletion
df_num = df.select_dtypes(include='number')
for column in df_num.columns:
    for i in df["Credit_Score"].unique():
        selected_i = df[df["Credit_Score"] == i]
        selected_column = selected_i[column]

        std = selected_column.std()
        mean= selected_column.mean()

        max = mean + (4 * std)
        min =  mean - (4 * std)

        outliers = selected_column[((selected_i[column] > max) | (selected_i[column] < m
in))].index
        df.drop(index=outliers, inplace=True)
        print(column, i, outliers)
```

```
Age Good Index([6005, 10438, 23704, 28718, 31217, 34967, 61535, 84261], dtype='int64')
Age Standard Index([ 1654,  5055,  8549,  8788, 10431, 11190, 13372, 14671, 15641, 17916,
       18578, 21195, 21800, 23121, 24634, 25095, 29174, 33864, 34517, 35557,
       36575, 37904, 38248, 40478, 40483, 43149, 45669, 45931, 46755, 48897,
       50233, 52065, 53434, 55609, 56864, 57307, 57513, 59143, 60190, 60625,
       61040, 61509, 63018, 63815, 63983, 64001, 64179, 64436, 65223, 65420,
       66068, 66153, 67401, 67579, 68122, 68166, 68946, 71215, 71542, 71732,
       72205, 72375, 74040, 75531, 77053, 78564, 81562, 81593, 82335, 84621,
       85741, 86769, 87236, 87755, 89933, 94475, 94945, 95620, 96689, 99512,
       99776],
      dtype='int64')
Age Poor Index([   56,  2102,  2902,  4520,  4777,  6532,  6684,  8726,  9532,  9707,
       10247, 10858, 11527, 12940, 14747, 17467, 17547, 18362, 18585, 19783,
       21069, 21498, 21502, 22277, 22612, 24730, 25769, 26550, 30084, 31459,
       31985, 32317, 32368, 33182, 34109, 38665, 42060, 42061, 43206, 43406,
       44052, 45793, 45875, 46304, 47068, 50670, 50994, 51699, 52004, 52039,
       54308, 55516, 60482, 61317, 62633, 64130, 64377, 66272, 66470, 66523,
       66662, 70826, 70830, 71903, 72389, 74301, 75072, 75592, 77229, 77406,
       77826, 80914, 81038, 81576, 82739, 84625, 85297, 86131, 86225, 89408,
       89744, 91002, 92520, 93534, 95513, 99012, 99738],
      dtype='int64')
Annual_Income Good Index([   54,   564,   895,  2684,  3390,  4453,  5254,  5647,  6767,
7420,
       ...
       88289, 88708, 90303, 91686, 93073, 94278, 94336, 96155, 98445, 99264],
      dtype='int64', length=129)
Annual_Income Standard Index([  231,   361,   368,   602,   617,  1253,  1737,  2099,  23
03,  2815,
       ...
```

```
       98478, 98864, 99107, 99191, 99260, 99280, 99714, 99721, 99882, 99945],
      dtype='int64', length=441)
Annual_Income Poor Index([  862,  1546,  1706,  3100,  3277,  3577,  4446,  4950,  5208,
6162,
       ...
       92902, 93399, 93487, 93662, 93821, 94073, 94114, 95170, 96701, 99432],
      dtype='int64', length=208)
Monthly_Inhand_Salary Good Index([], dtype='int64')
Monthly_Inhand_Salary Standard Index([], dtype='int64')
Monthly_Inhand_Salary Poor Index([ 8688,  8690,  8691,  8692,  8693,  8694,  8695,  9376,
9377, 14880,
       ...
       96234, 96235, 96236, 96237, 96238, 98257, 98258, 98259, 98260, 98261],
      dtype='int64', length=130)
Num_Bank_Accounts Good Index([  356,   807,  1238,  1602,  4343,  4430,  4449,  8062,  81
40,  8381,
       ...
       94265, 95521, 95729, 96595, 96722, 97377, 98156, 99029, 99570, 99591],
      dtype='int64', length=169)
Num_Bank_Accounts Standard Index([  267,   288,   339,  1245,  1299,  1438,  1469,  1697,
1864,  2137,
       ...
       97962, 98217, 98230, 98291, 98300, 98505, 98749, 98796, 99343, 99722],
      dtype='int64', length=494)
Num_Bank_Accounts Poor Index([ 1057,  1122,  1323,  1513,  1549,  1802,  1913,  1934,  20
34,  2148,
       ...
       96522, 97427, 97610, 97796, 98451, 98541, 98735, 98954, 99417, 99638],
      dtype='int64', length=289)
Num_Credit_Card Good Index([   40,   925,  1671,  3015,  3349,  3490,  3998,  4180,  4545
,  4740,
       ...
       97816, 98091, 98678, 98679, 98819, 98898, 99233, 99289, 99592, 99605],
      dtype='int64', length=258)
Num_Credit_Card Standard Index([   10,   207,   324,   340,   343,   520,   657,   702,
720,   758,
       ...
       98472, 98686, 98874, 98966, 99061, 99470, 99493, 99600, 99619, 99769],
      dtype='int64', length=761)
Num_Credit_Card Poor Index([  157,   215,   948,  1355,  1702,  1858,  2027,  2052,  2122
,  2461,
       ...
       97881, 97975, 98199, 98723, 98891, 98895, 99132, 99147, 99434, 99520],
      dtype='int64', length=415)
Interest_Rate Good Index([   44,   178,   770,   893,  1131,  1224,  1453,  1640,  1663,
2826,
       ...
       96567, 97032, 97652, 97963, 98482, 98490, 98666, 98854, 99141, 99791],
      dtype='int64', length=260)
Interest_Rate Standard Index([  514,   766,   835,   848,  1020,  1083,  1088,  1184,  12
15,  1277,
       ...
       98729, 98770, 99090, 99093, 99119, 99448, 99542, 99551, 99612, 99621],
      dtype='int64', length=681)
Interest_Rate Poor Index([  167,   472,   482,   559,   886,   901,   947,  1521,  1887,
2048,
       ...
       97053, 97350, 97356, 97391, 97608, 97675, 98508, 98893, 99010, 99997],
      dtype='int64', length=375)
Num_of_Loan Good Index([   21,  2295,  3078,  4619,  4621,  7885, 31259, 31261, 31263, 32
515,
       32516, 32519, 35251, 35253, 35254, 35255, 37035, 37036, 37037, 37038,
       37039, 37821, 38655, 39988, 45038, 48004, 50795, 50797, 50798, 50799,
       52996, 52998, 52999, 53059, 53060, 53061, 53062, 53063, 61515, 61517,
       61787, 61788, 61791, 64166, 65683, 65685, 65686, 65687, 72979, 72980,
       72981, 79643, 79645, 79646, 79647, 80022, 80023, 81870, 81871, 91093,
       91590, 91591, 93028, 93029, 93030, 93940, 93941, 93943],
      dtype='int64')
Num_of_Loan Standard Index([], dtype='int64')
Num_of_Loan Poor Index([], dtype='int64')
Delay_from_due_date Good Index([  686,   687,  1291,  1292,  1293,  1294,  1295,  6854,
6855, 14100,
```

```
       ...
       88199, 89419, 89420, 89422, 89423, 95286, 95287, 98619, 98620, 98621],
      dtype='int64', length=109)
Delay_from_due_date Standard Index([], dtype='int64')
Delay_from_due_date Poor Index([], dtype='int64')
Num_of_Delayed_Payment Good Index([  284,  1032,  2296,  3341,  4375,  6267,  6380,  7186
,  8383, 12300,
       ...
       89726, 90051, 90443, 90856, 91701, 94055, 95587, 96043, 97062, 98351],
      dtype='int64', length=102)
Num_of_Delayed_Payment Standard Index([  706,  1212,  1611,  1916,  2480,  3164,  3312,
3793,  4423,  4462,
       ...
       97645, 97775, 97984, 98007, 98212, 98362, 98549, 98733, 99562, 99825],
      dtype='int64', length=311)
Num_of_Delayed_Payment Poor Index([  252,   304,   409,  1616,  2846,  3556,  4665,  4861
,  5664,  6983,
       ...
       95099, 95100, 95776, 96699, 96711, 97488, 97973, 99069, 99133, 99402],
      dtype='int64', length=145)
Changed_Credit_Limit Good Index([ 7422, 11987, 11988, 11989, 13958, 19983, 22811, 22812,
22813, 22814,
       29588, 32107, 32109, 32111, 33111, 39859, 39860, 40915, 40916, 40917,
       44011, 44012, 44013, 44014, 46964, 46965, 46967, 50075, 50076, 50077,
       50078, 52206, 54271, 56966, 59036, 59037, 59038, 59039, 60195, 60196,
       60197, 60198, 60199, 74381, 83270, 83271, 90574, 90575, 90892, 90893,
       90895, 95588, 97259, 97260, 97261, 97263],
      dtype='int64')
Changed_Credit_Limit Standard Index([], dtype='int64')
Changed_Credit_Limit Poor Index([], dtype='int64')
Num_Credit_Inquiries Good Index([  503,  1810,  2704,  2990,  4009,  4718,  4880,  4912,
6751,  6837,
       ...
       92792, 93442, 93967, 94803, 95013, 96575, 96668, 96992, 98424, 99513],
      dtype='int64', length=181)
Num_Credit_Inquiries Standard Index([  193,   198,   312,  1246,  1369,  1390,  2135,  24
38,  2744,  3028,
       ...
       98314, 98558, 98641, 98873, 99043, 99076, 99152, 99502, 99717, 99800],
      dtype='int64', length=597)
Num_Credit_Inquiries Poor Index([  173,  1309,  1331,  1474,  1529,  1566,  1591,  1619,
1755,  2177,
       ...
       96378, 96507, 96750, 97191, 97389, 97418, 98050, 98060, 98282, 99163],
      dtype='int64', length=331)
Outstanding_Debt Good Index([ 1790,  1791,  4995,  4997,  4998,  4999,  7419,  7421,  742
3,  8739,
        8740,  8741,  8742, 13692, 13693, 16646, 16647, 19982, 21462, 21463,
       22815, 27827, 27830, 27831, 29589, 33110, 33870, 33871, 33884, 33886,
       33887, 34798, 34799, 39203, 39204, 45396, 46963, 48355, 48356, 48603,
       48604, 48605, 48822, 48823, 49638, 51883, 51886, 52596, 54615, 56419,
       56420, 56421, 56422, 56423, 58414, 58415, 61158, 61159, 62251, 63595,
       63596, 66669, 66670, 74379, 74382, 74383, 78502, 79724, 79727, 80171,
       80173, 80174, 80175, 89774, 90099, 90101, 90102, 90103, 91251, 91253,
       95518, 95519, 95589, 97262],
      dtype='int64')
Outstanding_Debt Standard Index([], dtype='int64')
Outstanding_Debt Poor Index([], dtype='int64')
Credit_Utilization_Ratio Good Index([], dtype='int64')
Credit_Utilization_Ratio Standard Index([], dtype='int64')
Credit_Utilization_Ratio Poor Index([], dtype='int64')
Total_EMI_per_month Good Index([   94,   359,   580,   723,   791,  1019,  1136,  1171,
1336,  2528,
       ...
       96671, 97382, 97785, 98202, 99266, 99580, 99872, 99903, 99960, 99970],
      dtype='int64', length=303)
Total_EMI_per_month Standard Index([  307,   440,   519,   548,   572,   589,   598,   68
4,  1082,  1275,
       ...
       98462, 98498, 99038, 99050, 99120, 99318, 99367, 99543, 99908, 99993],
      dtype='int64', length=934)
Total_EMI_per_month Poor Index([  158,   537,  1056,  1123,  1392,  1393,  1505,  1555,
```

```
1719,  2051,
        ...
        96745, 96950, 97348, 97354, 97357, 97394, 97864, 98509, 99485, 99737],
      dtype='int64', length=476)
Amount_invested_monthly Good Index([ 3088,  3089,  7691,  7692,  7694,  7695, 11696, 1169
7, 11698, 11699,
        11700, 11701, 13275, 13276, 13277, 21032, 21034, 21035, 21037, 21039,
        28681, 28682, 28686, 28687, 35163, 35164, 35165, 45824, 45825, 45827,
        45828, 45829, 45830, 62307, 62308, 62309, 62480, 62483, 62484, 62485,
        73675, 73676, 73677, 80240, 80241, 80242, 80243, 80245, 80246, 80247],
      dtype='int64')
Amount_invested_monthly Standard Index([  650,   776,   778,   779,   780,   781,   782,
783,   920,   923,
        ...
        99486, 99487, 99516, 99519, 99920, 99921, 99922, 99923, 99924, 99925],
      dtype='int64', length=2045)
Amount_invested_monthly Poor Index([ 1579,  1581,  1582,  1824,  1825,  1826,  1828,  182
9,  2040,  2041,
        ...
        99482, 99483, 99484, 99632, 99633, 99634, 99635, 99636, 99637, 99639],
      dtype='int64', length=464)
Monthly_Balance Good Index([26177], dtype='int64')
Monthly_Balance Standard Index([5545, 29158, 35570, 38622, 60009, 75251, 82918], dtype='i
nt64')
Monthly_Balance Poor Index([83255], dtype='int64')
Credit_History_Age_Months Good Index([], dtype='int64')
Credit_History_Age_Months Standard Index([], dtype='int64')
Credit_History_Age_Months Poor Index([], dtype='int64')
```

In [78]:

```
df.shape
```

Out[78]:

```
(88949, 22)
```

In [79]:

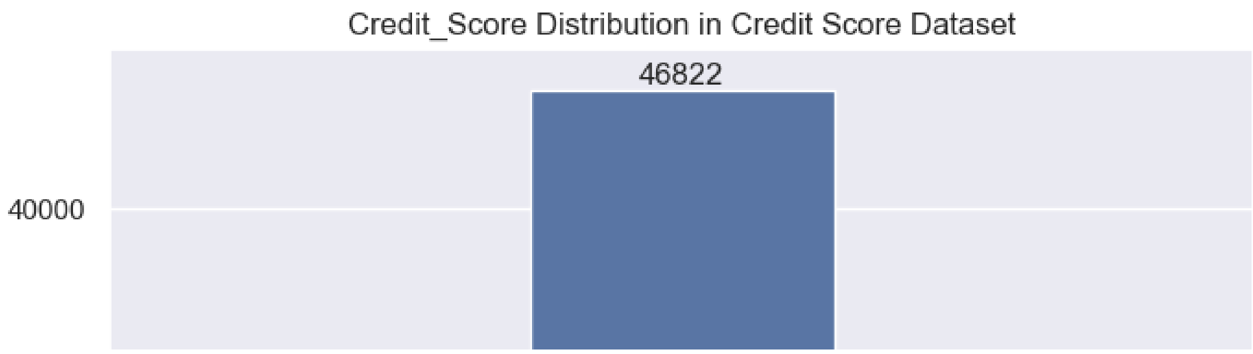
```
df.Credit_Score.value_counts()
```

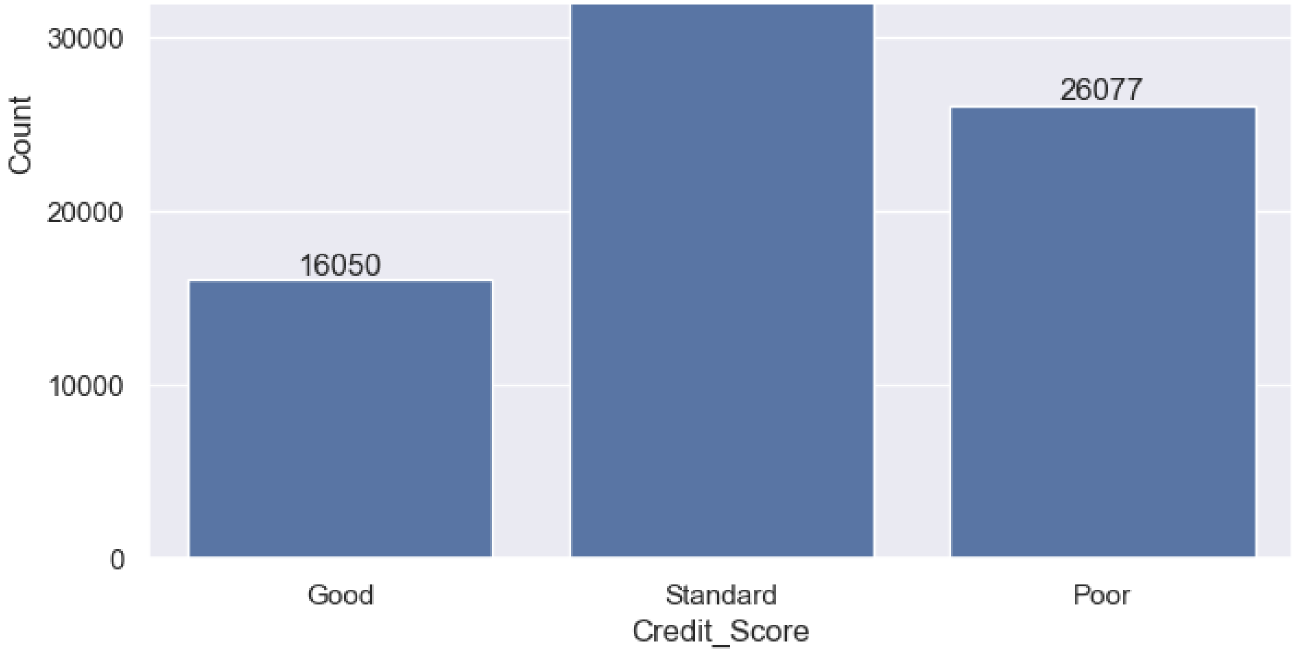
Out[79]:

```
Credit_Score
Standard    46822
Poor        26077
Good        16050
Name: count, dtype: int64
```

In [80]:

```
# Distribution of target variable

plt.figure(figsize=(8, 6))
ax=sns.countplot(data=df, x='Credit_Score')
ax.bar_label(ax.containers[0])
plt.title('Credit_Score Distribution in Credit Score Dataset')
plt.xlabel('Credit_Score')
plt.ylabel('Count')
plt.show()
```

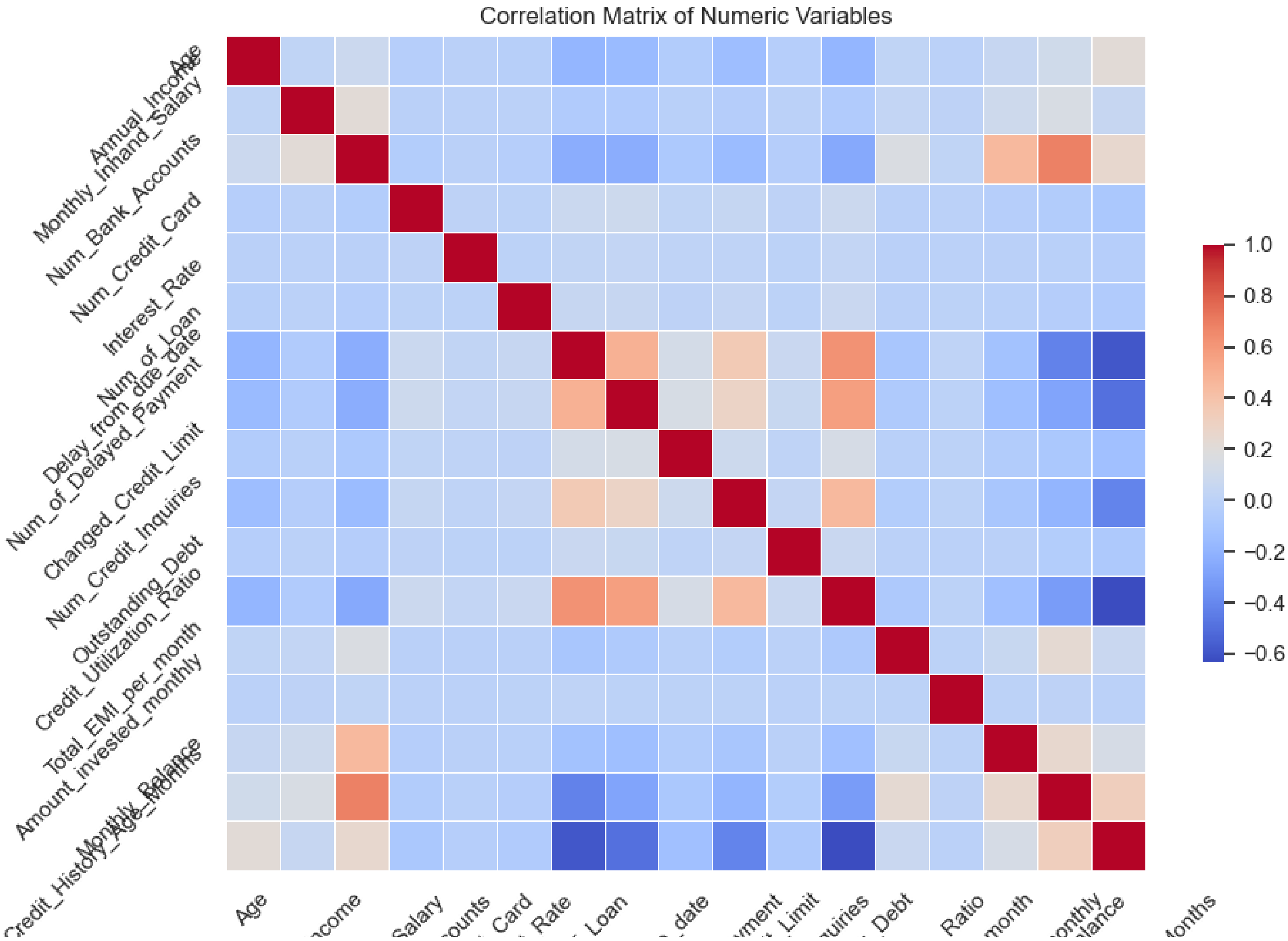


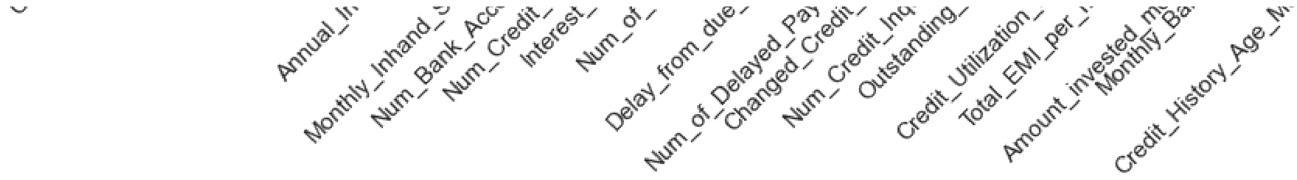


- **Credit_Score outputun degerlerinin dagilısı dengesiz.**

In [81]:

```
numeric_df = df.select_dtypes(include=['number'])

plt.figure(figsize=(10, 8))
sns.heatmap(numeric_df.corr(), annot=False, cmap="coolwarm", fmt=".2f", linewidths=.5, c
bar_kws={"shrink": .5})
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.title('Correlation Matrix of Numeric Variables')
plt.tight_layout()
plt.show()
```





Save and read clean data

In [82]:

```
df.to_csv("CreditScoreClassification_train_cleaned_outlier1.csv", index=False)
```

In [83]:

```
df = pd.read_csv('CreditScoreClassification_train_cleaned_outlier1.csv')
df.head()
```

Out[83]:

	Age	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest_Rate	Num_of_L
0	23	Scientist	19114.120	1824.843	3	4	3.000	4.
1	23	Scientist	19114.120	1824.843	3	4	3.000	4.
2	5	Scientist	19114.120	1824.843	3	4	3.000	4.
3	23	Scientist	19114.120	1824.843	3	4	3.000	4.
4	23	Scientist	19114.120	1824.843	3	4	3.000	4.

5 rows × 22 columns

In [84]:

```
df.isnull().sum()
```

Out[84]:

```
Age                         0
Occupation                  0
Annual_Income               0
Monthly_Inhand_Salary       0
Num_Bank_Accounts           0
Num_Credit_Card             0
Interest_Rate               0
Num_of_Loan                 0
Delay_from_due_date         0
Num_of_Delayed_Payment      0
Changed_Credit_Limit        0
Num_Credit_Inquiries        0
Credit_Mix                  0
Outstanding_Debt            0
Credit_Utilization_Ratio    0
Payment_of_Min_Amount       0
Total_EMI_per_month         0
Amount_invested_monthly     0
Payment_Behaviour           0
Monthly_Balance             0
Credit_Score                0
Credit_History_Age_Months   0
dtype: int64
```

In [85]:

```
df.shape
```

Out[85]:

```
(88949, 22)
```

LabelEncoding for output column

In [86]:

```
from sklearn.preprocessing import LabelEncoder,OrdinalEncoder
df["Credit_Score"] = LabelEncoder().fit_transform(df["Credit_Score"])
df["Credit_Score"]
```

Out[86]:

```
0        0
1        0
2        0
3        0
4        0
        ..
88944    1
88945    1
88946    1
88947    2
88948    1
Name: Credit_Score, Length: 88949, dtype: int32
```

In [87]:

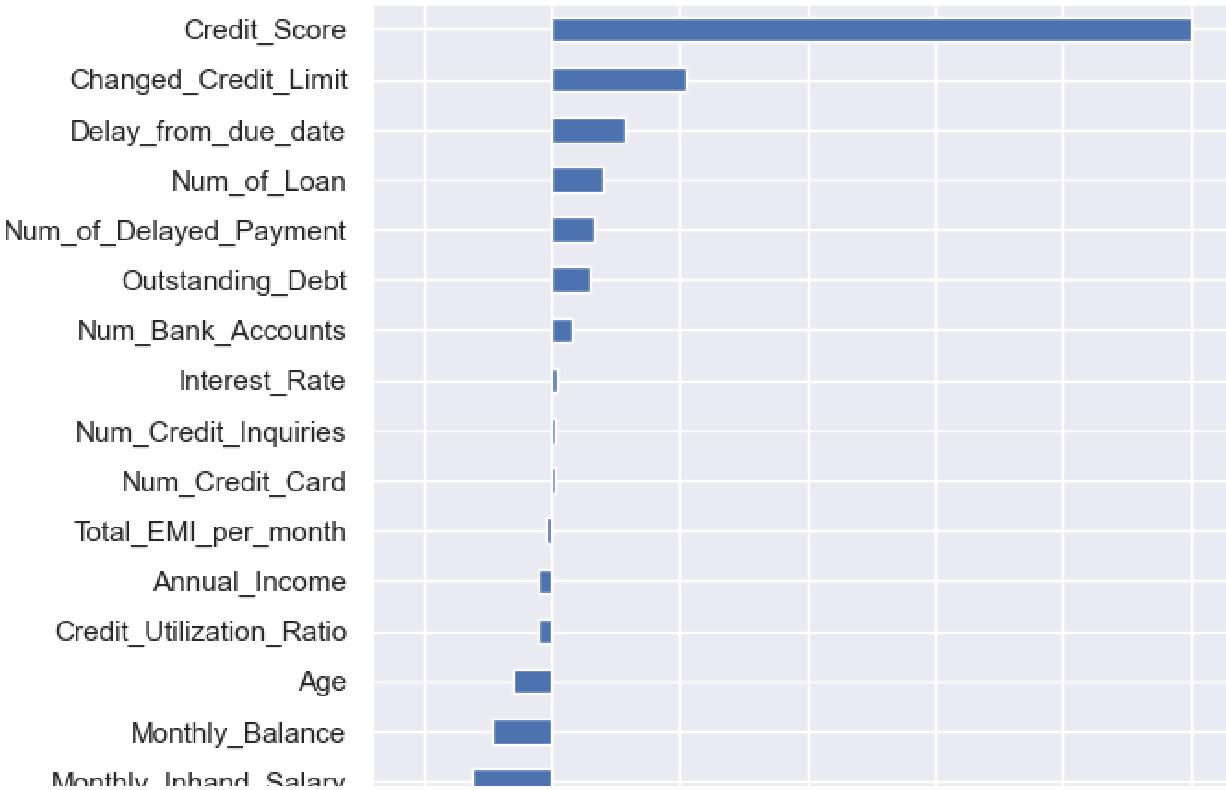
```
df["Credit_Score"].value_counts()
```

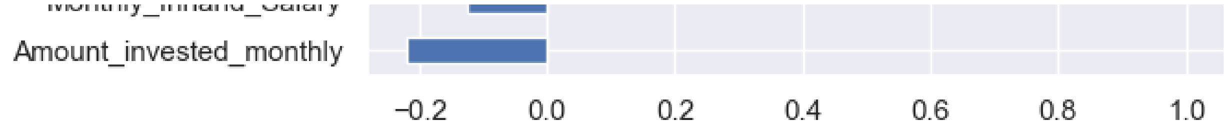
Out[87]:

```
Credit_Score
2    46822
1    26077
0    16050
Name: count, dtype: int64
```

In [88]:

```
# Correlation of target variable with features after numerical transformation


numerical_df = df.select_dtypes(include=[np.number])
correlation_series = numerical_df.corr()['Credit_Score'][:-1].sort_values()
correlation_series.plot.barh();
```





Encoding for categorical columns¶

In [89]:

```
# select columns of type 'object'
df.select_dtypes(include=['object']).columns
```

Out[89]:

```
Index(['Occupation', 'Credit_Mix', 'Payment_of_Min_Amount',
       'Payment_Behaviour'],
      dtype='object')
```

In [90]:

```
payment_behaviour_categories = ['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']

payment_behaviour_encoder = OrdinalEncoder(categories=[payment_behaviour_categories])

df['Payment_Behaviour'] = payment_behaviour_encoder.fit_transform(df[['Payment_Behaviour'
]])
```

In [91]:

```
#credit_mix_categories = ['Bad', 'Standard', 'Good']
#credit_mix_encoder = OrdinalEncoder(categories=[credit_mix_categories])


label_encoder1 = OrdinalEncoder()
df['Credit_Mix'] = label_encoder1.fit_transform(df[['Credit_Mix']])
```

In [92]:

```
label_encoder2 = LabelEncoder()
df['Payment_of_Min_Amount'] = label_encoder2.fit_transform(df['Payment_of_Min_Amount'])
```

In [93]:

```
label_encoder3 = LabelEncoder()
df['Occupation'] = label_encoder3.fit_transform(df['Occupation'])
```

In [94]:

```
df.shape
```

Out[94]:

```
(88949, 22)
```

In [95]:

```
df.head().T
```

Out[95]:

	0	1	2	3	4
Age	23.000	23.000	5.000	23.000	23.000
Occupation	12.000	12.000	12.000	12.000	12.000

	0	1	2	3	4
Annual_Income	19114.120	19114.120	19114.120	19114.120	19114.120
Monthly_Inhand_Salary	1824.843	1824.843	1824.843	1824.843	1824.843
Num_Bank_Accounts	3.000	3.000	3.000	3.000	3.000
Num_Credit_Card	4.000	4.000	4.000	4.000	4.000
Interest_Rate	3.000	3.000	3.000	3.000	3.000
Num_of_Loan	4.000	4.000	4.000	4.000	4.000
Delay_from_due_date	3.000	-1.000	3.000	5.000	6.000
Num_of_Delayed_Payment	7.000	0.000	7.000	4.000	0.000
Changed_Credit_Limit	11.270	11.270	10.389	6.270	11.270
Num_Credit_Inquiries	4.000	4.000	4.000	4.000	4.000
Credit_Mix	1.000	1.000	1.000	1.000	1.000
Outstanding_Debt	809.980	809.980	809.980	809.980	809.980
Credit_Utilization_Ratio	26.823	31.945	28.609	31.378	24.797
Payment_of_Min_Amount	1.000	1.000	1.000	1.000	1.000
Total_EMI_per_month	49.575	49.575	49.575	49.575	49.575
Amount_invested_monthly	118.280	118.280	118.280	118.280	118.280
Payment_Behaviour	3.000	2.000	1.000	0.000	4.000
Monthly_Balance	312.494	284.629	331.210	223.451	341.489
Credit_Score	0.000	0.000	0.000	0.000	0.000
Credit_History_Age_Months	265.000	265.000	267.000	268.000	269.000

In [96]:

```
# Separate properties and target variable

X = df.drop("Credit_Score", axis=1)
y = df.Credit_Score
```

In [97]:

```
y.value_counts(normalize=True)
```

Out[97]:

```
Credit_Score
2    0.526
1    0.293
0    0.180
Name: proportion, dtype: float64
```

SMOTE

In [98]:

```
from imblearn.over_sampling import SMOTE

 # Synthetic Minority Oversampling Technique

smote = SMOTE()
X, y = smote.fit_resample(X,y)
```

In [99]:

```
y.value_counts()
```

Out[99]:

```
Credit_Score
0    46823
```

```
0    46822
2    46822
1    46822
Name: count, dtype: int64
```

Train-Test Split

In [100]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15,
                                                    stratify=y, shuffle=True, random_st
ate=42)
```

In [101]:

```
print("Training set shape:", X_train.shape, y_train.shape)
print("Testing set shape:", X_test.shape, y_test.shape)
```

```
Training set shape: (119396, 21) (119396,)
Testing set shape: (21070, 21) (21070,)
```

In [102]:

```
df["Credit_Score"].value_counts()
```

Out[102]:

```
Credit_Score
2    46822
1    26077
0    16050
Name: count, dtype: int64
```

y_train.value_counts()

Normalization

In [103]:

```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [104]:

```
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))
```

ANN Model

In [105]:

```
model = 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(128, activation='relu'), # ekledim
    BatchNormalization(),
    Dropout(0.3),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.25),

    Dense(3, activation='softmax')
    ])
model.compile(optimizer = Adam(learning_rate=0.001),
              loss='sparse_categorical_crossentropy',
              metrics = ['accuracy'])




early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=60,
                               restore_best_weights=True)
model.fit(x=X_train,
          y=y_train,
          validation_data=(X_test, y_test),
          validation_split=0.1,
          batch_size=512,
          epochs=900,
          verbose=1,
          callbacks=[early_stopping])
```

```
Epoch 1/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - accuracy: 0.6019 - loss: 0.9961 - val_accurac
y: 0.5444 - val_loss: 1.0055
Epoch 2/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7177 - loss: 0.7111 - val_accuracy
: 0.7020 - val_loss: 0.7169
Epoch 3/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7281 - loss: 0.6905 - val_accuracy
: 0.7382 - val_loss: 0.6550
Epoch 4/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7302 - loss: 0.6779 - val_accuracy
: 0.7385 - val_loss: 0.6535
Epoch 5/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7331 - loss: 0.6728 - val_accuracy
: 0.7387 - val_loss: 0.6486
Epoch 6/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7341 - loss: 0.6669 - val_accuracy
: 0.7352 - val_loss: 0.6536
Epoch 7/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7350 - loss: 0.6637 - val_accuracy
: 0.7398 - val_loss: 0.6460
Epoch 8/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7357 - loss: 0.6629 - val_accuracy
: 0.7309 - val_loss: 0.6659
Epoch 9/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7345 - loss: 0.6630 - val_accuracy
: 0.7377 - val_loss: 0.6544
Epoch 10/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7393 - loss: 0.6547 - val_accuracy
: 0.7411 - val_loss: 0.6419
Epoch 11/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7401 - loss: 0.6531 - val_accuracy
: 0.7443 - val_loss: 0.6437
Epoch 12/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7386 - loss: 0.6542 - val_accuracy
```

```
: 0.7448 - val_loss: 0.6396
Epoch 13/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7411 - loss: 0.6493 - val_accuracy
: 0.7404 - val_loss: 0.6462
Epoch 14/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7416 - loss: 0.6493 - val_accuracy
: 0.7426 - val_loss: 0.6394
Epoch 15/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7435 - loss: 0.6441 - val_accuracy
: 0.7470 - val_loss: 0.6288
Epoch 16/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7443 - loss: 0.6406 - val_accuracy
: 0.7448 - val_loss: 0.6311
Epoch 17/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7452 - loss: 0.6360 - val_accuracy
: 0.7448 - val_loss: 0.6370
Epoch 18/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7464 - loss: 0.6323 - val_accuracy
: 0.7448 - val_loss: 0.6286
Epoch 19/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7473 - loss: 0.6332 - val_accuracy
: 0.7440 - val_loss: 0.6320
Epoch 20/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7470 - loss: 0.6304 - val_accuracy
: 0.7496 - val_loss: 0.6169
Epoch 21/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7475 - loss: 0.6259 - val_accuracy
: 0.7365 - val_loss: 0.6381
Epoch 22/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7491 - loss: 0.6234 - val_accuracy
: 0.7490 - val_loss: 0.6153
Epoch 23/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7499 - loss: 0.6205 - val_accuracy
: 0.7475 - val_loss: 0.6379
Epoch 24/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7483 - loss: 0.6235 - val_accuracy
: 0.7252 - val_loss: 0.6399
Epoch 25/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7485 - loss: 0.6226 - val_accuracy
: 0.7504 - val_loss: 0.6144
Epoch 26/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7514 - loss: 0.6147 - val_accuracy
: 0.7523 - val_loss: 0.6057
Epoch 27/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7544 - loss: 0.6136 - val_accuracy
: 0.7518 - val_loss: 0.6180
Epoch 28/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7522 - loss: 0.6138 - val_accuracy
: 0.7496 - val_loss: 0.6155
Epoch 29/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7519 - loss: 0.6145 - val_accuracy
: 0.7562 - val_loss: 0.6072
Epoch 30/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7543 - loss: 0.6097 - val_accuracy
: 0.6728 - val_loss: 0.7842
Epoch 31/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7530 - loss: 0.6124 - val_accuracy
: 0.7483 - val_loss: 0.6057
Epoch 32/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7533 - loss: 0.6078 - val_accuracy
: 0.7439 - val_loss: 0.6192
Epoch 33/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7514 - loss: 0.6109 - val_accuracy
: 0.7557 - val_loss: 0.6060
Epoch 34/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7544 - loss: 0.6060 - val_accuracy
: 0.7577 - val_loss: 0.5941
Epoch 35/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7562 - loss: 0.6030 - val_accuracy
: 0.7520 - val_loss: 0.5973
Epoch 36/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7566 - loss: 0.6035 - val_accuracy
```

```
: 0.7542 - val_loss: 0.6021
Epoch 37/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7554 - loss: 0.6022 - val_accuracy
: 0.7636 - val_loss: 0.5868
Epoch 38/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7601 - loss: 0.5972 - val_accuracy
: 0.7538 - val_loss: 0.6002
Epoch 39/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7572 - loss: 0.5974 - val_accuracy
: 0.7496 - val_loss: 0.6251
Epoch 40/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7589 - loss: 0.5958 - val_accuracy
: 0.7572 - val_loss: 0.5900
Epoch 41/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7594 - loss: 0.5955 - val_accuracy
: 0.7509 - val_loss: 0.6033
Epoch 42/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7589 - loss: 0.5952 - val_accuracy
: 0.7525 - val_loss: 0.6048
Epoch 43/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7601 - loss: 0.5944 - val_accuracy
: 0.7637 - val_loss: 0.5769
Epoch 44/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7609 - loss: 0.5912 - val_accuracy
: 0.7664 - val_loss: 0.5771
Epoch 45/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7615 - loss: 0.5897 - val_accuracy
: 0.7661 - val_loss: 0.5739
Epoch 46/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7608 - loss: 0.5904 - val_accuracy
: 0.7665 - val_loss: 0.5714
Epoch 47/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7612 - loss: 0.5901 - val_accuracy
: 0.7096 - val_loss: 0.7501
Epoch 48/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7583 - loss: 0.5925 - val_accuracy
: 0.7524 - val_loss: 0.5979
Epoch 49/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7601 - loss: 0.5913 - val_accuracy
: 0.7624 - val_loss: 0.5811
Epoch 50/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7606 - loss: 0.5907 - val_accuracy
: 0.6337 - val_loss: 1.0171
Epoch 51/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7584 - loss: 0.5938 - val_accuracy
: 0.7584 - val_loss: 0.5920
Epoch 52/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7622 - loss: 0.5880 - val_accuracy
: 0.7645 - val_loss: 0.5713
Epoch 53/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7621 - loss: 0.5862 - val_accuracy
: 0.7672 - val_loss: 0.5691
Epoch 54/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7633 - loss: 0.5824 - val_accuracy
: 0.7487 - val_loss: 0.6023
Epoch 55/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7638 - loss: 0.5841 - val_accuracy
: 0.7372 - val_loss: 0.6124
Epoch 56/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7623 - loss: 0.5841 - val_accuracy
: 0.7613 - val_loss: 0.5798
Epoch 57/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7640 - loss: 0.5810 - val_accuracy
: 0.7688 - val_loss: 0.5625
Epoch 58/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7654 - loss: 0.5815 - val_accuracy
: 0.7464 - val_loss: 0.6357
Epoch 59/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7640 - loss: 0.5808 - val_accuracy
: 0.7663 - val_loss: 0.5755
Epoch 60/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7631 - loss: 0.5820 - val_accuracy
```

```
: 0.6518 - val_loss: 0.8310
Epoch 61/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7617 - loss: 0.5862 - val_accuracy
: 0.7382 - val_loss: 0.7820
Epoch 62/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7616 - loss: 0.5862 - val_accuracy
: 0.6973 - val_loss: 0.9375
Epoch 63/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7619 - loss: 0.5892 - val_accuracy
: 0.7663 - val_loss: 0.5731
Epoch 64/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7617 - loss: 0.5844 - val_accuracy
: 0.7427 - val_loss: 0.6716
Epoch 65/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7644 - loss: 0.5823 - val_accuracy
: 0.7557 - val_loss: 0.6150
Epoch 66/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7656 - loss: 0.5828 - val_accuracy
: 0.7720 - val_loss: 0.5589
Epoch 67/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7656 - loss: 0.5764 - val_accuracy
: 0.7684 - val_loss: 0.5639
Epoch 68/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7645 - loss: 0.5783 - val_accuracy
: 0.7629 - val_loss: 0.5712
Epoch 69/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7657 - loss: 0.5773 - val_accuracy
: 0.7251 - val_loss: 0.7141
Epoch 70/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7643 - loss: 0.5828 - val_accuracy
: 0.7552 - val_loss: 0.5939
Epoch 71/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7665 - loss: 0.5771 - val_accuracy
: 0.7697 - val_loss: 0.5644
Epoch 72/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7634 - loss: 0.5797 - val_accuracy
: 0.7711 - val_loss: 0.5544
Epoch 73/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7663 - loss: 0.5764 - val_accuracy
: 0.7416 - val_loss: 0.6455
Epoch 74/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7632 - loss: 0.5804 - val_accuracy
: 0.7610 - val_loss: 0.6674
Epoch 75/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7660 - loss: 0.5799 - val_accuracy
: 0.6763 - val_loss: 0.7261
Epoch 76/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7656 - loss: 0.5793 - val_accuracy
: 0.7564 - val_loss: 0.5827
Epoch 77/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7626 - loss: 0.5818 - val_accuracy
: 0.7657 - val_loss: 0.5663
Epoch 78/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7641 - loss: 0.5784 - val_accuracy
: 0.7653 - val_loss: 0.5731
Epoch 79/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7641 - loss: 0.5733 - val_accuracy
: 0.7749 - val_loss: 0.5517
Epoch 80/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7639 - loss: 0.5778 - val_accuracy
: 0.7708 - val_loss: 0.5551
Epoch 81/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7634 - loss: 0.5798 - val_accuracy
: 0.7703 - val_loss: 0.5562
Epoch 82/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7660 - loss: 0.5743 - val_accuracy
: 0.7676 - val_loss: 0.5901
Epoch 83/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7669 - loss: 0.5725 - val_accuracy
: 0.7723 - val_loss: 0.5537
Epoch 84/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7679 - loss: 0.5701 - val_accuracy
```

```
: 0.7654 - val_loss: 0.5956
Epoch 85/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7685 - loss: 0.5726 - val_accuracy
: 0.4610 - val_loss: 2.4096
Epoch 86/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7666 - loss: 0.5776 - val_accuracy
: 0.7623 - val_loss: 0.6123
Epoch 87/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7673 - loss: 0.5751 - val_accuracy
: 0.7730 - val_loss: 0.5507
Epoch 88/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7684 - loss: 0.5724 - val_accuracy
: 0.7469 - val_loss: 0.6049
Epoch 89/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7673 - loss: 0.5734 - val_accuracy
: 0.7648 - val_loss: 0.5626
Epoch 90/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7663 - loss: 0.5753 - val_accuracy
: 0.7693 - val_loss: 0.5580
Epoch 91/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7671 - loss: 0.5713 - val_accuracy
: 0.7709 - val_loss: 0.5626
Epoch 92/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7677 - loss: 0.5710 - val_accuracy
: 0.7761 - val_loss: 0.5480
Epoch 93/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7684 - loss: 0.5678 - val_accuracy
: 0.7760 - val_loss: 0.5480
Epoch 94/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7687 - loss: 0.5666 - val_accuracy
: 0.7727 - val_loss: 0.5525
Epoch 95/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7689 - loss: 0.5675 - val_accuracy
: 0.7774 - val_loss: 0.5447
Epoch 96/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7695 - loss: 0.5657 - val_accuracy
: 0.7796 - val_loss: 0.5412
Epoch 97/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7699 - loss: 0.5662 - val_accuracy
: 0.6743 - val_loss: 0.7420
Epoch 98/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7681 - loss: 0.5686 - val_accuracy
: 0.7704 - val_loss: 0.5594
Epoch 99/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7701 - loss: 0.5658 - val_accuracy
: 0.4132 - val_loss: 2.9529
Epoch 100/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7662 - loss: 0.5754 - val_accuracy
: 0.7667 - val_loss: 0.5758
Epoch 101/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7674 - loss: 0.5687 - val_accuracy
: 0.7773 - val_loss: 0.5434
Epoch 102/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7698 - loss: 0.5686 - val_accuracy
: 0.7773 - val_loss: 0.5475
Epoch 103/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7693 - loss: 0.5653 - val_accuracy
: 0.7702 - val_loss: 0.5529
Epoch 104/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7698 - loss: 0.5686 - val_accuracy
: 0.7751 - val_loss: 0.5482
Epoch 105/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7686 - loss: 0.5666 - val_accuracy
: 0.7706 - val_loss: 0.5579
Epoch 106/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7719 - loss: 0.5634 - val_accuracy
: 0.7725 - val_loss: 0.5558
Epoch 107/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7712 - loss: 0.5644 - val_accuracy
: 0.7745 - val_loss: 0.5558
Epoch 108/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7716 - loss: 0.5619 - val_accuracy
```

```
: 0.7751 - val_loss: 0.5418
Epoch 109/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7733 - loss: 0.5612 - val_accuracy
: 0.7749 - val_loss: 0.5484
Epoch 110/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7720 - loss: 0.5624 - val_accuracy
: 0.7773 - val_loss: 0.5443
Epoch 111/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7706 - loss: 0.5670 - val_accuracy
: 0.7776 - val_loss: 0.5476
Epoch 112/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7716 - loss: 0.5646 - val_accuracy
: 0.7773 - val_loss: 0.5367
Epoch 113/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7726 - loss: 0.5583 - val_accurac
y: 0.7774 - val_loss: 0.5402
Epoch 114/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7716 - loss: 0.5604 - val_accuracy
: 0.7817 - val_loss: 0.5408
Epoch 115/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7703 - loss: 0.5608 - val_accuracy
: 0.7745 - val_loss: 0.5453
Epoch 116/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7714 - loss: 0.5609 - val_accuracy
: 0.7842 - val_loss: 0.5351
Epoch 117/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7728 - loss: 0.5594 - val_accuracy
: 0.7768 - val_loss: 0.5453
Epoch 118/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7709 - loss: 0.5584 - val_accuracy
: 0.7309 - val_loss: 0.7536
Epoch 119/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7705 - loss: 0.5656 - val_accuracy
: 0.7769 - val_loss: 0.5512
Epoch 120/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7690 - loss: 0.5651 - val_accuracy
: 0.7771 - val_loss: 0.5415
Epoch 121/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7720 - loss: 0.5585 - val_accuracy
: 0.7767 - val_loss: 0.5422
Epoch 122/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7701 - loss: 0.5622 - val_accurac
y: 0.7818 - val_loss: 0.5374
Epoch 123/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7715 - loss: 0.5592 - val_accuracy
: 0.7824 - val_loss: 0.5290
Epoch 124/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7735 - loss: 0.5589 - val_accuracy
: 0.7831 - val_loss: 0.5292
Epoch 125/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7690 - loss: 0.5617 - val_accuracy
: 0.7837 - val_loss: 0.5332
Epoch 126/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7723 - loss: 0.5590 - val_accuracy
: 0.7197 - val_loss: 0.7893
Epoch 127/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7731 - loss: 0.5593 - val_accurac
y: 0.7744 - val_loss: 0.5450
Epoch 128/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7712 - loss: 0.5622 - val_accurac
y: 0.7819 - val_loss: 0.5283
Epoch 129/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7741 - loss: 0.5574 - val_accuracy
: 0.7807 - val_loss: 0.5314
Epoch 130/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7707 - loss: 0.5597 - val_accuracy
: 0.7695 - val_loss: 0.6125
Epoch 131/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7742 - loss: 0.5568 - val_accuracy
: 0.7828 - val_loss: 0.5277
Epoch 132/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7732 - loss: 0.5571 - val_accuracy
```

```
: 0.7821 - val_loss: 0.5281
Epoch 133/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7728 - loss: 0.5573 - val_accuracy
: 0.7797 - val_loss: 0.5327
Epoch 134/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7744 - loss: 0.5549 - val_accuracy
: 0.7867 - val_loss: 0.5243
Epoch 135/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7736 - loss: 0.5563 - val_accuracy
: 0.7629 - val_loss: 0.6324
Epoch 136/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7733 - loss: 0.5541 - val_accuracy
: 0.7810 - val_loss: 0.5278
Epoch 137/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7745 - loss: 0.5532 - val_accuracy
: 0.7810 - val_loss: 0.5287
Epoch 138/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7727 - loss: 0.5565 - val_accuracy
: 0.7864 - val_loss: 0.5249
Epoch 139/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7762 - loss: 0.5522 - val_accurac
y: 0.7813 - val_loss: 0.5263
Epoch 140/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7730 - loss: 0.5585 - val_accuracy
: 0.7775 - val_loss: 0.5344
Epoch 141/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7742 - loss: 0.5530 - val_accuracy
: 0.7847 - val_loss: 0.5245
Epoch 142/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7735 - loss: 0.5544 - val_accurac
y: 0.7819 - val_loss: 0.5305
Epoch 143/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7769 - loss: 0.5514 - val_accurac
y: 0.7804 - val_loss: 0.5342
Epoch 144/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7762 - loss: 0.5510 - val_accurac
y: 0.7833 - val_loss: 0.5285
Epoch 145/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7743 - loss: 0.5514 - val_accurac
y: 0.7858 - val_loss: 0.5234
Epoch 146/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7734 - loss: 0.5529 - val_accurac
y: 0.7719 - val_loss: 0.5697
Epoch 147/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7759 - loss: 0.5516 - val_accurac
y: 0.7610 - val_loss: 0.7532
Epoch 148/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7717 - loss: 0.5625 - val_accurac
y: 0.7590 - val_loss: 0.7317
Epoch 149/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7735 - loss: 0.5654 - val_accurac
y: 0.7743 - val_loss: 0.5486
Epoch 150/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7707 - loss: 0.5682 - val_accurac
y: 0.7791 - val_loss: 0.5393
Epoch 151/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7723 - loss: 0.5636 - val_accuracy
: 0.7803 - val_loss: 0.5362
Epoch 152/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7738 - loss: 0.5592 - val_accuracy
: 0.7807 - val_loss: 0.5350
Epoch 153/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7712 - loss: 0.5639 - val_accuracy
: 0.7778 - val_loss: 0.5412
Epoch 154/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7722 - loss: 0.5603 - val_accuracy
: 0.7798 - val_loss: 0.5364
Epoch 155/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7724 - loss: 0.5622 - val_accuracy
: 0.7824 - val_loss: 0.5321
Epoch 156/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7736 - loss: 0.5618 - val_accurac
```

```
y: 0.7438 - val_loss: 0.6548
Epoch 157/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7744 - loss: 0.5590 - val_accuracy
: 0.7700 - val_loss: 0.5573
Epoch 158/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7739 - loss: 0.5609 - val_accuracy
: 0.7799 - val_loss: 0.5375
Epoch 159/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7719 - loss: 0.5623 - val_accuracy
: 0.7822 - val_loss: 0.5304
Epoch 160/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7729 - loss: 0.5613 - val_accuracy
: 0.7819 - val_loss: 0.5318
Epoch 161/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7740 - loss: 0.5605 - val_accurac
y: 0.7821 - val_loss: 0.5326
Epoch 162/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7735 - loss: 0.5601 - val_accuracy
: 0.7812 - val_loss: 0.5332
Epoch 163/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7739 - loss: 0.5600 - val_accuracy
: 0.7826 - val_loss: 0.5312
Epoch 164/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7732 - loss: 0.5581 - val_accuracy
: 0.7773 - val_loss: 0.5372
Epoch 165/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7733 - loss: 0.5600 - val_accuracy
: 0.7825 - val_loss: 0.5287
Epoch 166/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7769 - loss: 0.5555 - val_accurac
y: 0.7550 - val_loss: 0.5853
Epoch 167/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 4s 16ms/step - accuracy: 0.7733 - loss: 0.5594 - val_accurac
y: 0.7680 - val_loss: 0.5609
Epoch 168/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7730 - loss: 0.5564 - val_accurac
y: 0.7803 - val_loss: 0.5328
Epoch 169/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7747 - loss: 0.5592 - val_accuracy
: 0.7838 - val_loss: 0.5308
Epoch 170/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7719 - loss: 0.5613 - val_accuracy
: 0.7851 - val_loss: 0.5252
Epoch 171/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7752 - loss: 0.5554 - val_accuracy
: 0.7809 - val_loss: 0.5324
Epoch 172/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7735 - loss: 0.5576 - val_accuracy
: 0.7841 - val_loss: 0.5298
Epoch 173/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7733 - loss: 0.5606 - val_accuracy
: 0.7843 - val_loss: 0.5276
Epoch 174/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7760 - loss: 0.5572 - val_accuracy
: 0.7802 - val_loss: 0.5309
Epoch 175/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7737 - loss: 0.5563 - val_accuracy
: 0.7804 - val_loss: 0.5316
Epoch 176/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7717 - loss: 0.5601 - val_accuracy
: 0.7863 - val_loss: 0.5240
Epoch 177/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7731 - loss: 0.5578 - val_accuracy
: 0.7875 - val_loss: 0.5237
Epoch 178/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7745 - loss: 0.5545 - val_accuracy
: 0.7862 - val_loss: 0.5218
Epoch 179/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7753 - loss: 0.5534 - val_accuracy
: 0.7841 - val_loss: 0.5263
Epoch 180/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7747 - loss: 0.5559 - val_accuracy
```

```
: 0.7777 - val_loss: 0.5333
Epoch 181/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7740 - loss: 0.5596 - val_accuracy
: 0.7819 - val_loss: 0.5325
Epoch 182/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7730 - loss: 0.5591 - val_accuracy
: 0.7860 - val_loss: 0.5242
Epoch 183/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7740 - loss: 0.5567 - val_accuracy
: 0.7854 - val_loss: 0.5228
Epoch 184/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7759 - loss: 0.5525 - val_accuracy
: 0.7846 - val_loss: 0.5239
Epoch 185/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7755 - loss: 0.5531 - val_accuracy
: 0.7861 - val_loss: 0.5203
Epoch 186/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7757 - loss: 0.5528 - val_accuracy
: 0.7866 - val_loss: 0.5183
Epoch 187/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7760 - loss: 0.5496 - val_accuracy
: 0.7255 - val_loss: 0.7557
Epoch 188/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7757 - loss: 0.5556 - val_accuracy
: 0.7781 - val_loss: 0.5436
Epoch 189/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7748 - loss: 0.5555 - val_accuracy
: 0.7829 - val_loss: 0.5252
Epoch 190/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7745 - loss: 0.5563 - val_accuracy
: 0.7793 - val_loss: 0.5339
Epoch 191/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7751 - loss: 0.5530 - val_accuracy
: 0.7844 - val_loss: 0.5247
Epoch 192/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7733 - loss: 0.5563 - val_accuracy
: 0.7882 - val_loss: 0.5195
Epoch 193/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7743 - loss: 0.5568 - val_accuracy
: 0.7861 - val_loss: 0.5226
Epoch 194/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7754 - loss: 0.5568 - val_accuracy
: 0.7825 - val_loss: 0.5249
Epoch 195/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7757 - loss: 0.5533 - val_accuracy
: 0.7842 - val_loss: 0.5250
Epoch 196/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7761 - loss: 0.5520 - val_accuracy
: 0.7860 - val_loss: 0.5233
Epoch 197/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7753 - loss: 0.5540 - val_accuracy
: 0.7819 - val_loss: 0.5269
Epoch 198/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7753 - loss: 0.5529 - val_accuracy
: 0.7876 - val_loss: 0.5221
Epoch 199/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7761 - loss: 0.5515 - val_accuracy
: 0.7872 - val_loss: 0.5218
Epoch 200/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7774 - loss: 0.5508 - val_accuracy
: 0.7894 - val_loss: 0.5159
Epoch 201/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7765 - loss: 0.5525 - val_accuracy
: 0.7562 - val_loss: 0.6090
Epoch 202/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7742 - loss: 0.5544 - val_accuracy
: 0.7871 - val_loss: 0.5218
Epoch 203/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7786 - loss: 0.5460 - val_accuracy
: 0.7875 - val_loss: 0.5169
Epoch 204/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7776 - loss: 0.5486 - val_accuracy
```

```
: 0.7899 - val_loss: 0.5143
Epoch 205/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7786 - loss: 0.5466 - val_accuracy
: 0.7839 - val_loss: 0.5240
Epoch 206/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7786 - loss: 0.5442 - val_accuracy
: 0.7818 - val_loss: 0.5266
Epoch 207/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7794 - loss: 0.5446 - val_accuracy
: 0.7885 - val_loss: 0.5139
Epoch 208/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7774 - loss: 0.5418 - val_accuracy
: 0.7822 - val_loss: 0.5239
Epoch 209/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7774 - loss: 0.5486 - val_accuracy
: 0.7902 - val_loss: 0.5132
Epoch 210/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7799 - loss: 0.5437 - val_accuracy
: 0.7907 - val_loss: 0.5099
Epoch 211/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7790 - loss: 0.5443 - val_accuracy
: 0.7908 - val_loss: 0.5082
Epoch 212/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7787 - loss: 0.5427 - val_accuracy
: 0.7913 - val_loss: 0.5122
Epoch 213/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7779 - loss: 0.5441 - val_accuracy
: 0.7880 - val_loss: 0.5131
Epoch 214/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7807 - loss: 0.5416 - val_accuracy
: 0.7890 - val_loss: 0.5131
Epoch 215/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7781 - loss: 0.5450 - val_accuracy
: 0.7916 - val_loss: 0.5083
Epoch 216/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7807 - loss: 0.5407 - val_accuracy
: 0.7891 - val_loss: 0.5145
Epoch 217/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7797 - loss: 0.5400 - val_accuracy
: 0.7897 - val_loss: 0.5171
Epoch 218/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7815 - loss: 0.5402 - val_accuracy
: 0.7896 - val_loss: 0.5071
Epoch 219/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7797 - loss: 0.5418 - val_accuracy
: 0.7897 - val_loss: 0.5085
Epoch 220/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7810 - loss: 0.5381 - val_accuracy
: 0.7904 - val_loss: 0.5109
Epoch 221/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7806 - loss: 0.5407 - val_accuracy
: 0.7891 - val_loss: 0.5108
Epoch 222/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7795 - loss: 0.5409 - val_accuracy
: 0.7903 - val_loss: 0.5096
Epoch 223/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7806 - loss: 0.5400 - val_accuracy
: 0.7877 - val_loss: 0.5181
Epoch 224/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7813 - loss: 0.5393 - val_accuracy
: 0.7934 - val_loss: 0.5072
Epoch 225/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7810 - loss: 0.5415 - val_accuracy
: 0.7869 - val_loss: 0.5337
Epoch 226/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7803 - loss: 0.5406 - val_accuracy
: 0.7907 - val_loss: 0.5147
Epoch 227/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7802 - loss: 0.5399 - val_accuracy
: 0.7883 - val_loss: 0.5132
Epoch 228/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7813 - loss: 0.5398 - val_accuracy
```

```
: 0.7951 - val_loss: 0.5064
Epoch 229/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7820 - loss: 0.5343 - val_accuracy
: 0.7923 - val_loss: 0.5049
Epoch 230/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7791 - loss: 0.5415 - val_accuracy
: 0.6377 - val_loss: 1.4245
Epoch 231/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7750 - loss: 0.5533 - val_accuracy
: 0.7792 - val_loss: 0.5456
Epoch 232/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7768 - loss: 0.5524 - val_accuracy
: 0.7865 - val_loss: 0.5214
Epoch 233/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7771 - loss: 0.5499 - val_accuracy
: 0.7877 - val_loss: 0.5187
Epoch 234/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7790 - loss: 0.5468 - val_accuracy
: 0.7870 - val_loss: 0.5152
Epoch 235/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7787 - loss: 0.5479 - val_accuracy
: 0.7862 - val_loss: 0.5214
Epoch 236/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7767 - loss: 0.5456 - val_accuracy
: 0.7896 - val_loss: 0.5135
Epoch 237/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7780 - loss: 0.5489 - val_accuracy
: 0.7923 - val_loss: 0.5076
Epoch 238/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7793 - loss: 0.5468 - val_accuracy
: 0.7846 - val_loss: 0.5188
Epoch 239/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7793 - loss: 0.5465 - val_accuracy
: 0.7905 - val_loss: 0.5114
Epoch 240/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7784 - loss: 0.5482 - val_accuracy
: 0.7877 - val_loss: 0.5161
Epoch 241/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7762 - loss: 0.5509 - val_accuracy
: 0.7845 - val_loss: 0.5203
Epoch 242/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7792 - loss: 0.5432 - val_accuracy
: 0.7898 - val_loss: 0.5101
Epoch 243/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7804 - loss: 0.5431 - val_accuracy
: 0.7912 - val_loss: 0.5077
Epoch 244/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7769 - loss: 0.5480 - val_accuracy
: 0.7903 - val_loss: 0.5098
Epoch 245/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7824 - loss: 0.5382 - val_accuracy
: 0.7887 - val_loss: 0.5118
Epoch 246/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7807 - loss: 0.5411 - val_accuracy
: 0.7905 - val_loss: 0.5103
Epoch 247/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7806 - loss: 0.5405 - val_accuracy
: 0.7897 - val_loss: 0.5117
Epoch 248/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7793 - loss: 0.5449 - val_accuracy
: 0.7908 - val_loss: 0.5082
Epoch 249/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7806 - loss: 0.5417 - val_accurac
y: 0.7876 - val_loss: 0.5104
Epoch 250/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7820 - loss: 0.5383 - val_accurac
y: 0.7308 - val_loss: 0.6865
Epoch 251/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7802 - loss: 0.5446 - val_accuracy
: 0.7833 - val_loss: 0.5267
Epoch 252/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7780 - loss: 0.5458 - val_accuracy
```

```
: 0.7910 - val_loss: 0.5060
Epoch 253/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7808 - loss: 0.5383 - val_accuracy
: 0.7893 - val_loss: 0.5126
Epoch 254/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7800 - loss: 0.5402 - val_accuracy
: 0.7895 - val_loss: 0.5130
Epoch 255/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7815 - loss: 0.5381 - val_accuracy
: 0.7921 - val_loss: 0.5108
Epoch 256/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7808 - loss: 0.5396 - val_accuracy
: 0.7888 - val_loss: 0.5087
Epoch 257/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7822 - loss: 0.5380 - val_accuracy
: 0.7949 - val_loss: 0.5001
Epoch 258/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7822 - loss: 0.5377 - val_accuracy
: 0.7900 - val_loss: 0.5062
Epoch 259/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7809 - loss: 0.5366 - val_accuracy
: 0.7932 - val_loss: 0.5053
Epoch 260/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7816 - loss: 0.5358 - val_accuracy
: 0.7943 - val_loss: 0.5011
Epoch 261/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7799 - loss: 0.5388 - val_accuracy
: 0.7910 - val_loss: 0.5062
Epoch 262/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7809 - loss: 0.5377 - val_accuracy
: 0.7950 - val_loss: 0.5056
Epoch 263/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7815 - loss: 0.5342 - val_accuracy
: 0.7925 - val_loss: 0.5049
Epoch 264/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7825 - loss: 0.5358 - val_accuracy
: 0.7909 - val_loss: 0.5053
Epoch 265/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7804 - loss: 0.5383 - val_accuracy
: 0.7944 - val_loss: 0.5042
Epoch 266/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7823 - loss: 0.5373 - val_accuracy
: 0.7936 - val_loss: 0.5055
Epoch 267/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7824 - loss: 0.5344 - val_accuracy
: 0.7943 - val_loss: 0.4976
Epoch 268/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7821 - loss: 0.5361 - val_accuracy
: 0.7963 - val_loss: 0.4978
Epoch 269/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7839 - loss: 0.5311 - val_accuracy
: 0.7962 - val_loss: 0.5041
Epoch 270/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7841 - loss: 0.5318 - val_accuracy
: 0.7962 - val_loss: 0.4995
Epoch 271/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7825 - loss: 0.5346 - val_accuracy
: 0.7589 - val_loss: 0.7656
Epoch 272/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7814 - loss: 0.5371 - val_accuracy
: 0.7880 - val_loss: 0.5169
Epoch 273/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7827 - loss: 0.5389 - val_accuracy
: 0.7953 - val_loss: 0.4975
Epoch 274/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7790 - loss: 0.5438 - val_accuracy
: 0.7946 - val_loss: 0.5006
Epoch 275/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7817 - loss: 0.5383 - val_accuracy
: 0.7947 - val_loss: 0.5022
Epoch 276/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7809 - loss: 0.5390 - val_accuracy
```

```
: 0.7934 - val_loss: 0.5038
Epoch 277/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7797 - loss: 0.5400 - val_accuracy
: 0.7950 - val_loss: 0.5013
Epoch 278/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7811 - loss: 0.5361 - val_accuracy
: 0.7937 - val_loss: 0.5016
Epoch 279/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7817 - loss: 0.5353 - val_accuracy
: 0.7947 - val_loss: 0.4998
Epoch 280/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7805 - loss: 0.5378 - val_accuracy
: 0.7480 - val_loss: 0.7985
Epoch 281/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7774 - loss: 0.5492 - val_accuracy
: 0.7858 - val_loss: 0.5250
Epoch 282/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7797 - loss: 0.5450 - val_accuracy
: 0.7903 - val_loss: 0.5095
Epoch 283/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7768 - loss: 0.5454 - val_accuracy
: 0.7918 - val_loss: 0.5057
Epoch 284/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7784 - loss: 0.5453 - val_accuracy
: 0.7924 - val_loss: 0.5086
Epoch 285/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7824 - loss: 0.5420 - val_accuracy
: 0.7921 - val_loss: 0.5076
Epoch 286/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7810 - loss: 0.5428 - val_accuracy
: 0.7911 - val_loss: 0.5095
Epoch 287/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7803 - loss: 0.5436 - val_accuracy
: 0.7920 - val_loss: 0.5064
Epoch 288/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7783 - loss: 0.5451 - val_accurac
y: 0.7934 - val_loss: 0.5076
Epoch 289/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7776 - loss: 0.5503 - val_accurac
y: 0.7914 - val_loss: 0.5095
Epoch 290/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7810 - loss: 0.5400 - val_accuracy
: 0.7930 - val_loss: 0.5033
Epoch 291/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7820 - loss: 0.5380 - val_accuracy
: 0.7923 - val_loss: 0.5053
Epoch 292/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7818 - loss: 0.5409 - val_accuracy
: 0.7888 - val_loss: 0.5098
Epoch 293/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7816 - loss: 0.5389 - val_accuracy
: 0.7919 - val_loss: 0.5082
Epoch 294/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7798 - loss: 0.5430 - val_accuracy
: 0.7927 - val_loss: 0.5035
Epoch 295/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7790 - loss: 0.5403 - val_accuracy
: 0.7920 - val_loss: 0.5046
Epoch 296/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7830 - loss: 0.5365 - val_accuracy
: 0.7944 - val_loss: 0.5006
Epoch 297/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7828 - loss: 0.5394 - val_accurac
y: 0.7916 - val_loss: 0.5058
Epoch 298/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7824 - loss: 0.5415 - val_accuracy
: 0.7915 - val_loss: 0.5069
Epoch 299/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7804 - loss: 0.5427 - val_accuracy
: 0.7806 - val_loss: 0.5287
Epoch 300/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7798 - loss: 0.5425 - val_accuracy
```

```
: 0.7921 - val_loss: 0.5055
Epoch 301/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7818 - loss: 0.5398 - val_accuracy
: 0.7904 - val_loss: 0.5035
Epoch 302/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7833 - loss: 0.5350 - val_accuracy
: 0.7911 - val_loss: 0.5085
Epoch 303/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7817 - loss: 0.5413 - val_accuracy
: 0.7935 - val_loss: 0.5050
Epoch 304/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7820 - loss: 0.5392 - val_accuracy
: 0.7944 - val_loss: 0.5041
Epoch 305/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7796 - loss: 0.5423 - val_accuracy
: 0.7904 - val_loss: 0.5105
Epoch 306/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7834 - loss: 0.5388 - val_accuracy
: 0.7921 - val_loss: 0.5046
Epoch 307/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7821 - loss: 0.5373 - val_accuracy
: 0.7945 - val_loss: 0.5027
Epoch 308/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7815 - loss: 0.5407 - val_accuracy
: 0.7913 - val_loss: 0.5041
Epoch 309/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7810 - loss: 0.5363 - val_accuracy
: 0.7932 - val_loss: 0.5012
Epoch 310/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7814 - loss: 0.5389 - val_accuracy
: 0.7916 - val_loss: 0.5022
Epoch 311/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7828 - loss: 0.5340 - val_accuracy
: 0.7967 - val_loss: 0.4957
Epoch 312/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7829 - loss: 0.5341 - val_accuracy
: 0.7922 - val_loss: 0.5096
Epoch 313/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7839 - loss: 0.5337 - val_accuracy
: 0.7967 - val_loss: 0.4990
Epoch 314/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7829 - loss: 0.5333 - val_accuracy
: 0.7927 - val_loss: 0.5057
Epoch 315/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7833 - loss: 0.5329 - val_accuracy
: 0.7974 - val_loss: 0.4947
Epoch 316/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7836 - loss: 0.5364 - val_accuracy
: 0.7933 - val_loss: 0.4971
Epoch 317/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7840 - loss: 0.5324 - val_accuracy
: 0.7981 - val_loss: 0.4979
Epoch 318/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7869 - loss: 0.5257 - val_accuracy
: 0.7973 - val_loss: 0.4964
Epoch 319/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7842 - loss: 0.5330 - val_accuracy
: 0.7928 - val_loss: 0.5000
Epoch 320/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7845 - loss: 0.5299 - val_accuracy
: 0.7960 - val_loss: 0.4936
Epoch 321/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7868 - loss: 0.5250 - val_accurac
y: 0.8000 - val_loss: 0.4900
Epoch 322/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7846 - loss: 0.5305 - val_accuracy
: 0.7977 - val_loss: 0.4925
Epoch 323/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7839 - loss: 0.5330 - val_accuracy
: 0.7991 - val_loss: 0.4937
Epoch 324/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7864 - loss: 0.5265 - val_accuracy
```

```
: 0.7990 - val_loss: 0.4902
Epoch 325/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7829 - loss: 0.5318 - val_accuracy
: 0.7986 - val_loss: 0.4958
Epoch 326/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7837 - loss: 0.5315 - val_accuracy
: 0.7981 - val_loss: 0.4942
Epoch 327/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7839 - loss: 0.5287 - val_accuracy
: 0.7991 - val_loss: 0.4905
Epoch 328/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7859 - loss: 0.5278 - val_accuracy
: 0.8000 - val_loss: 0.4948
Epoch 329/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7855 - loss: 0.5262 - val_accuracy
: 0.6449 - val_loss: 1.4825
Epoch 330/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7816 - loss: 0.5417 - val_accuracy
: 0.7837 - val_loss: 0.5310
Epoch 331/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7827 - loss: 0.5405 - val_accuracy
: 0.7902 - val_loss: 0.5072
Epoch 332/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7843 - loss: 0.5378 - val_accuracy
: 0.7961 - val_loss: 0.5016
Epoch 333/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7832 - loss: 0.5361 - val_accuracy
: 0.7956 - val_loss: 0.4999
Epoch 334/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7841 - loss: 0.5375 - val_accuracy
: 0.7966 - val_loss: 0.5005
Epoch 335/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7825 - loss: 0.5381 - val_accuracy
: 0.7935 - val_loss: 0.5049
Epoch 336/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7827 - loss: 0.5377 - val_accuracy
: 0.7949 - val_loss: 0.4987
Epoch 337/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7831 - loss: 0.5373 - val_accuracy
: 0.7862 - val_loss: 0.5134
Epoch 338/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7835 - loss: 0.5347 - val_accuracy
: 0.7948 - val_loss: 0.5010
Epoch 339/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7829 - loss: 0.5361 - val_accuracy
: 0.7950 - val_loss: 0.5001
Epoch 340/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7814 - loss: 0.5375 - val_accuracy
: 0.7951 - val_loss: 0.5003
Epoch 341/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7821 - loss: 0.5378 - val_accuracy
: 0.7950 - val_loss: 0.4979
Epoch 342/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7853 - loss: 0.5344 - val_accuracy
: 0.7976 - val_loss: 0.4950
Epoch 343/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7820 - loss: 0.5391 - val_accuracy
: 0.7964 - val_loss: 0.4989
Epoch 344/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7809 - loss: 0.5371 - val_accuracy
: 0.7931 - val_loss: 0.5032
Epoch 345/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7858 - loss: 0.5331 - val_accuracy
: 0.7864 - val_loss: 0.5128
Epoch 346/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7867 - loss: 0.5284 - val_accuracy
: 0.7978 - val_loss: 0.4950
Epoch 347/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7822 - loss: 0.5372 - val_accuracy
: 0.7942 - val_loss: 0.5018
Epoch 348/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7835 - loss: 0.5320 - val_accuracy
```

```
: 0.7953 - val_loss: 0.4979
Epoch 349/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7852 - loss: 0.5313 - val_accuracy
: 0.7964 - val_loss: 0.4998
Epoch 350/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7809 - loss: 0.5392 - val_accuracy
: 0.7967 - val_loss: 0.4990
Epoch 351/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7856 - loss: 0.5288 - val_accuracy
: 0.7944 - val_loss: 0.5010
Epoch 352/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7828 - loss: 0.5371 - val_accuracy
: 0.7957 - val_loss: 0.5001
Epoch 353/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7823 - loss: 0.5358 - val_accuracy
: 0.7949 - val_loss: 0.4968
Epoch 354/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7844 - loss: 0.5325 - val_accuracy
: 0.7955 - val_loss: 0.4972
Epoch 355/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7825 - loss: 0.5375 - val_accuracy
: 0.7873 - val_loss: 0.5144
Epoch 356/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7833 - loss: 0.5346 - val_accuracy
: 0.7714 - val_loss: 0.5655
Epoch 357/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7843 - loss: 0.5308 - val_accuracy
: 0.7919 - val_loss: 0.5103
Epoch 358/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7798 - loss: 0.5392 - val_accuracy
: 0.7948 - val_loss: 0.5005
Epoch 359/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7822 - loss: 0.5360 - val_accuracy
: 0.7959 - val_loss: 0.5008
Epoch 360/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7809 - loss: 0.5404 - val_accuracy
: 0.7982 - val_loss: 0.4957
Epoch 361/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7822 - loss: 0.5374 - val_accuracy
: 0.7965 - val_loss: 0.4959
Epoch 362/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7847 - loss: 0.5342 - val_accuracy
: 0.7978 - val_loss: 0.4954
Epoch 363/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7842 - loss: 0.5323 - val_accuracy
: 0.7946 - val_loss: 0.5002
Epoch 364/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7825 - loss: 0.5362 - val_accuracy
: 0.7981 - val_loss: 0.4951
Epoch 365/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7829 - loss: 0.5349 - val_accuracy
: 0.7939 - val_loss: 0.5001
Epoch 366/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7818 - loss: 0.5381 - val_accuracy
: 0.7953 - val_loss: 0.5014
Epoch 367/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7849 - loss: 0.5329 - val_accuracy
: 0.7961 - val_loss: 0.4968
Epoch 368/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7838 - loss: 0.5348 - val_accuracy
: 0.7985 - val_loss: 0.4953
Epoch 369/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7861 - loss: 0.5296 - val_accuracy
: 0.7953 - val_loss: 0.4966
Epoch 370/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7855 - loss: 0.5329 - val_accuracy
: 0.7968 - val_loss: 0.4975
Epoch 371/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7814 - loss: 0.5365 - val_accuracy
: 0.7976 - val_loss: 0.4942
Epoch 372/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7831 - loss: 0.5337 - val_accuracy
```

```
: 0.7975 - val_loss: 0.4917
Epoch 373/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7843 - loss: 0.5326 - val_accuracy
: 0.7982 - val_loss: 0.4928
Epoch 374/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7859 - loss: 0.5306 - val_accuracy
: 0.7973 - val_loss: 0.4951
Epoch 375/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7828 - loss: 0.5333 - val_accuracy
: 0.7981 - val_loss: 0.4921
Epoch 376/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7817 - loss: 0.5348 - val_accuracy
: 0.7972 - val_loss: 0.4958
Epoch 377/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7840 - loss: 0.5329 - val_accuracy
: 0.7983 - val_loss: 0.4924
Epoch 378/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7836 - loss: 0.5338 - val_accuracy
: 0.8015 - val_loss: 0.4919
Epoch 379/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7848 - loss: 0.5319 - val_accuracy
: 0.7967 - val_loss: 0.4959
Epoch 380/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7844 - loss: 0.5345 - val_accuracy
: 0.7966 - val_loss: 0.4957
Epoch 381/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7842 - loss: 0.5334 - val_accuracy
: 0.7993 - val_loss: 0.4937
Epoch 382/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7858 - loss: 0.5332 - val_accuracy
: 0.7920 - val_loss: 0.5072
Epoch 383/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7842 - loss: 0.5304 - val_accuracy
: 0.7995 - val_loss: 0.4906
Epoch 384/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7848 - loss: 0.5320 - val_accurac
: 0.7956 - val_loss: 0.4980
Epoch 385/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7846 - loss: 0.5303 - val_accuracy
: 0.7976 - val_loss: 0.4942
Epoch 386/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7855 - loss: 0.5285 - val_accuracy
: 0.7971 - val_loss: 0.4926
Epoch 387/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7844 - loss: 0.5321 - val_accuracy
: 0.7995 - val_loss: 0.4936
Epoch 388/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7835 - loss: 0.5307 - val_accuracy
: 0.7970 - val_loss: 0.4963
Epoch 389/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7850 - loss: 0.5305 - val_accuracy
: 0.7976 - val_loss: 0.4930
Epoch 390/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7837 - loss: 0.5311 - val_accuracy
: 0.7987 - val_loss: 0.4917
Epoch 391/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7852 - loss: 0.5305 - val_accuracy
: 0.7976 - val_loss: 0.4930
Epoch 392/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7835 - loss: 0.5323 - val_accuracy
: 0.7999 - val_loss: 0.4945
Epoch 393/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7837 - loss: 0.5311 - val_accuracy
: 0.7995 - val_loss: 0.4946
Epoch 394/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7871 - loss: 0.5304 - val_accuracy
: 0.7969 - val_loss: 0.4970
Epoch 395/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7853 - loss: 0.5296 - val_accuracy
: 0.7983 - val_loss: 0.4923
Epoch 396/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7892 - loss: 0.5225 - val_accurac
```

```
y: 0.8003 - val_loss: 0.4937
Epoch 397/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7864 - loss: 0.5309 - val_accurac
y: 0.7942 - val_loss: 0.5061
Epoch 398/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7858 - loss: 0.5290 - val_accurac
y: 0.7925 - val_loss: 0.5068
Epoch 399/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7838 - loss: 0.5342 - val_accuracy
: 0.7950 - val_loss: 0.5003
Epoch 400/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7869 - loss: 0.5275 - val_accuracy
: 0.8012 - val_loss: 0.4880
Epoch 401/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7858 - loss: 0.5286 - val_accuracy
: 0.7993 - val_loss: 0.4895
Epoch 402/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7855 - loss: 0.5315 - val_accuracy
: 0.7981 - val_loss: 0.4955
Epoch 403/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7854 - loss: 0.5318 - val_accuracy
: 0.7996 - val_loss: 0.4916
Epoch 404/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7833 - loss: 0.5318 - val_accuracy
: 0.8005 - val_loss: 0.4919
Epoch 405/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7827 - loss: 0.5313 - val_accuracy
: 0.7983 - val_loss: 0.4952
Epoch 406/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7836 - loss: 0.5318 - val_accuracy
: 0.7996 - val_loss: 0.4947
Epoch 407/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7839 - loss: 0.5305 - val_accuracy
: 0.7961 - val_loss: 0.4972
Epoch 408/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7857 - loss: 0.5310 - val_accuracy
: 0.8011 - val_loss: 0.4887
Epoch 409/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7866 - loss: 0.5307 - val_accuracy
: 0.8024 - val_loss: 0.4890
Epoch 410/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7832 - loss: 0.5309 - val_accuracy
: 0.8009 - val_loss: 0.4893
Epoch 411/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7836 - loss: 0.5324 - val_accurac
y: 0.7991 - val_loss: 0.4931
Epoch 412/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7845 - loss: 0.5301 - val_accurac
y: 0.8003 - val_loss: 0.4901
Epoch 413/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7878 - loss: 0.5269 - val_accuracy
: 0.8024 - val_loss: 0.4864
Epoch 414/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7879 - loss: 0.5232 - val_accuracy
: 0.7981 - val_loss: 0.4936
Epoch 415/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7882 - loss: 0.5256 - val_accurac
y: 0.8014 - val_loss: 0.4896
Epoch 416/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step - accuracy: 0.7855 - loss: 0.5308 - val_accurac
y: 0.7997 - val_loss: 0.4881
Epoch 417/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7846 - loss: 0.5293 - val_accurac
y: 0.7998 - val_loss: 0.4894
Epoch 418/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7881 - loss: 0.5260 - val_accurac
y: 0.8001 - val_loss: 0.4911
Epoch 419/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7874 - loss: 0.5256 - val_accurac
y: 0.7981 - val_loss: 0.4918
Epoch 420/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7856 - loss: 0.5296 - val_accurac
```

```
y: 0.8037 - val_loss: 0.4856
Epoch 421/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7858 - loss: 0.5289 - val_accurac
y: 0.7840 - val_loss: 0.5159
Epoch 422/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7837 - loss: 0.5327 - val_accurac
y: 0.7970 - val_loss: 0.4965
Epoch 423/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7849 - loss: 0.5300 - val_accurac
y: 0.7979 - val_loss: 0.4936
Epoch 424/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7835 - loss: 0.5319 - val_accurac
y: 0.8009 - val_loss: 0.4898
Epoch 425/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7853 - loss: 0.5284 - val_accuracy
: 0.8012 - val_loss: 0.4866
Epoch 426/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7886 - loss: 0.5248 - val_accuracy
: 0.7960 - val_loss: 0.4970
Epoch 427/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7865 - loss: 0.5263 - val_accurac
y: 0.8033 - val_loss: 0.4868
Epoch 428/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7857 - loss: 0.5263 - val_accurac
y: 0.8019 - val_loss: 0.4873
Epoch 429/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7863 - loss: 0.5309 - val_accurac
y: 0.7998 - val_loss: 0.4887
Epoch 430/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7860 - loss: 0.5291 - val_accurac
y: 0.8033 - val_loss: 0.4876
Epoch 431/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7852 - loss: 0.5274 - val_accurac
y: 0.8034 - val_loss: 0.4881
Epoch 432/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7882 - loss: 0.5234 - val_accuracy
: 0.8019 - val_loss: 0.4890
Epoch 433/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7882 - loss: 0.5256 - val_accuracy
: 0.8018 - val_loss: 0.4854
Epoch 434/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7855 - loss: 0.5276 - val_accuracy
: 0.8027 - val_loss: 0.4843
Epoch 435/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7877 - loss: 0.5268 - val_accuracy
: 0.8009 - val_loss: 0.4889
Epoch 436/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7851 - loss: 0.5293 - val_accuracy
: 0.8006 - val_loss: 0.4907
Epoch 437/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7867 - loss: 0.5288 - val_accurac
y: 0.8013 - val_loss: 0.4874
Epoch 438/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7866 - loss: 0.5257 - val_accurac
y: 0.8026 - val_loss: 0.4845
Epoch 439/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7879 - loss: 0.5250 - val_accurac
y: 0.8014 - val_loss: 0.4857
Epoch 440/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7848 - loss: 0.5276 - val_accurac
y: 0.8019 - val_loss: 0.4851
Epoch 441/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7856 - loss: 0.5270 - val_accurac
y: 0.7998 - val_loss: 0.4887
Epoch 442/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7841 - loss: 0.5288 - val_accurac
y: 0.8000 - val_loss: 0.4869
Epoch 443/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7882 - loss: 0.5214 - val_accurac
y: 0.8035 - val_loss: 0.4813
Epoch 444/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7833 - loss: 0.5302 - val_accurac
```

```
y: 0.8023 - val_loss: 0.4867
Epoch 445/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7873 - loss: 0.5252 - val_accurac
y: 0.7980 - val_loss: 0.4916
Epoch 446/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7858 - loss: 0.5251 - val_accurac
y: 0.7926 - val_loss: 0.5020
Epoch 447/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7856 - loss: 0.5301 - val_accuracy
: 0.8028 - val_loss: 0.4857
Epoch 448/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7851 - loss: 0.5288 - val_accurac
y: 0.8007 - val_loss: 0.4873
Epoch 449/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7876 - loss: 0.5222 - val_accurac
y: 0.8015 - val_loss: 0.4881
Epoch 450/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7868 - loss: 0.5278 - val_accurac
y: 0.8018 - val_loss: 0.4870
Epoch 451/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7892 - loss: 0.5210 - val_accurac
y: 0.8012 - val_loss: 0.4871
Epoch 452/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7850 - loss: 0.5255 - val_accurac
y: 0.8028 - val_loss: 0.4846
Epoch 453/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7883 - loss: 0.5234 - val_accuracy
: 0.8012 - val_loss: 0.4854
Epoch 454/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7892 - loss: 0.5219 - val_accuracy
: 0.8009 - val_loss: 0.4862
Epoch 455/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7864 - loss: 0.5239 - val_accurac
y: 0.8008 - val_loss: 0.4869
Epoch 456/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7866 - loss: 0.5253 - val_accurac
y: 0.7952 - val_loss: 0.4962
Epoch 457/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7860 - loss: 0.5261 - val_accurac
y: 0.7932 - val_loss: 0.4961
Epoch 458/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7887 - loss: 0.5240 - val_accurac
y: 0.7383 - val_loss: 0.6356
Epoch 459/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7833 - loss: 0.5288 - val_accurac
y: 0.7901 - val_loss: 0.5056
Epoch 460/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7848 - loss: 0.5300 - val_accurac
y: 0.8018 - val_loss: 0.4840
Epoch 461/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7869 - loss: 0.5280 - val_accurac
y: 0.7993 - val_loss: 0.4869
Epoch 462/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7853 - loss: 0.5291 - val_accurac
y: 0.8017 - val_loss: 0.4858
Epoch 463/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7866 - loss: 0.5277 - val_accurac
y: 0.7996 - val_loss: 0.4866
Epoch 464/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7864 - loss: 0.5260 - val_accurac
y: 0.8023 - val_loss: 0.4852
Epoch 465/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7871 - loss: 0.5250 - val_accurac
y: 0.7980 - val_loss: 0.4898
Epoch 466/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7878 - loss: 0.5254 - val_accurac
y: 0.8010 - val_loss: 0.4884
Epoch 467/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7863 - loss: 0.5238 - val_accurac
y: 0.8034 - val_loss: 0.4851
Epoch 468/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7831 - loss: 0.5301 - val_accurac
```

```
y: 0.7991 - val_loss: 0.4880
Epoch 469/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7861 - loss: 0.5263 - val_accurac
y: 0.8007 - val_loss: 0.4852
Epoch 470/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7889 - loss: 0.5220 - val_accurac
y: 0.8016 - val_loss: 0.4859
Epoch 471/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7863 - loss: 0.5272 - val_accurac
y: 0.8025 - val_loss: 0.4827
Epoch 472/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7857 - loss: 0.5281 - val_accurac
y: 0.8007 - val_loss: 0.4874
Epoch 473/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7839 - loss: 0.5298 - val_accurac
y: 0.8047 - val_loss: 0.4818
Epoch 474/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7867 - loss: 0.5249 - val_accurac
y: 0.8018 - val_loss: 0.4836
Epoch 475/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7884 - loss: 0.5260 - val_accurac
y: 0.8032 - val_loss: 0.4871
Epoch 476/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7861 - loss: 0.5236 - val_accurac
y: 0.8032 - val_loss: 0.4834
Epoch 477/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7868 - loss: 0.5260 - val_accurac
y: 0.8020 - val_loss: 0.4863
Epoch 478/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7863 - loss: 0.5257 - val_accurac
y: 0.8000 - val_loss: 0.4890
Epoch 479/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7868 - loss: 0.5238 - val_accurac
y: 0.8028 - val_loss: 0.4836
Epoch 480/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7840 - loss: 0.5279 - val_accurac
y: 0.8047 - val_loss: 0.4802
Epoch 481/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7864 - loss: 0.5253 - val_accurac
y: 0.8022 - val_loss: 0.4839
Epoch 482/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7867 - loss: 0.5271 - val_accurac
y: 0.8026 - val_loss: 0.4850
Epoch 483/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7863 - loss: 0.5241 - val_accurac
y: 0.8022 - val_loss: 0.4851
Epoch 484/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7879 - loss: 0.5247 - val_accurac
y: 0.8038 - val_loss: 0.4815
Epoch 485/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7866 - loss: 0.5231 - val_accurac
y: 0.8030 - val_loss: 0.4832
Epoch 486/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 3s 13ms/step - accuracy: 0.7849 - loss: 0.5246 - val_accurac
y: 0.8045 - val_loss: 0.4835
Epoch 487/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - accuracy: 0.7855 - loss: 0.5266 - val_accurac
y: 0.8026 - val_loss: 0.4894
Epoch 488/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - accuracy: 0.7845 - loss: 0.5248 - val_accurac
y: 0.8045 - val_loss: 0.4825
Epoch 489/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - accuracy: 0.7902 - loss: 0.5193 - val_accurac
y: 0.8056 - val_loss: 0.4801
Epoch 490/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7916 - loss: 0.5160 - val_accurac
y: 0.8027 - val_loss: 0.4834
Epoch 491/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - accuracy: 0.7896 - loss: 0.5209 - val_accurac
y: 0.8007 - val_loss: 0.4837
Epoch 492/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7881 - loss: 0.5211 - val_accurac
```

```
y: 0.8016 - val_loss: 0.4857
Epoch 493/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - accuracy: 0.7869 - loss: 0.5218 - val_accurac
y: 0.8043 - val_loss: 0.4813
Epoch 494/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7913 - loss: 0.5172 - val_accurac
y: 0.8047 - val_loss: 0.4786
Epoch 495/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7895 - loss: 0.5181 - val_accurac
y: 0.8022 - val_loss: 0.4817
Epoch 496/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7887 - loss: 0.5205 - val_accurac
y: 0.8008 - val_loss: 0.4819
Epoch 497/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7886 - loss: 0.5204 - val_accurac
y: 0.8039 - val_loss: 0.4824
Epoch 498/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7863 - loss: 0.5238 - val_accurac
y: 0.7969 - val_loss: 0.4954
Epoch 499/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7897 - loss: 0.5200 - val_accurac
y: 0.8037 - val_loss: 0.4823
Epoch 500/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7899 - loss: 0.5194 - val_accurac
y: 0.8042 - val_loss: 0.4804
Epoch 501/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7867 - loss: 0.5211 - val_accurac
y: 0.8030 - val_loss: 0.4812
Epoch 502/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7892 - loss: 0.5182 - val_accurac
y: 0.8026 - val_loss: 0.4865
Epoch 503/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 3s 13ms/step - accuracy: 0.7893 - loss: 0.5205 - val_accurac
y: 0.8053 - val_loss: 0.4804
Epoch 504/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - accuracy: 0.7890 - loss: 0.5212 - val_accurac
y: 0.8032 - val_loss: 0.4840
Epoch 505/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7904 - loss: 0.5190 - val_accurac
y: 0.8044 - val_loss: 0.4826
Epoch 506/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7889 - loss: 0.5186 - val_accurac
y: 0.8036 - val_loss: 0.4856
Epoch 507/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7902 - loss: 0.5178 - val_accuracy
: 0.8057 - val_loss: 0.4788
Epoch 508/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7871 - loss: 0.5203 - val_accuracy
: 0.8074 - val_loss: 0.4774
Epoch 509/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7904 - loss: 0.5189 - val_accuracy
: 0.8077 - val_loss: 0.4767
Epoch 510/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7894 - loss: 0.5197 - val_accuracy
: 0.8067 - val_loss: 0.4795
Epoch 511/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7900 - loss: 0.5211 - val_accuracy
: 0.8021 - val_loss: 0.4823
Epoch 512/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7910 - loss: 0.5167 - val_accuracy
: 0.8046 - val_loss: 0.4810
Epoch 513/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7887 - loss: 0.5221 - val_accuracy
: 0.8022 - val_loss: 0.4814
Epoch 514/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7890 - loss: 0.5199 - val_accuracy
: 0.8031 - val_loss: 0.4843
Epoch 515/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7891 - loss: 0.5178 - val_accuracy
: 0.8063 - val_loss: 0.4779
Epoch 516/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7883 - loss: 0.5182 - val_accuracy
```

```
: 0.8040 - val_loss: 0.4787
Epoch 517/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7893 - loss: 0.5193 - val_accuracy
: 0.8043 - val_loss: 0.4768
Epoch 518/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7898 - loss: 0.5177 - val_accuracy
: 0.8065 - val_loss: 0.4772
Epoch 519/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7881 - loss: 0.5204 - val_accuracy
: 0.8047 - val_loss: 0.4797
Epoch 520/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7906 - loss: 0.5153 - val_accuracy
: 0.8054 - val_loss: 0.4785
Epoch 521/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7891 - loss: 0.5194 - val_accuracy
: 0.8036 - val_loss: 0.4815
Epoch 522/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7893 - loss: 0.5169 - val_accuracy
: 0.8045 - val_loss: 0.4766
Epoch 523/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7896 - loss: 0.5189 - val_accuracy
: 0.8061 - val_loss: 0.4794
Epoch 524/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7899 - loss: 0.5172 - val_accuracy
: 0.8073 - val_loss: 0.4769
Epoch 525/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7914 - loss: 0.5175 - val_accurac
y: 0.8029 - val_loss: 0.4807
Epoch 526/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7912 - loss: 0.5162 - val_accuracy
: 0.8036 - val_loss: 0.4817
Epoch 527/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7883 - loss: 0.5212 - val_accuracy
: 0.8068 - val_loss: 0.4786
Epoch 528/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7900 - loss: 0.5158 - val_accuracy
: 0.7998 - val_loss: 0.4878
Epoch 529/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7909 - loss: 0.5147 - val_accuracy
: 0.8057 - val_loss: 0.4777
Epoch 530/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7935 - loss: 0.5128 - val_accuracy
: 0.8071 - val_loss: 0.4860
Epoch 531/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7903 - loss: 0.5153 - val_accurac
y: 0.8088 - val_loss: 0.4749
Epoch 532/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7913 - loss: 0.5132 - val_accuracy
: 0.8027 - val_loss: 0.4821
Epoch 533/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7909 - loss: 0.5167 - val_accurac
y: 0.8069 - val_loss: 0.4772
Epoch 534/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7894 - loss: 0.5173 - val_accuracy
: 0.8054 - val_loss: 0.4798
Epoch 535/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7923 - loss: 0.5137 - val_accuracy
: 0.8083 - val_loss: 0.4744
Epoch 536/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7910 - loss: 0.5168 - val_accuracy
: 0.8052 - val_loss: 0.4815
Epoch 537/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7903 - loss: 0.5176 - val_accuracy
: 0.8050 - val_loss: 0.4801
Epoch 538/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7909 - loss: 0.5149 - val_accuracy
: 0.8055 - val_loss: 0.4808
Epoch 539/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7914 - loss: 0.5161 - val_accuracy
: 0.8092 - val_loss: 0.4725
Epoch 540/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7911 - loss: 0.5134 - val_accuracy
```

```
: 0.8100 - val_loss: 0.4761
Epoch 541/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7909 - loss: 0.5152 - val_accuracy
: 0.8052 - val_loss: 0.4820
Epoch 542/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7909 - loss: 0.5133 - val_accuracy
: 0.8073 - val_loss: 0.4740
Epoch 543/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7904 - loss: 0.5154 - val_accuracy
: 0.8063 - val_loss: 0.4743
Epoch 544/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7901 - loss: 0.5185 - val_accuracy
: 0.8070 - val_loss: 0.4757
Epoch 545/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7910 - loss: 0.5137 - val_accuracy
: 0.8060 - val_loss: 0.4749
Epoch 546/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7938 - loss: 0.5115 - val_accuracy
: 0.8093 - val_loss: 0.4755
Epoch 547/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7919 - loss: 0.5134 - val_accuracy
: 0.7935 - val_loss: 0.5146
Epoch 548/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7910 - loss: 0.5149 - val_accuracy
: 0.8030 - val_loss: 0.4826
Epoch 549/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7908 - loss: 0.5155 - val_accuracy
: 0.8036 - val_loss: 0.4822
Epoch 550/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7928 - loss: 0.5146 - val_accuracy
: 0.8019 - val_loss: 0.4991
Epoch 551/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7874 - loss: 0.5188 - val_accuracy
: 0.7952 - val_loss: 0.4921
Epoch 552/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7942 - loss: 0.5095 - val_accuracy
: 0.8067 - val_loss: 0.4781
Epoch 553/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7895 - loss: 0.5183 - val_accuracy
: 0.8067 - val_loss: 0.4813
Epoch 554/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7905 - loss: 0.5134 - val_accuracy
: 0.8101 - val_loss: 0.4678
Epoch 555/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7930 - loss: 0.5145 - val_accuracy
: 0.8066 - val_loss: 0.4773
Epoch 556/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7918 - loss: 0.5135 - val_accuracy
: 0.8063 - val_loss: 0.4794
Epoch 557/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7919 - loss: 0.5102 - val_accuracy
: 0.8083 - val_loss: 0.4730
Epoch 558/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7918 - loss: 0.5143 - val_accuracy
: 0.8129 - val_loss: 0.4696
Epoch 559/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7912 - loss: 0.5107 - val_accuracy
: 0.8122 - val_loss: 0.4723
Epoch 560/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7916 - loss: 0.5139 - val_accuracy
: 0.8078 - val_loss: 0.4715
Epoch 561/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7937 - loss: 0.5097 - val_accuracy
: 0.8110 - val_loss: 0.4715
Epoch 562/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7942 - loss: 0.5091 - val_accuracy
: 0.8103 - val_loss: 0.4723
Epoch 563/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7931 - loss: 0.5093 - val_accuracy
: 0.8067 - val_loss: 0.4724
Epoch 564/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7925 - loss: 0.5103 - val_accuracy
```

```
: 0.8089 - val_loss: 0.4730
Epoch 565/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7937 - loss: 0.5073 - val_accuracy
: 0.7923 - val_loss: 0.5122
Epoch 566/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7937 - loss: 0.5086 - val_accuracy
: 0.8099 - val_loss: 0.4714
Epoch 567/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7921 - loss: 0.5150 - val_accuracy
: 0.8088 - val_loss: 0.4754
Epoch 568/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7932 - loss: 0.5081 - val_accuracy
: 0.8109 - val_loss: 0.4704
Epoch 569/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7920 - loss: 0.5137 - val_accuracy
: 0.8112 - val_loss: 0.4679
Epoch 570/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7923 - loss: 0.5088 - val_accuracy
: 0.8111 - val_loss: 0.4694
Epoch 571/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7928 - loss: 0.5082 - val_accuracy
: 0.8108 - val_loss: 0.4718
Epoch 572/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7940 - loss: 0.5097 - val_accurac
y: 0.8094 - val_loss: 0.4710
Epoch 573/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7936 - loss: 0.5110 - val_accuracy
: 0.8103 - val_loss: 0.4715
Epoch 574/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7917 - loss: 0.5137 - val_accuracy
: 0.8084 - val_loss: 0.4717
Epoch 575/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7925 - loss: 0.5112 - val_accurac
y: 0.8071 - val_loss: 0.4741
Epoch 576/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7945 - loss: 0.5100 - val_accuracy
: 0.8084 - val_loss: 0.4697
Epoch 577/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7947 - loss: 0.5081 - val_accuracy
: 0.7588 - val_loss: 0.7355
Epoch 578/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7882 - loss: 0.5232 - val_accuracy
: 0.8009 - val_loss: 0.4910
Epoch 579/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7878 - loss: 0.5229 - val_accuracy
: 0.8037 - val_loss: 0.4817
Epoch 580/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7882 - loss: 0.5209 - val_accuracy
: 0.8058 - val_loss: 0.4797
Epoch 581/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7919 - loss: 0.5155 - val_accuracy
: 0.8053 - val_loss: 0.4775
Epoch 582/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7939 - loss: 0.5133 - val_accuracy
: 0.8035 - val_loss: 0.4806
Epoch 583/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7912 - loss: 0.5167 - val_accuracy
: 0.8035 - val_loss: 0.4806
Epoch 584/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7882 - loss: 0.5213 - val_accuracy
: 0.8071 - val_loss: 0.4787
Epoch 585/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7927 - loss: 0.5123 - val_accuracy
: 0.8037 - val_loss: 0.4816
Epoch 586/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7904 - loss: 0.5166 - val_accuracy
: 0.8068 - val_loss: 0.4792
Epoch 587/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7902 - loss: 0.5153 - val_accuracy
: 0.8059 - val_loss: 0.4753
Epoch 588/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7890 - loss: 0.5165 - val_accuracy
```

```
: 0.8070 - val_loss: 0.4760
Epoch 589/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7916 - loss: 0.5134 - val_accuracy
: 0.8047 - val_loss: 0.4769
Epoch 590/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7904 - loss: 0.5140 - val_accuracy
: 0.8058 - val_loss: 0.4776
Epoch 591/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7913 - loss: 0.5157 - val_accuracy
: 0.8074 - val_loss: 0.4765
Epoch 592/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7905 - loss: 0.5153 - val_accurac
y: 0.8016 - val_loss: 0.4831
Epoch 593/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7907 - loss: 0.5160 - val_accuracy
: 0.8063 - val_loss: 0.4780
Epoch 594/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7911 - loss: 0.5147 - val_accuracy
: 0.8069 - val_loss: 0.4730
Epoch 595/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7926 - loss: 0.5136 - val_accuracy
: 0.8072 - val_loss: 0.4784
Epoch 596/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7917 - loss: 0.5150 - val_accuracy
: 0.8080 - val_loss: 0.4752
Epoch 597/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7891 - loss: 0.5173 - val_accuracy
: 0.8060 - val_loss: 0.4778
Epoch 598/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7931 - loss: 0.5132 - val_accurac
y: 0.8077 - val_loss: 0.4756
Epoch 599/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7911 - loss: 0.5156 - val_accuracy
: 0.8090 - val_loss: 0.4724
Epoch 600/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7905 - loss: 0.5150 - val_accuracy
: 0.8046 - val_loss: 0.4761
Epoch 601/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7891 - loss: 0.5171 - val_accuracy
: 0.8054 - val_loss: 0.4767
Epoch 602/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7929 - loss: 0.5142 - val_accuracy
: 0.8085 - val_loss: 0.4731
Epoch 603/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7919 - loss: 0.5142 - val_accuracy
: 0.8071 - val_loss: 0.4780
Epoch 604/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7911 - loss: 0.5148 - val_accuracy
: 0.8063 - val_loss: 0.4770
Epoch 605/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7888 - loss: 0.5168 - val_accuracy
: 0.8092 - val_loss: 0.4715
Epoch 606/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7916 - loss: 0.5159 - val_accuracy
: 0.8073 - val_loss: 0.4757
Epoch 607/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7944 - loss: 0.5098 - val_accuracy
: 0.8082 - val_loss: 0.4751
Epoch 608/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7902 - loss: 0.5164 - val_accuracy
: 0.8083 - val_loss: 0.4736
Epoch 609/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7894 - loss: 0.5169 - val_accuracy
: 0.8097 - val_loss: 0.4725
Epoch 610/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7934 - loss: 0.5098 - val_accuracy
: 0.8075 - val_loss: 0.4730
Epoch 611/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7939 - loss: 0.5113 - val_accuracy
: 0.8081 - val_loss: 0.4713
Epoch 612/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7919 - loss: 0.5161 - val_accuracy
```

```
: 0.8088 - val_loss: 0.4725
Epoch 613/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7909 - loss: 0.5127 - val_accurac
y: 0.8074 - val_loss: 0.4742
Epoch 614/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7900 - loss: 0.5144 - val_accuracy
: 0.8101 - val_loss: 0.4707
Epoch 615/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7917 - loss: 0.5125 - val_accuracy
: 0.8095 - val_loss: 0.4721
Epoch 616/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7927 - loss: 0.5127 - val_accuracy
: 0.8104 - val_loss: 0.4712
Epoch 617/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7943 - loss: 0.5125 - val_accuracy
: 0.8063 - val_loss: 0.4746
Epoch 618/900
234/234 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7918 - loss: 0.5120 - val_accuracy
: 0.8079 - val_loss: 0.4729
```

Out[105]:

```
<keras.src.callbacks.history.History at 0x2b2d705a3c0>
```

In [106]:

```
model.evaluate(X_train, y_train)
```

```
3732/3732 ━━━━━━━━━━━━━━━━━━━━ 3s 864us/step - accuracy: 0.8373 - loss: 0.4099
```

Out[106]:

```
[0.40938422083854675, 0.8377835154533386]
```

In [107]:

```
eval_metric(model, X_train, y_train, X_test, y_test)
```

```
3732/3732 ━━━━━━━━━━━━━━━━━━━━ 3s 891us/step
659/659 ━━━━━━━━━━━━━━━━━━━━ 1s 845us/step
Test Set:
[[6483   72  468]
 [ 266 6007  750]
 [1061 1326 4637]]
              precision    recall  f1-score   support

           0       0.83      0.92      0.87      7023
           1       0.81      0.86      0.83      7023
           2       0.79      0.66      0.72      7024

    accuracy                           0.81     21070
   macro avg       0.81      0.81      0.81     21070
weighted avg       0.81      0.81      0.81     21070


Train Set:
[[37366   253  2180]
 [ 1060 35023  3716]
 [ 5281  6878 27639]]
              precision    recall  f1-score   support

           0       0.85      0.94      0.89     39799
           1       0.83      0.88      0.85     39799
           2       0.82      0.69      0.75     39798

    accuracy                           0.84    119396
   macro avg       0.84      0.84      0.83    119396
weighted avg       0.84      0.84      0.83    119396
```

In [108]:

```
sample_input = pd.DataFrame({
```

```
    'Age': [23],
    'Occupation': ['Scientist'],
    'Annual_Income': [19114.12],
    'Monthly_Inhand_Salary': [1824.8433333333328],
    'Num_Bank_Accounts': [3],
    'Num_Credit_Card': [4],
    'Interest_Rate': [3.0],
    'Num_of_Loan': [4.0],
    'Delay_from_due_date': [3],
    'Num_of_Delayed_Payment': [7.0],
    'Changed_Credit_Limit': [11.27],
    'Num_Credit_Inquiries': [4.0],
    'Credit_Mix': ['Good'],
    'Outstanding_Debt': [809.98],
    'Credit_Utilization_Ratio': [26.822619623699016],
    'Payment_of_Min_Amount': ['No'],
    'Total_EMI_per_month': [49.57494921489417],
    'Amount_invested_monthly': [118.28022162236736],
    'Payment_Behaviour': ['High_spent_Small_value_payments'],
    'Monthly_Balance': [312.49408867943663],
    'Credit_History_Age_Months': [265]
})

# Apply encodings
sample_input['Occupation'] = label_encoder3.transform(sample_input['Occupation'])
sample_input['Payment_of_Min_Amount'] = label_encoder2.transform(sample_input['Payment_of
_Min_Amount'])
sample_input['Payment_Behaviour'] = payment_behaviour_encoder.fit_transform(sample_input[
['Payment_Behaviour']])
sample_input['Credit_Mix'] = label_encoder1.transform(sample_input[['Credit_Mix']])

# Apply scaling (ensure that the same scaler used during training is used here)
scaled_input = scaler.transform(sample_input)

# Make prediction
predicted_class = model.predict(scaled_input)
predicted_class_label = np.argmax(predicted_class, axis=1)  # Assuming it's a classifica
tion model with softmax

print(f"The predicted class is: {predicted_class_label[0]}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
The predicted class is: 0
```

In [109]:

```
def predict_credit_score(Age, Occupation, 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, Credit_Mix, Outstan
ding_Debt,
                         Credit_Utilization_Ratio, Payment_of_Min_Amount, Total_EMI_per_
month,
                         Amount_invested_monthly, Payment_Behaviour, Monthly_Balance, Cr
edit_History_Age_Months):

    # Create the input DataFrame
    sample_input = pd.DataFrame({
        'Age': [Age],
        'Occupation': [Occupation],
        'Annual_Income': [Annual_Income],
        'Monthly_Inhand_Salary': [Monthly_Inhand_Salary],
        'Num_Bank_Accounts': [Num_Bank_Accounts],
        'Num_Credit_Card': [Num_Credit_Card],
        'Interest_Rate': [Interest_Rate],
        'Num_of_Loan': [Num_of_Loan],
        'Delay_from_due_date': [Delay_from_due_date],
        'Num_of_Delayed_Payment': [Num_of_Delayed_Payment],
        'Changed_Credit_Limit': [Changed_Credit_Limit],
        'Num_Credit_Inquiries': [Num_Credit_Inquiries],
        'Credit_Mix': [Credit_Mix],
```

```
        'Outstanding_Debt': [Outstanding_Debt],
        'Credit_Utilization_Ratio': [Credit_Utilization_Ratio],
        'Payment_of_Min_Amount': [Payment_of_Min_Amount],
        'Total_EMI_per_month': [Total_EMI_per_month],
        'Amount_invested_monthly': [Amount_invested_monthly],
        'Payment_Behaviour': [Payment_Behaviour],
        'Monthly_Balance': [Monthly_Balance],
        'Credit_History_Age_Months': [Credit_History_Age_Months]
    })

    # Apply encodings
    sample_input['Occupation'] = label_encoder3.transform(sample_input['Occupation'])
    sample_input['Payment_of_Min_Amount'] = label_encoder2.transform(sample_input['Payme
nt_of_Min_Amount'])
    sample_input['Payment_Behaviour'] = payment_behaviour_encoder.transform(sample_input
[['Payment_Behaviour']])
    sample_input['Credit_Mix'] = label_encoder1.transform(sample_input[['Credit_Mix']])

    # Apply scaling (ensure that the same scaler used during training is used here)
    scaled_input = scaler.transform(sample_input)

    # Make prediction
    predicted_class = model.predict(scaled_input)
    predicted_class_label = np.argmax(predicted_class, axis=1)[0]

    # Map numeric labels to credit score classes
    credit_score_map = {0: "Credit score GOOD", 1: "Credit score POOR", 2: "Credit score
STANDARD"}
    return credit_score_map.get(predicted_class_label, "Unknown credit score")
```

In [ ]:

```
import gradio as gr
interface = gr.Interface(
    fn=predict_credit_score,
    inputs=[
        gr.Number(label="Age"),
        gr.Dropdown(choices=['Lawyer', 'Mechanic', 'Architect', 'Engineer', 'Accountant'
,
                                      'Scientist', 'Developer', 'Teacher', 'Doctor', 'Medi
aManager',
                                      'Journalist', 'Entrepreneur', 'Musician', 'Manager',
'Writer'], label="Occupation"),
        gr.Number(label="Annual Income"),
        gr.Number(label="Monthly Inhand Salary"),
        gr.Number(label="Num Bank Accounts"),
        gr.Number(label="Num Credit Card"),
        gr.Number(label="Interest Rate"),
        gr.Number(label="Num of Loan"),
        gr.Number(label="Delay from due date"),
        gr.Number(label="Num of Delayed Payment"),
        gr.Number(label="Changed Credit Limit"),
        gr.Number(label="Num Credit Inquiries"),
        gr.Dropdown(choices=['Bad', 'Good', 'Standard'], label="Credit Mix"),
        gr.Number(label="Outstanding Debt"),
        gr.Number(label="Credit Utilization Ratio"),
        gr.Dropdown(choices=['Yes', 'No', 'NM'], label="Payment of Min Amount"),
        gr.Number(label="Total EMI per month"),
        gr.Number(label="Amount invested monthly"),
        gr.Dropdown(choices=['Low_spent_Small_value_payments', 'Low_spent_Medium_value_p
ayments',
                                      'Low_spent_Large_value_payments', 'High_spent_Small_
value_payments',
                                      'High_spent_Medium_value_payments', 'High_spent_Larg
e_value_payments'],
                           label="Payment Behaviour"),
        gr.Number(label="Monthly Balance"),
        gr.Number(label="Credit History Age in Months")
    ],
    outputs="text",
    title="Credit Score Prediction",
```

```
    description="Enter customer details to predict their credit score class."
)

# Launch the Gradio interface
interface.launch()
```