Online news popularity (CEU-ML 2024)

Viet H. Nguyen

Introduction

This notebook is part of a Kaggle competition to predict wether online news article is popular. MOre details can be found here. In this notebook, I will guide you through all of my modeling experiments and discuss how and why I choose the model for submission!

In [1]:
```
%%capture
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np

import seaborn as sns
from matplotlib import pyplot as plt
```

In [2]:
```
news_df = pd.read_csv('online-news-popularity-ceu-ml-2024/train.csv')
news_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29733 entries, 0 to 29732
Data columns (total 61 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   timedelta                      29733 non-null  int64  
 1   n_tokens_title                 29733 non-null  int64  
 2   n_tokens_content               29733 non-null  int64  
 3   n_unique_tokens                29733 non-null  float64
 4   n_non_stop_words               29733 non-null  float64
 5   n_non_stop_unique_tokens       29733 non-null  float64
 6   num_hrefs                      29733 non-null  int64  
 7   num_self_hrefs                 29733 non-null  int64  
 8   num_imgs                       29733 non-null  int64  
 9   num_videos                     29733 non-null  int64  
 10  average_token_length           29733 non-null  float64
 11  num_keywords                   29733 non-null  int64  
 12  data_channel_is_lifestyle      29733 non-null  int64  
 13  data_channel_is_entertainment  29733 non-null  int64  
 14  data_channel_is_bus            29733 non-null  int64  
 15  data_channel_is_socmed         29733 non-null  int64  
 16  data_channel_is_tech           29733 non-null  int64  
 17  data_channel_is_world          29733 non-null  int64  
 18  kw_min_min                     29733 non-null  int64  
 19  kw_max_min                     29733 non-null  float64
 20  kw_avg_min                     29733 non-null  float64
 21  kw_min_max                     29733 non-null  int64  
 22  kw_max_max                     29733 non-null  int64  
 23  kw_avg_max                     29733 non-null  float64
 24  kw_min_avg                     29733 non-null  float64
 25  kw_max_avg                     29733 non-null  float64
 26  kw_avg_avg                     29733 non-null  float64
 27  self_reference_min_shares      29733 non-null  float64
 28  self_reference_max_shares      29733 non-null  float64
 29  self_reference_avg_sharess     29733 non-null  float64
 30  weekday_is_monday              29733 non-null  int64  
 31  weekday_is_tuesday             29733 non-null  int64  
 32  weekday_is_wednesday           29733 non-null  int64  
 33  weekday_is_thursday            29733 non-null  int64  
 34  weekday_is_friday              29733 non-null  int64  
 35  weekday_is_saturday            29733 non-null  int64  
 36  weekday_is_sunday              29733 non-null  int64  
 37  is_weekend                     29733 non-null  int64  
 38  LDA_00                         29733 non-null  float64
 39  LDA_01                         29733 non-null  float64
 40  LDA_02                         29733 non-null  float64
 41  LDA_03                         29733 non-null  float64
 42  LDA_04                         29733 non-null  float64
 43  global_subjectivity            29733 non-null  float64
 44  global_sentiment_polarity      29733 non-null  float64
 45  global_rate_positive_words     29733 non-null  float64
 46  global_rate_negative_words     29733 non-null  float64
 47  rate_positive_words            29733 non-null  float64
 48  rate_negative_words            29733 non-null  float64
 49  avg_positive_polarity          29733 non-null  float64
 50  min_positive_polarity          29733 non-null  float64
 51  max_positive_polarity          29733 non-null  float64
 52  avg_negative_polarity          29733 non-null  float64
 53  min_negative_polarity          29733 non-null  float64
 54  max_negative_polarity          29733 non-null  float64
 55  title_subjectivity             29733 non-null  float64
 56  title_sentiment_polarity       29733 non-null  float64
 57  abs_title_subjectivity         29733 non-null  float64
 58  abs_title_sentiment_polarity   29733 non-null  float64
 59  is_popular                     29733 non-null  int64  
 60  article_id                     29733 non-null  int64  
dtypes: float64(34), int64(27)
memory usage: 13.8 MB
```

In [3]:
```
news_df.head(10)
```

Out[3]:

	timedelta	n_tokens_title	n_tokens_content	n_unique_tokens	n_non_stop_words	n_non_stop_unique_tokens	num_hr
0	594	9	702	0.454545	1.0	0.620438	
1	346	8	1197	0.470143	1.0	0.666209	
2	484	9	214	0.618090	1.0	0.748092	
3	639	8	249	0.621951	1.0	0.664740	
4	177	12	1219	0.397841	1.0	0.583578	
5	568	7	126	0.723577	1.0	0.774194	
6	318	12	1422	0.367994	1.0	0.469256	
7	582	6	1102	0.451287	1.0	0.642089	
8	269	9	0	0.000000	0.0	0.000000	
9	567	7	94	0.755319	1.0	0.812500	

10 rows × 61 columns

Data Cleaning

Upon investigation, there's no data cleaning needed since the dataset has no missing values and the values seem to make sense.

EDA

Checking the correlation between all the existing variables, I noticed that some of them have very high correlations. To reduce the redundancies, I removed the highly correlated variables before splitting the data into training and test set.

In [4]:
```
plot_data = news_df.drop(columns=['timedelta', 'is_popular', 'article_id']).corr()
test = plot_data.applymap(lambda x: 1 if x >= 0.8 else -1 if x <= -0.8 else 0)

high_correlation_pairs = []
for row_index, row in test.iterrows():
    for column_name, cell_value in row.items():
        if (cell_value == 1 or cell_value == -1) and row_index != column_name and (row_index + '*' + column
            high_correlation_pairs.append(row_index + '*' + column_name)
high_correlation_pairs = [(x.split('*')[0], x.split('*')[1]) for x in high_correlation_pairs]
high_correlation_pairs
```

Out[4]:
```
[('n_unique_tokens', 'n_non_stop_words'),
 ('n_unique_tokens', 'n_non_stop_unique_tokens'),
 ('n_non_stop_words', 'n_non_stop_unique_tokens'),
 ('data_channel_is_world', 'LDA_02'),
 ('kw_min_min', 'kw_max_max'),
 ('kw_max_min', 'kw_avg_min'),
 ('kw_max_avg', 'kw_avg_avg'),
 ('self_reference_min_shares', 'self_reference_avg_sharess'),
 ('self_reference_max_shares', 'self_reference_avg_sharess')]
```

In [5]:
```
from sklearn.model_selection import train_test_split

exclude_cols = ['timedelta', 'is_popular', 'article_id', 'kw_min_min', 'kw_max_min', 'kw_max_avg', 'n_non_s
binary_cols = [col for col in news_df.columns if col.startswith('weekday_is_')] + [col for col in news_df.c

# split data to train & val & test
outcome = news_df["is_popular"]
features = news_df.drop(columns=exclude_cols)
# features = news_df.drop(columns=['timedelta', 'is_popular', 'article_id'])
prng = np.random.RandomState(42)
X_train, X_test, y_train, y_test = train_test_split(features, outcome, test_size=0.1, random_state=prng)
# X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.1, random_state=prng)
X_val, y_val = X_test, y_test

print(X_train.shape, X_val.shape, X_test.shape)
```

```
(26759, 39) (2974, 39) (2974, 39)
```

Modeling

Simple Logit

This simple logit model only use the raw variables that is in the dataset to predict whether the article is popular.

In [6]:
```
from sklearn.linear_model import LogisticRegressionCV
from sklearn.metrics import accuracy_score, roc_auc_score, f1_score

# no regularisation needed so setting the parameter to very high value
Cs_value_logit = [1e20]
scoring='roc_auc'

logit_model = LogisticRegressionCV(
        Cs=Cs_value_logit,
        refit=True,
        scoring=scoring,
        solver="liblinear",
        tol=1e-7,
        random_state=prng
    )

logit_model.fit(X_train, y_train)

summary_df = pd.DataFrame({'Model': ['Logit as Benchmark'],
                          'Train AUC': [round(roc_auc_score(y_train, logit_model.predict_proba(X_train)[:,
                          'Val AUC': [round(roc_auc_score(y_val, logit_model.predict_proba(X_val)[:,1]), 4
                          'Test AUC': [round(roc_auc_score(y_test, logit_model.predict_proba(X_test)[:,1])
                          # 'Train accuracy': [round(accuracy_score(y_train, logit_model.predict(X_train))
                          # 'Val accuracy': [round(accuracy_score(y_val, logit_model.predict(X_val)), 4)],
                          # 'Test accuracy': [round(accuracy_score(y_test, logit_model.predict(X_test)), 4
                          # 'Train F1 score': [round(f1_score(y_train, logit_model.predict(X_train)), 4)],
                          # 'Val F1 score': [round(f1_score(y_val, logit_model.predict(X_val)), 4)],
                          # 'Test F1 score': [round(f1_score(y_test, logit_model.predict(X_test)), 4)],
                         })
summary_df
```

Out[6]:

	Model	Train AUC	Val AUC	Test AUC
0	Logit as Benchmark	0.6707	0.6637	0.6637

The AUC score for the logit model will be the benchmark for the subsequent models that I experiment with.

In the summary table, there's a column for the validation set AUC score. For non-neural network model, this column is the same as the test AUC score since I will only split once for the training and test set. For the neural network, there is another split of the training set to obtain the validation set.

Feature Engineering

I also experiment with feature engineering via creating some ratio variables like:

- Keyword density
- Links ratio
- Media ratio
- Sentiment balance
- Emotional intensity

After engineering more features, I test for correlation again among all the variables and remove any that has high correlation. I then split the data again afterward, reseting the reandom seed to obtain the same splits.

In [7]:
```
def feature_engineer(df):
    # normalized unique tokens and keywords
    # df['e_unique_tokens_normalized'] = df.apply(lambda x: x['n_unique_tokens'] / x['n_tokens_content'] if
    # df['e_non_stop_unique_tokens_normalized'] = df.apply(lambda x: x['n_non_stop_unique_tokens'] / x['n_t
    df['e_keyword_density'] = df.apply(lambda x: x['num_keywords'] / x['n_tokens_content'] if x['n_tokens_c
    df['e_title_length_ratio'] = df.apply(lambda x: x['n_tokens_title'] / x['n_tokens_content'] if x['n_tok

    # links ratio
    df['e_external_link_ratio'] = df.apply(lambda x: x['num_hrefs'] / x['n_tokens_content'] if x['n_tokens_
    df['e_self_reference_link_ratio'] = df.apply(lambda x: x['num_self_hrefs'] / x['num_hrefs'] if x['num_h

    # media ratio
    df['e_multimedia_content_ratio'] = df.apply(lambda x: (x['num_imgs'] + x['num_videos']) / x['n_tokens_c
```

```python
    # composite indicators of sentiment balance or emotional intensity
    df['e_sentiment_balance'] = df['global_rate_positive_words'] - df['global_rate_negative_words']
    df['e_emotional_intensity'] = df['global_sentiment_polarity'] * df['global_subjectivity']

    # count of channels associated with each article
    channel_cols = [col for col in news_df.columns if col.startswith('data_channel_is')]
    df['e_num_channels'] = df[channel_cols].sum(axis=1)
    df['e_is_multi_channel'] = df['e_num_channels'].apply(lambda x: 1 if x > 1 else 0)

    binary_cols = ['is_weekend', 'e_is_multi_channel']

    return df, binary_cols

news_df, binary_cols = feature_engineer(news_df)
```

In [8]:
```python
plot_data = news_df.drop(columns=['timedelta', 'is_popular', 'article_id']).corr()
test = plot_data.applymap(lambda x: 1 if x >= 0.8 else -1 if x <= -0.8 else 0)

high_correlation_pairs = []
for row_index, row in test.iterrows():
    for column_name, cell_value in row.items():
        if (cell_value == 1 or cell_value == -1) and row_index != column_name and (row_index + '*' + column
            high_correlation_pairs.append(row_index + '*' + column_name)
high_correlation_pairs = [(x.split('*')[0], x.split('*')[1]) for x in high_correlation_pairs]
high_correlation_pairs
```

Out[8]:
```
[('n_unique_tokens', 'n_non_stop_words'),
 ('n_unique_tokens', 'n_non_stop_unique_tokens'),
 ('n_non_stop_words', 'n_non_stop_unique_tokens'),
 ('average_token_length', 'e_keyword_density'),
 ('average_token_length', 'e_title_length_ratio'),
 ('data_channel_is_world', 'LDA_02'),
 ('kw_min_min', 'kw_max_max'),
 ('kw_max_min', 'kw_avg_min'),
 ('kw_max_avg', 'kw_avg_avg'),
 ('self_reference_min_shares', 'self_reference_avg_sharess'),
 ('self_reference_max_shares', 'self_reference_avg_sharess'),
 ('global_sentiment_polarity', 'e_emotional_intensity'),
 ('global_rate_positive_words', 'e_sentiment_balance'),
 ('e_keyword_density', 'e_title_length_ratio')]
```

In [9]:
```python
# exclude_cols_tmp = ['n_non_stop_words','kw_min_min','kw_max_min','kw_max_avg','self_reference_min_shares'
exclude_cols_tmp = ['average_token_length','global_sentiment_polarity','global_rate_negative_words', 'e_tit
[exclude_cols.append(x) for x in exclude_cols_tmp if x not in exclude_cols]
exclude_cols
```

Out[9]:
```
['timedelta',
 'is_popular',
 'article_id',
 'kw_min_min',
 'kw_max_min',
 'kw_max_avg',
 'n_non_stop_unique_tokens',
 'self_reference_min_shares',
 'self_reference_max_shares',
 'weekday_is_monday',
 'weekday_is_tuesday',
 'weekday_is_wednesday',
 'weekday_is_thursday',
 'weekday_is_friday',
 'weekday_is_saturday',
 'weekday_is_sunday',
 'data_channel_is_lifestyle',
 'data_channel_is_entertainment',
 'data_channel_is_bus',
 'data_channel_is_socmed',
 'data_channel_is_tech',
 'data_channel_is_world',
 'average_token_length',
 'global_sentiment_polarity',
 'global_rate_negative_words',
 'e_title_length_ratio']
```

In [10]:
```python
# split train, val, test again with engineered features
outcome = news_df["is_popular"]
features = news_df.drop(columns=exclude_cols)
# features = news_df.drop(columns=['timedelta', 'is_popular', 'article_id'])
prng = np.random.RandomState(42)
```

```
X_train, X_test, y_train, y_test = train_test_split(features, outcome, test_size=0.1, random_state=prng)
# X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.1, random_state=prng)
X_val, y_val = X_test, y_test

print(X_train.shape, X_val.shape, X_test.shape, y_train.shape, y_val.shape, y_test.shape)
```

```
(26759, 44) (2974, 44) (2974, 44) (26759,) (2974,) (2974,)
```

In [11]:
```
def update_summary(df, model_name, y_train_true, y_train_pred, y_val_true, y_val_pred, y_test_true, y_test_
    if class1_only:
        if model_name not in df.Model.values:
            df.loc[len(df.index)] = [model_name,
                                     '{:.4f}'.format(roc_auc_score(y_train_true, y_train_pred)),
                                     '{:.4f}'.format(roc_auc_score(y_val_true, y_val_pred)),
                                     '{:.4f}'.format(roc_auc_score(y_test_true, y_test_pred)),]
        else:
            df.loc[df.Model == model_name] = [model_name,
                                             '{:.4f}'.format(roc_auc_score(y_train_true, y_train_pred)),
                                             '{:.4f}'.format(roc_auc_score(y_val_true, y_val_pred)),
                                             '{:.4f}'.format(roc_auc_score(y_test_true, y_test_pred)),]
    else:
        if model_name not in df.Model.values:
            df.loc[len(df.index)] = [model_name,
                                     '{:.4f}'.format(roc_auc_score(y_train_true, y_train_pred[:,1])),
                                     '{:.4f}'.format(roc_auc_score(y_val_true, y_val_pred[:,1])),
                                     '{:.4f}'.format(roc_auc_score(y_test_true, y_test_pred[:,1])),]
                                     # '{:.4f}'.format(accuracy_score(y_train_true, y_train_pred)),
                                     # '{:.4f}'.format(accuracy_score(y_val_true, y_val_pred)),
                                     # '{:.4f}'.format(accuracy_score(y_test_true, y_test_pred)),
                                     # '{:.4f}'.format(f1_score(y_train_true, y_train_pred)),
                                     # '{:.4f}'.format(f1_score(y_val_true, y_val_pred)),
                                     # '{:.4f}'.format(f1_score(y_test_true, y_test_pred))]
        else:
            df.loc[df.Model == model_name] = [model_name,
                                             '{:.4f}'.format(roc_auc_score(y_train_true, y_train_pred[:,1])
                                             '{:.4f}'.format(roc_auc_score(y_val_true, y_val_pred[:,1])),
                                             '{:.4f}'.format(roc_auc_score(y_test_true, y_test_pred[:,1])),
                                             # '{:.4f}'.format(accuracy_score(y_train_true, y_train_pred)),
                                             # '{:.4f}'.format(accuracy_score(y_val_true, y_val_pred)),
                                             # '{:.4f}'.format(accuracy_score(y_test_true, y_test_pred)),
                                             # '{:.4f}'.format(f1_score(y_train_true, y_train_pred)),
                                             # '{:.4f}'.format(f1_score(y_val_true, y_val_pred)),
                                             # '{:.4f}'.format(f1_score(y_test_true, y_test_pred))]
```

Modeling with Feature Engineering

LASSO Logit

After doing feature engineering, I combine it with a LASSO logit model to regularize the impact by adding more features into the model.

In [12]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.pipeline import Pipeline

lambdas = list(10**np.arange(-1, -3.01, -1/3))
n_obs = len(X_train)
Cs_values = [1/(l*n_obs) for l in lambdas]

lasso_search = LogisticRegressionCV(
    Cs = Cs_values,
    penalty = 'l1', # L1 makes it lasso
    cv = 5,
    refit = True,
    scoring = scoring,
    solver = 'liblinear',
    random_state = prng,
    # verbose=True
)

preprocessor = ColumnTransformer(
    transformers=[
        ('scale', StandardScaler(), features.drop(columns=binary_cols).columns),
        ('leave_out', 'passthrough', binary_cols)  # Leave out columns without transformation
    ],
    remainder='passthrough'  # Drop columns not specified in transformers
```

```
)

lasso_model = Pipeline(
     [('preprocessor', preprocessor),
     ("regressor", lasso_search)
     ], verbose=True
)

lasso_model.fit(X_train, y_train)

update_summary(summary_df,
               'LASSO Logit',
               y_train,
               lasso_model.predict_proba(X_train),
               y_val,
               lasso_model.predict_proba(X_val),
               y_test,
               lasso_model.predict_proba(X_test))
summary_df
```

```
[Pipeline] ...... (step 1 of 2) Processing preprocessor, total=   0.0s
[Pipeline] ......... (step 2 of 2) Processing regressor, total=   3.9s
```

Out[12]:

	Model	Train AUC	Val AUC	Test AUC
0	Logit as Benchmark	0.6707	0.6637	0.6637
1	LASSO Logit	0.6835	0.6791	0.6791

From the summary table, the LASSO logit's AUC score in both the training and test set improves compared to the benchmark. This gives me some confidence that the feature engineering does provide more information for the model to capture the data better.

Random Forest

I experiment with ensemble methods starting with a random forest model. In the code, there are some optimizations that I have done to optimize this model:

- Cross validation to obtain the best hyperparameters (the hyperparameters grid is commented out and replaced with the best parameters to cut down on the rerun time)
- Perform permutation importance analysis to find the most impactful predictors
- Refit the model after removing unimportant predictors from the available variables

In [13]:

```
rf_high_perm = ['kw_avg_avg',
 'self_reference_avg_sharess',
 'kw_min_avg',
 'LDA_03',
 'e_multimedia_content_ratio',
 'e_num_channels',
 'num_hrefs',
 'num_imgs',
 'LDA_04',
 'kw_min_max',
 'LDA_02',
 'e_self_reference_link_ratio',
 'n_unique_tokens',
 'num_videos',
 'global_subjectivity',
 'n_tokens_title',
 'e_external_link_ratio',
 'n_tokens_content',
 'avg_negative_polarity',
 'title_sentiment_polarity',
 'e_sentiment_balance',
 'abs_title_sentiment_polarity',
 'e_emotional_intensity',
 'global_rate_positive_words',
 'is_weekend',
 'kw_max_max',
 'kw_avg_max',
 'num_self_hrefs',
 'title_subjectivity',
 'rate_positive_words',
 'rate_negative_words',
 'max_negative_polarity']
```

In [14]:
```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline

# max_depth = [int(x) for x in np.linspace(1, 100, num = 4)]
max_depth = [15, 18, 20, 25]
max_depth.append(None)

# grid = {'max_features': [0.3, 0.5, 1],
#         'criterion':['gini'],
#         'max_depth': max_depth,
#         'min_samples_split': [5, 10, 15],
#         "min_samples_leaf": [2, 4, 8]
#         }

# grid =  {'criterion': ['gini'],
#  'max_depth': [50],
#  'max_features': [1],
#  'min_samples_leaf': [4],
#  'min_samples_split': [10]}

grid = {'criterion': ['gini'],
        'max_depth': [15],
        'max_features': [0.3],
        'min_samples_leaf': [8],
        'min_samples_split': [5]}

prob_forest_search = GridSearchCV(
    RandomForestClassifier(random_state = prng, oob_score=True, n_estimators=500, bootstrap=True),
    grid,
    cv=5,
    refit='roc_auc',
    scoring = ['roc_auc'],
    verbose=True,
    # random_state=prng,
    n_jobs=-1)

rf_model = Pipeline(
    [("rf", prob_forest_search)
     ], verbose=True
)

rf_model.fit(X_train[rf_high_perm], y_train)
predictions_rf = rf_model.predict(X_val[rf_high_perm])
accuracy_score(y_val, predictions_rf)
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
[Pipeline] ................ (step 1 of 1) Processing rf, total= 2.0min
```

Out[14]: 0.8910558170813719

In [15]:
```
prob_forest_search.best_params_
```

Out[15]:
```
{'criterion': 'gini',
 'max_depth': 15,
 'max_features': 0.3,
 'min_samples_leaf': 8,
 'min_samples_split': 5}
```

In [16]:
```
update_summary(summary_df,
               'Random Forest CV',
               y_train,
               rf_model.predict_proba(X_train[rf_high_perm]),
               y_val,
               rf_model.predict_proba(X_val[rf_high_perm]),
               y_test,
               rf_model.predict_proba(X_test[rf_high_perm]))
summary_df
```

Out[16]:

	Model	Train AUC	Val AUC	Test AUC
0	Logit as Benchmark	0.6707	0.6637	0.6637
1	LASSO Logit	0.6835	0.6791	0.6791
2	Random Forest CV	0.9686	0.7086	0.7086

There is a significant jump in the AUC score using the optimized random forest model by almost 3%. It seems that the ensemble method is better at fitting the data and capture more patterns compared to the other models, possibly because the random forest is able to pick up the non-linear and interaction relationship among the predictors.

In [17]:
```
# from sklearn.inspection import permutation_importance
#
# rf_imp = permutation_importance(
#     rf_model,
#     X_test[rf_high_perm],
#     y_test,
#     n_repeats=10,
#     random_state=prng,
#     # scoring="neg_root_mean_squared_error"
# )
#
# grouped_var_imp = (pd.DataFrame(
#         rf_imp.importances_mean,
#         features.columns)
#                      .sort_values(by = 0, ascending = False)
#                      .reset_index()
#                      .rename(columns={'index': 'variable', 0: 'imp'}))
# grouped_var_imp['cumulative_imp'] = grouped_var_imp.imp.cumsum()
#
# rf_fig = sns.barplot(
#     data = grouped_var_imp,
#     x="imp", y="variable")
# rf_fig.set(title='Random forest model grouped feature importances', xlabel="importance", ylabel="variable
# plt.show()
```

In [18]:
```
# grouped_var_imp[(grouped_var_imp['imp'] >= 0.0001)]['variable'].tolist()
```

GBM

I also experiment with the gradient boosting model (specifically a variant of it using sklearn HistGradientBoostingClassifier as it runs faster than the traditional GradientBoostingClassifier). I also run the optimization similarly to that of the random forest model. Again, the tuning grid is raplaced by the optimized grid for re-running purpose.

In [19]:
```
gbm_high_perm = ['kw_avg_avg',
 'self_reference_avg_sharess',
 'kw_min_avg',
 'e_self_reference_link_ratio',
 'kw_min_max',
 'kw_avg_max',
 'num_videos',
 'LDA_02',
 'num_imgs',
 'num_hrefs',
 'e_num_channels',
 'LDA_04',
 'n_tokens_content',
 'is_weekend',
 'global_subjectivity',
 'num_self_hrefs',
 'n_non_stop_words',
 'kw_max_max',
 'n_tokens_title',
 'title_subjectivity',
 'e_emotional_intensity',
 'e_multimedia_content_ratio',
 'abs_title_subjectivity',
 'kw_avg_min',
 'abs_title_sentiment_polarity',
 'LDA_03',
 'title_sentiment_polarity',
 'avg_negative_polarity',
 'e_external_link_ratio',
 'e_sentiment_balance',
 'max_positive_polarity',
 'LDA_01']
```

In [20]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import HistGradientBoostingClassifier

# max_depth = [int(x) for x in np.linspace(1, 100, num = 5)]
```

```
max_depth = [15, 18, 20, 25]
max_depth.append(None)

# gbm_grid = {'max_features': [0.1, 0.15],
#          'max_depth': max_depth,
#          "min_samples_leaf": [15, 18, 20],
#          'l2_regularization': [0.05, 0.08, 0.1],
#          'class_weight': [None],
#          'max_iter': [500],
#          'learning_rate': [0.01, 0.001]
#          }

gbm_grid =  {'class_weight': [None],
         'l2_regularization': [0.05],
         'learning_rate': [0.01],
         'max_depth': [20],
         'max_features': [0.1],
         'max_iter': [500],
         'min_samples_leaf': [15]}

gbm_search = GridSearchCV(
    HistGradientBoostingClassifier(random_state = prng),
    gbm_grid,
    cv=5,
    refit='roc_auc',
    scoring = ['roc_auc'],
    verbose=True,
    # random_state=prng,
    n_jobs=-1)

# RF as benchmark
gbm_model = Pipeline(
    [("gbm", gbm_search)
     ], verbose=True
)

gbm_model.fit(X_train[gbm_high_perm], y_train)
predictions_gbm = gbm_model.predict(X_val[gbm_high_perm])
accuracy_score(y_val, predictions_gbm)
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
[Pipeline] ............... (step 1 of 1) Processing gbm, total=  12.6s
```

Out[20]: 0.8910558170813719

In [21]:
```
gbm_search.best_params_
```

Out[21]:
```
{'class_weight': None,
 'l2_regularization': 0.05,
 'learning_rate': 0.01,
 'max_depth': 20,
 'max_features': 0.1,
 'max_iter': 500,
 'min_samples_leaf': 15}
```

In [22]:
```
update_summary(summary_df,
               'GBM CV',
               y_train,
               gbm_model.predict_proba(X_train[gbm_high_perm]),
               y_val,
               gbm_model.predict_proba(X_val[gbm_high_perm]),
               y_test,
               gbm_model.predict_proba(X_test[gbm_high_perm]))
summary_df
```

Out[22]:

	Model	Train AUC	Val AUC	Test AUC
0	Logit as Benchmark	0.6707	0.6637	0.6637
1	LASSO Logit	0.6835	0.6791	0.6791
2	Random Forest CV	0.9686	0.7086	0.7086
3	GBM CV	0.8193	0.7133	0.7133

From the summary table, the gradient boosting is the best model so far in terms of the AUC score for the unseen data. The lower AUC score in the training set compared to the random forest suggest that there might be some training overfitting happen with the random forest. The prediction submission on Kaggle also confirms this hypothesis as the score for the gradient boosting model is the highest.

In [23]:
```
# from sklearn.inspection import permutation_importance
#
# gbm_imp = permutation_importance(
#     gbm_model,
#     X_test,
#     y_test,
#     n_repeats=10,
#     random_state=prng,
#     # scoring="neg_root_mean_squared_error"
# )
#
# grouped_var_imp = (pd.DataFrame(
#         gbm_imp.importances_mean,
#         features.columns)
#                       .sort_values(by = 0, ascending = False)
#                       .reset_index()
#                       .rename(columns={'index': 'variable', 0: 'imp'}))
# grouped_var_imp['cumulative_imp'] = grouped_var_imp.imp.cumsum()
#
# gbm_fig = sns.barplot(
#     data = grouped_var_imp,
#     x="imp", y="variable")
# gbm_fig.set(title='GBM model grouped feature importances', xlabel="importance", ylabel="variable")
# plt.show()
```

In [24]:
```
# grouped_var_imp[(grouped_var_imp['imp'] >= 0.0001)]['variable'].tolist();
```

In [25]:
```
# save original X, y data
X_ori_sets = [X_train.copy(), X_val.copy(), X_test.copy()]
y_ori_sets = [y_train.copy(), y_val.copy(), y_test.copy()]

print(X_ori_sets[0].shape, X_ori_sets[1].shape, X_ori_sets[2].shape, y_ori_sets[0].shape, y_ori_sets[1].sha
```

```
(26759, 44) (2974, 44) (2974, 44) (26759,) (2974,) (2974,)
```

NN with Sigmoid activation

Starting from here, all the subsequent models are neural network or include the neural network as part of the model. To train these models, I split the training set to obtain the validation set with the 9:1 ratio.

The first neural network model consists of 1 hidden layer and an output layer with 1 neuron and use the sigmoid function as activation. The output thus can be directly interpreted as the probability for whether an article is popular.

I have been optimizing the network to obtain the best AUC score by:

- Increase the number of layers
- Increase the number of neurons
- Try different mini batch size
- Introduce the kernel_regularizer with L1 regularization with different strength
- Add dropout layer with different ratios
- Introduce the kernel_initializer to pre-set the weight for different layers
- Try different learning rates
- Customize the loss function to heavily penalize false negative (FN) predictions

I also think about convolution layers but since this is structured tabular data, there's no real relevant connection to the surrounding data points like in unstructured data. Hence, applying convolution results in less accurate data to train with.

Subsequent neural network models will also undergo the same optimizations as above to obtain the best AUC score.

In [26]:
```
# split train, val, test again with engineered features
outcome = news_df["is_popular"]
# features = news_df[high_performance_predictors]
features = news_df.drop(columns=exclude_cols)
# features = news_df.drop(columns=['timedelta', 'is_popular', 'article_id'])
prng = np.random.RandomState(42)
X_train, X_test, y_train, y_test = train_test_split(features, outcome, test_size=0.1, random_state=prng)
# X_val, y_val = X_test, y_test
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.1, random_state=prng)


print(X_train.shape, X_val.shape, X_test.shape)
```

```
(24083, 44) (2676, 44) (2974, 44)
```

In [27]:
```python
from sklearn.preprocessing import MinMaxScaler

# normalize data
scaler = MinMaxScaler(feature_range=(-1, 1))
# scaler = StandardScaler()
# scaler.fit(features)
columns_not_to_scale = [col for col in X_train.columns if col not in binary_cols]
scaler.fit(X_train[columns_not_to_scale])

X_train[columns_not_to_scale] = scaler.transform(X_train[columns_not_to_scale])
X_val[columns_not_to_scale] = scaler.transform(X_val[columns_not_to_scale])
X_test[columns_not_to_scale] = scaler.transform(X_test[columns_not_to_scale])
```

In [28]:
```python
def plot_history(fit_history):
    train_auc_col = [col for col in fit_history.keys() if col.startswith('auc')][0]
    val_auc_col = [col for col in fit_history.keys() if col.startswith('val_auc')][0]
    plt.plot(fit_history[train_auc_col], label='Training AUC')
    plt.plot(fit_history[val_auc_col], label='Validation AUC')
    plt.xlabel('Epoch')
    plt.ylabel('AUC Score')
    plt.title('Training and Validation AUC score')
    plt.legend()
    plt.show()
```

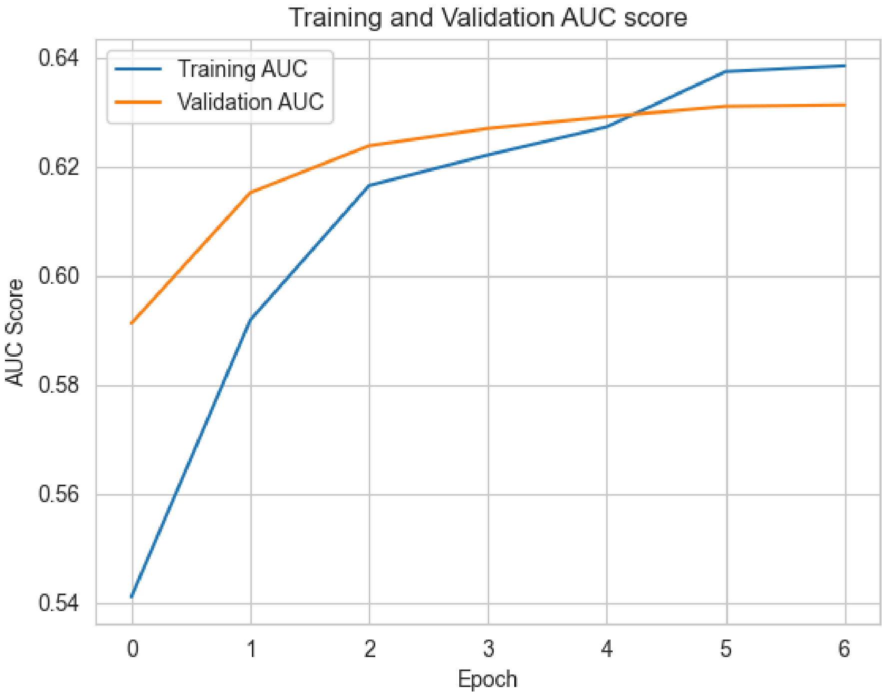
In [29]:
```python
y_train_sigmoid, y_val_sigmoid, y_test_sigmoid = y_train, y_val, y_test
```

In [30]:
```python
import tensorflow as tf

def custom_loss(y_true, y_pred):
    # Define weights
    false_positive_weight = 1.0
    false_negative_weight = 10000.0

    # Calculate binary cross entropy
    bce = tf.keras.losses.BinaryCrossentropy()

    # Calculate loss
    loss = bce(y_true, y_pred)

    # Calculate weighted loss
    weighted_loss = tf.where(tf.greater(y_true, y_pred), false_negative_weight * loss, false_positive_weigh

    return tf.reduce_mean(weighted_loss)
```

In [31]:
```python
from keras.metrics import AUC, F1Score
from keras.models import Sequential
from keras.layers import Input, Dense, Normalization, Dropout, BatchNormalization
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
from keras.regularizers import l1
import keras

# Build the simple fully connected single hidden layer network model
# simple_model = Sequential([
#     Input(shape=X_train.shape[1:]),
#     Dense(22, activation='relu', kernel_regularizer=l1(0.5)),
#     Dropout(0.7),
#     Dense(1, activation='sigmoid', kernel_regularizer=l1(0.5), kernel_initializer='glorot_normal')
# ])
simple_model = Sequential([
    Input(shape=X_train.shape[1:]),
    # Normalization(axis=-1),
    Dense(256, activation='relu', kernel_regularizer=l1(0.5)),
    Dropout(0.4),
    Dense(1, activation='sigmoid', kernel_regularizer=l1(0.5), kernel_initializer='glorot_normal')
])

# Compile the model
opt = Adam(learning_rate=0.00001)
simple_model.compile(loss=custom_loss, optimizer=opt, metrics=[AUC(), 'accuracy', F1Score()])
# print(simple_model.summary())

# Fit the model
keras.utils.set_random_seed(42)  # for reproducibility
# simple_history = simple_model.fit(X_train, y_train_sigmoid, validation_data=(X_val, y_val_sigmoid), epoch
simple_history = simple_model.fit(X_train, y_train_sigmoid, validation_data=(X_val, y_val_sigmoid), epochs=
```

```
plot_history(simple_history.history)
```

```
Epoch 1/200
8028/8028 ━━━━━━━━━━━━━━━━━━━━ 6s 712us/step - accuracy: 0.7593 - auc: 0.5217 - f1_score: 0.2174 - loss: 1227.2429 - val_accuracy: 0.8610 - val_auc: 0.5912 - val_f1_score: 0.2231 - val_loss: 1187.1758
Epoch 2/200
8028/8028 ━━━━━━━━━━━━━━━━━━━━ 5s 645us/step - accuracy: 0.7919 - auc: 0.5843 - f1_score: 0.2174 - loss: 1151.6146 - val_accuracy: 0.8333 - val_auc: 0.6152 - val_f1_score: 0.2231 - val_loss: 1143.6550
Epoch 3/200
8028/8028 ━━━━━━━━━━━━━━━━━━━━ 5s 618us/step - accuracy: 0.7922 - auc: 0.6138 - f1_score: 0.2174 - loss: 1114.5966 - val_accuracy: 0.8188 - val_auc: 0.6238 - val_f1_score: 0.2231 - val_loss: 1115.6023
Epoch 4/200
8028/8028 ━━━━━━━━━━━━━━━━━━━━ 5s 617us/step - accuracy: 0.7836 - auc: 0.6218 - f1_score: 0.2174 - loss: 1084.9539 - val_accuracy: 0.8079 - val_auc: 0.6269 - val_f1_score: 0.2231 - val_loss: 1094.1700
Epoch 5/200
8028/8028 ━━━━━━━━━━━━━━━━━━━━ 5s 665us/step - accuracy: 0.7831 - auc: 0.6221 - f1_score: 0.2174 - loss: 1068.6283 - val_accuracy: 0.8046 - val_auc: 0.6291 - val_f1_score: 0.2231 - val_loss: 1078.6821
Epoch 6/200
8028/8028 ━━━━━━━━━━━━━━━━━━━━ 5s 651us/step - accuracy: 0.7860 - auc: 0.6342 - f1_score: 0.2174 - loss: 1046.2969 - val_accuracy: 0.8038 - val_auc: 0.6310 - val_f1_score: 0.2231 - val_loss: 1064.0073
Epoch 7/200
8028/8028 ━━━━━━━━━━━━━━━━━━━━ 5s 639us/step - accuracy: 0.7879 - auc: 0.6387 - f1_score: 0.2174 - loss: 1031.2081 - val_accuracy: 0.8008 - val_auc: 0.6312 - val_f1_score: 0.2231 - val_loss: 1050.4274
Epoch 7: early stopping
```



In [32]:
```
update_summary(summary_df,
               'Sigmoid NN',
               y_train_sigmoid,
               simple_model.predict(X_train),
               y_val_sigmoid,
               simple_model.predict(X_val),
               y_test_sigmoid,
               simple_model.predict(X_test),
               class1_only=True)
summary_df
```

```
753/753 ━━━━━━━━━━━━━━━━━━━━ 0s 361us/step
84/84 ━━━━━━━━━━━━━━━━━━━━ 0s 316us/step
93/93 ━━━━━━━━━━━━━━━━━━━━ 0s 337us/step
```

Out[32]:

	Model	Train AUC	Val AUC	Test AUC
0	Logit as Benchmark	0.6707	0.6637	0.6637
1	LASSO Logit	0.6835	0.6791	0.6791
2	Random Forest CV	0.9686	0.7086	0.7086
3	GBM CV	0.8193	0.7133	0.7133
4	Sigmoid NN	0.6578	0.6321	0.6611

The sigmoid neural network performs worse compared to the ensemble models even with all the trial and errors with optimizations. Interestingly, the more layers or neurons added, the worse the performance becomes. Although I do not have good explanation for this behavior, it seems likely that the neural network is not fit for structured data compared to ensemble models. There are also possible improvements for label engineering or other network type/configuration. However, with the dataset, I have already exhausted possible enegineering option that can be done. Hence, moving forward in the notebook, I will experiment with other network configurations.

NN with Softmax activation

This network is similar to the network using sigmoid activation functions, with the exception in the output layer: instead of a single neuron using sigmoid function, it uses a 2-neuron layer with a softmax activiation function. The network is also optimized with all the options like the sigmoid neural network.

In [33]:
```
from keras.utils import to_categorical

print(f"Dimension of y: {y_train.shape}")

# Convert target variables to categorical
num_classes = 2
y_sets = [y_train, y_val, y_test]
y_train, y_val, y_test = [to_categorical(y, num_classes=num_classes) for y in y_sets]
print(f"Dimension of y: {y_train.shape}")
```

```
Dimension of y: (24083,)
Dimension of y: (24083, 2)
```

In [34]:
```
def custom_categorical_loss(y_true, y_pred):
    # Define class weights
    class_weights = tf.constant([1.0, 1000.0])  # Assuming there are 2 classes with different weights

    # Calculate Categorical Crossentropy
    cat_crossentropy = tf.keras.losses.CategoricalCrossentropy(from_logits=False)

    # Calculate raw loss
    raw_loss = cat_crossentropy(y_true, y_pred)

    # Apply class weights
    weighted_loss = raw_loss * class_weights

    # Reduce along the class axis
    weighted_loss = tf.reduce_mean(weighted_loss, axis=-1)

    return weighted_loss
```

In [35]:
```
from keras.models import Sequential
from keras.layers import Input, Dense
import keras

# Build the simple fully connected single hidden layer network model
simple_softmax_model = Sequential([
    Input(shape=X_train.shape[1:]),
    # Dropout(0.4),
    # Normalization(),
    # Dense(16384, activation='relu', kernel_regularizer=l1(0.5), kernel_initializer='glorot_normal'),
    Dense(22, activation='relu', kernel_regularizer=l1(0.5)),
    Dropout(0.7),
    Dense(2, activation='softmax', kernel_regularizer=l1(0.5), kernel_initializer='glorot_normal')
])

# Compile the model
opt = Adam(learning_rate=0.00001)
simple_softmax_model.compile(loss=custom_categorical_loss, optimizer=opt, metrics=[AUC(), 'accuracy', F1Sco
# print(simple_model.summary())

# Fit the model
keras.utils.set_random_seed(42)  # for reproducibility
simple_softmax_history = simple_softmax_model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=
# simple_softmax_history = simple_softmax_model.fit(X_train, y_train, validation_data=(X_test, y_test), epo

# Evaluation of the model on the validation set
scores = simple_softmax_model.evaluate(X_val, y_val)
# scores = simple_softmax_model.evaluate(X_test, y_test)

plot_history(simple_softmax_history.history)
```

```
Epoch 1/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.7979 - auc_1: 0.8498 - f1_score: 0.5033 - loss: 362.8
992 - val_accuracy: 0.8744 - val_auc_1: 0.8720 - val_f1_score: 0.4665 - val_loss: 278.6981
Epoch 2/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 711us/step - accuracy: 0.8102 - auc_1: 0.8540 - f1_score: 0.5027 - loss: 35
6.7601 - val_accuracy: 0.8744 - val_auc_1: 0.8721 - val_f1_score: 0.4665 - val_loss: 277.3033
Epoch 3/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 686us/step - accuracy: 0.8213 - auc_1: 0.8601 - f1_score: 0.5021 - loss: 34
6.4251 - val_accuracy: 0.8744 - val_auc_1: 0.8721 - val_f1_score: 0.4665 - val_loss: 275.6946
Epoch 4/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 699us/step - accuracy: 0.8281 - auc_1: 0.8627 - f1_score: 0.5004 - loss: 33
9.9158 - val_accuracy: 0.8744 - val_auc_1: 0.8728 - val_f1_score: 0.4665 - val_loss: 274.0124
Epoch 5/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 657us/step - accuracy: 0.8379 - auc_1: 0.8674 - f1_score: 0.5025 - loss: 33
0.7993 - val_accuracy: 0.8744 - val_auc_1: 0.8732 - val_f1_score: 0.4665 - val_loss: 272.4661
Epoch 6/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 700us/step - accuracy: 0.8386 - auc_1: 0.8699 - f1_score: 0.5088 - loss: 32
3.7843 - val_accuracy: 0.8744 - val_auc_1: 0.8735 - val_f1_score: 0.4665 - val_loss: 271.1880
Epoch 7/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 648us/step - accuracy: 0.8473 - auc_1: 0.8710 - f1_score: 0.5026 - loss: 32
0.2536 - val_accuracy: 0.8744 - val_auc_1: 0.8739 - val_f1_score: 0.4665 - val_loss: 269.8339
Epoch 8/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 817us/step - accuracy: 0.8463 - auc_1: 0.8699 - f1_score: 0.5040 - loss: 31
7.9762 - val_accuracy: 0.8744 - val_auc_1: 0.8740 - val_f1_score: 0.4665 - val_loss: 268.7202
Epoch 9/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 775us/step - accuracy: 0.8514 - auc_1: 0.8731 - f1_score: 0.4986 - loss: 31
1.6431 - val_accuracy: 0.8744 - val_auc_1: 0.8745 - val_f1_score: 0.4665 - val_loss: 267.7656
Epoch 10/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 861us/step - accuracy: 0.8555 - auc_1: 0.8728 - f1_score: 0.4971 - loss: 31
1.7391 - val_accuracy: 0.8744 - val_auc_1: 0.8747 - val_f1_score: 0.4665 - val_loss: 266.8932
Epoch 11/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 816us/step - accuracy: 0.8572 - auc_1: 0.8730 - f1_score: 0.4870 - loss: 30
8.0860 - val_accuracy: 0.8744 - val_auc_1: 0.8751 - val_f1_score: 0.4665 - val_loss: 266.0588
Epoch 12/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 656us/step - accuracy: 0.8626 - auc_1: 0.8744 - f1_score: 0.4966 - loss: 30
4.9740 - val_accuracy: 0.8744 - val_auc_1: 0.8751 - val_f1_score: 0.4665 - val_loss: 265.3321
Epoch 13/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 679us/step - accuracy: 0.8617 - auc_1: 0.8726 - f1_score: 0.4964 - loss: 30
5.2619 - val_accuracy: 0.8744 - val_auc_1: 0.8753 - val_f1_score: 0.4665 - val_loss: 264.6334
Epoch 14/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 663us/step - accuracy: 0.8653 - auc_1: 0.8784 - f1_score: 0.4888 - loss: 29
7.0869 - val_accuracy: 0.8744 - val_auc_1: 0.8758 - val_f1_score: 0.4665 - val_loss: 263.9635
Epoch 15/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 758us/step - accuracy: 0.8671 - auc_1: 0.8809 - f1_score: 0.4939 - loss: 29
2.7044 - val_accuracy: 0.8744 - val_auc_1: 0.8761 - val_f1_score: 0.4665 - val_loss: 263.2576
Epoch 16/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 720us/step - accuracy: 0.8670 - auc_1: 0.8757 - f1_score: 0.4860 - loss: 29
5.8681 - val_accuracy: 0.8744 - val_auc_1: 0.8767 - val_f1_score: 0.4665 - val_loss: 262.5631
Epoch 17/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 702us/step - accuracy: 0.8683 - auc_1: 0.8749 - f1_score: 0.4895 - loss: 29
5.3177 - val_accuracy: 0.8744 - val_auc_1: 0.8771 - val_f1_score: 0.4665 - val_loss: 261.9109
Epoch 18/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 713us/step - accuracy: 0.8690 - auc_1: 0.8795 - f1_score: 0.4883 - loss: 28
8.2843 - val_accuracy: 0.8744 - val_auc_1: 0.8774 - val_f1_score: 0.4665 - val_loss: 261.2415
Epoch 19/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 682us/step - accuracy: 0.8705 - auc_1: 0.8794 - f1_score: 0.4835 - loss: 28
8.2390 - val_accuracy: 0.8744 - val_auc_1: 0.8780 - val_f1_score: 0.4665 - val_loss: 260.5918
Epoch 20/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 714us/step - accuracy: 0.8713 - auc_1: 0.8781 - f1_score: 0.4856 - loss: 28
8.9445 - val_accuracy: 0.8744 - val_auc_1: 0.8786 - val_f1_score: 0.4665 - val_loss: 259.9552
Epoch 21/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 692us/step - accuracy: 0.8716 - auc_1: 0.8801 - f1_score: 0.4845 - loss: 28
5.4436 - val_accuracy: 0.8744 - val_auc_1: 0.8790 - val_f1_score: 0.4665 - val_loss: 259.3060
Epoch 22/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 731us/step - accuracy: 0.8723 - auc_1: 0.8801 - f1_score: 0.4851 - loss: 28
4.1415 - val_accuracy: 0.8744 - val_auc_1: 0.8794 - val_f1_score: 0.4665 - val_loss: 258.7033
Epoch 23/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 697us/step - accuracy: 0.8725 - auc_1: 0.8796 - f1_score: 0.4803 - loss: 28
4.2573 - val_accuracy: 0.8744 - val_auc_1: 0.8799 - val_f1_score: 0.4665 - val_loss: 258.0611
Epoch 24/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8735 - auc_1: 0.8792 - f1_score: 0.4845 - loss: 282.3
493 - val_accuracy: 0.8744 - val_auc_1: 0.8803 - val_f1_score: 0.4665 - val_loss: 257.4241
Epoch 25/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 933us/step - accuracy: 0.8724 - auc_1: 0.8769 - f1_score: 0.4744 - loss: 28
4.5573 - val_accuracy: 0.8744 - val_auc_1: 0.8810 - val_f1_score: 0.4665 - val_loss: 256.8414
Epoch 26/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 990us/step - accuracy: 0.8739 - auc_1: 0.8790 - f1_score: 0.4738 - loss: 28
1.0407 - val_accuracy: 0.8744 - val_auc_1: 0.8815 - val_f1_score: 0.4665 - val_loss: 256.2506
Epoch 27/300
```

```
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 796us/step - accuracy: 0.8746 - auc_1: 0.8759 - f1_score: 0.4795 - loss: 28
3.3280 - val_accuracy: 0.8744 - val_auc_1: 0.8823 - val_f1_score: 0.4665 - val_loss: 255.6639
Epoch 28/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 747us/step - accuracy: 0.8748 - auc_1: 0.8810 - f1_score: 0.4832 - loss: 27
8.7050 - val_accuracy: 0.8744 - val_auc_1: 0.8833 - val_f1_score: 0.4665 - val_loss: 255.0251
Epoch 29/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 689us/step - accuracy: 0.8748 - auc_1: 0.8800 - f1_score: 0.4742 - loss: 27
7.6123 - val_accuracy: 0.8744 - val_auc_1: 0.8841 - val_f1_score: 0.4665 - val_loss: 254.4524
Epoch 30/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 681us/step - accuracy: 0.8754 - auc_1: 0.8811 - f1_score: 0.4759 - loss: 27
5.9504 - val_accuracy: 0.8744 - val_auc_1: 0.8845 - val_f1_score: 0.4665 - val_loss: 253.8326
Epoch 31/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 651us/step - accuracy: 0.8755 - auc_1: 0.8821 - f1_score: 0.4835 - loss: 27
5.2339 - val_accuracy: 0.8744 - val_auc_1: 0.8852 - val_f1_score: 0.4665 - val_loss: 253.2269
Epoch 32/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 709us/step - accuracy: 0.8751 - auc_1: 0.8811 - f1_score: 0.4756 - loss: 27
5.4437 - val_accuracy: 0.8744 - val_auc_1: 0.8866 - val_f1_score: 0.4665 - val_loss: 252.5927
Epoch 33/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 677us/step - accuracy: 0.8755 - auc_1: 0.8824 - f1_score: 0.4743 - loss: 27
2.7816 - val_accuracy: 0.8744 - val_auc_1: 0.8872 - val_f1_score: 0.4665 - val_loss: 251.9821
Epoch 34/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 703us/step - accuracy: 0.8746 - auc_1: 0.8822 - f1_score: 0.4735 - loss: 27
1.9933 - val_accuracy: 0.8744 - val_auc_1: 0.8880 - val_f1_score: 0.4665 - val_loss: 251.4530
Epoch 35/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 707us/step - accuracy: 0.8757 - auc_1: 0.8826 - f1_score: 0.4726 - loss: 27
1.4977 - val_accuracy: 0.8744 - val_auc_1: 0.8887 - val_f1_score: 0.4665 - val_loss: 250.9027
Epoch 36/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 676us/step - accuracy: 0.8760 - auc_1: 0.8797 - f1_score: 0.4722 - loss: 27
2.7246 - val_accuracy: 0.8744 - val_auc_1: 0.8892 - val_f1_score: 0.4665 - val_loss: 250.3896
Epoch 37/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 640us/step - accuracy: 0.8761 - auc_1: 0.8837 - f1_score: 0.4749 - loss: 26
9.2080 - val_accuracy: 0.8744 - val_auc_1: 0.8900 - val_f1_score: 0.4665 - val_loss: 249.8452
Epoch 38/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 665us/step - accuracy: 0.8764 - auc_1: 0.8825 - f1_score: 0.4701 - loss: 26
8.6750 - val_accuracy: 0.8744 - val_auc_1: 0.8908 - val_f1_score: 0.4665 - val_loss: 249.3342
Epoch 39/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 649us/step - accuracy: 0.8762 - auc_1: 0.8836 - f1_score: 0.4738 - loss: 26
7.5653 - val_accuracy: 0.8744 - val_auc_1: 0.8912 - val_f1_score: 0.4665 - val_loss: 248.8061
Epoch 40/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 796us/step - accuracy: 0.8763 - auc_1: 0.8823 - f1_score: 0.4751 - loss: 26
8.1950 - val_accuracy: 0.8744 - val_auc_1: 0.8919 - val_f1_score: 0.4665 - val_loss: 248.2999
Epoch 41/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 776us/step - accuracy: 0.8766 - auc_1: 0.8853 - f1_score: 0.4723 - loss: 26
5.0376 - val_accuracy: 0.8744 - val_auc_1: 0.8926 - val_f1_score: 0.4665 - val_loss: 247.8090
Epoch 42/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 848us/step - accuracy: 0.8769 - auc_1: 0.8829 - f1_score: 0.4725 - loss: 26
6.5608 - val_accuracy: 0.8744 - val_auc_1: 0.8933 - val_f1_score: 0.4665 - val_loss: 247.3102
Epoch 43/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 858us/step - accuracy: 0.8769 - auc_1: 0.8834 - f1_score: 0.4706 - loss: 26
5.8503 - val_accuracy: 0.8744 - val_auc_1: 0.8936 - val_f1_score: 0.4665 - val_loss: 246.8232
Epoch 44/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 714us/step - accuracy: 0.8763 - auc_1: 0.8869 - f1_score: 0.4734 - loss: 26
2.3146 - val_accuracy: 0.8744 - val_auc_1: 0.8936 - val_f1_score: 0.4665 - val_loss: 246.3559
Epoch 45/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 725us/step - accuracy: 0.8767 - auc_1: 0.8858 - f1_score: 0.4705 - loss: 26
1.8951 - val_accuracy: 0.8744 - val_auc_1: 0.8941 - val_f1_score: 0.4665 - val_loss: 245.8770
Epoch 46/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 688us/step - accuracy: 0.8769 - auc_1: 0.8868 - f1_score: 0.4722 - loss: 26
1.4983 - val_accuracy: 0.8744 - val_auc_1: 0.8945 - val_f1_score: 0.4665 - val_loss: 245.4679
Epoch 47/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 684us/step - accuracy: 0.8765 - auc_1: 0.8858 - f1_score: 0.4692 - loss: 26
0.5423 - val_accuracy: 0.8744 - val_auc_1: 0.8950 - val_f1_score: 0.4665 - val_loss: 245.0021
Epoch 48/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 712us/step - accuracy: 0.8772 - auc_1: 0.8859 - f1_score: 0.4724 - loss: 26
0.6205 - val_accuracy: 0.8744 - val_auc_1: 0.8950 - val_f1_score: 0.4665 - val_loss: 244.5989
Epoch 49/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 732us/step - accuracy: 0.8766 - auc_1: 0.8850 - f1_score: 0.4702 - loss: 26
1.2877 - val_accuracy: 0.8744 - val_auc_1: 0.8956 - val_f1_score: 0.4665 - val_loss: 244.1949
Epoch 50/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 769us/step - accuracy: 0.8773 - auc_1: 0.8873 - f1_score: 0.4716 - loss: 25
8.7734 - val_accuracy: 0.8744 - val_auc_1: 0.8957 - val_f1_score: 0.4665 - val_loss: 243.7528
Epoch 51/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 663us/step - accuracy: 0.8774 - auc_1: 0.8834 - f1_score: 0.4717 - loss: 26
0.8684 - val_accuracy: 0.8744 - val_auc_1: 0.8958 - val_f1_score: 0.4665 - val_loss: 243.3228
Epoch 52/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 674us/step - accuracy: 0.8774 - auc_1: 0.8884 - f1_score: 0.4724 - loss: 25
6.6356 - val_accuracy: 0.8744 - val_auc_1: 0.8966 - val_f1_score: 0.4665 - val_loss: 242.8704
Epoch 53/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 680us/step - accuracy: 0.8770 - auc_1: 0.8864 - f1_score: 0.4697 - loss: 25
```

```
7.8554 - val_accuracy: 0.8744 - val_auc_1: 0.8972 - val_f1_score: 0.4665 - val_loss: 242.4455
Epoch 54/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 701us/step - accuracy: 0.8777 - auc_1: 0.8863 - f1_score: 0.4741 - loss: 25
7.8239 - val_accuracy: 0.8744 - val_auc_1: 0.8972 - val_f1_score: 0.4665 - val_loss: 242.0477
Epoch 55/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 673us/step - accuracy: 0.8774 - auc_1: 0.8875 - f1_score: 0.4731 - loss: 25
5.9737 - val_accuracy: 0.8744 - val_auc_1: 0.8977 - val_f1_score: 0.4665 - val_loss: 241.6282
Epoch 56/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 669us/step - accuracy: 0.8774 - auc_1: 0.8856 - f1_score: 0.4707 - loss: 25
7.1150 - val_accuracy: 0.8744 - val_auc_1: 0.8974 - val_f1_score: 0.4665 - val_loss: 241.2356
Epoch 57/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 672us/step - accuracy: 0.8775 - auc_1: 0.8878 - f1_score: 0.4713 - loss: 25
4.9570 - val_accuracy: 0.8744 - val_auc_1: 0.8974 - val_f1_score: 0.4665 - val_loss: 240.8176
Epoch 58/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 680us/step - accuracy: 0.8777 - auc_1: 0.8875 - f1_score: 0.4701 - loss: 25
4.3116 - val_accuracy: 0.8744 - val_auc_1: 0.8977 - val_f1_score: 0.4665 - val_loss: 240.4065
Epoch 59/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 619us/step - accuracy: 0.8774 - auc_1: 0.8875 - f1_score: 0.4717 - loss: 25
4.4640 - val_accuracy: 0.8744 - val_auc_1: 0.8981 - val_f1_score: 0.4665 - val_loss: 240.0066
Epoch 60/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 683us/step - accuracy: 0.8773 - auc_1: 0.8877 - f1_score: 0.4697 - loss: 25
3.6056 - val_accuracy: 0.8744 - val_auc_1: 0.8986 - val_f1_score: 0.4665 - val_loss: 239.5902
Epoch 61/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 674us/step - accuracy: 0.8772 - auc_1: 0.8854 - f1_score: 0.4698 - loss: 25
5.0900 - val_accuracy: 0.8744 - val_auc_1: 0.8985 - val_f1_score: 0.4665 - val_loss: 239.2228
Epoch 62/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 597us/step - accuracy: 0.8773 - auc_1: 0.8888 - f1_score: 0.4679 - loss: 25
2.0257 - val_accuracy: 0.8744 - val_auc_1: 0.8986 - val_f1_score: 0.4665 - val_loss: 238.8229
Epoch 63/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 692us/step - accuracy: 0.8779 - auc_1: 0.8892 - f1_score: 0.4705 - loss: 25
1.1526 - val_accuracy: 0.8744 - val_auc_1: 0.8990 - val_f1_score: 0.4665 - val_loss: 238.4407
Epoch 64/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 804us/step - accuracy: 0.8777 - auc_1: 0.8866 - f1_score: 0.4683 - loss: 25
2.7232 - val_accuracy: 0.8744 - val_auc_1: 0.8993 - val_f1_score: 0.4665 - val_loss: 238.0845
Epoch 65/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 694us/step - accuracy: 0.8776 - auc_1: 0.8882 - f1_score: 0.4698 - loss: 25
1.6288 - val_accuracy: 0.8744 - val_auc_1: 0.8991 - val_f1_score: 0.4665 - val_loss: 237.6721
Epoch 66/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 829us/step - accuracy: 0.8776 - auc_1: 0.8867 - f1_score: 0.4684 - loss: 25
1.6686 - val_accuracy: 0.8744 - val_auc_1: 0.8994 - val_f1_score: 0.4665 - val_loss: 237.2928
Epoch 67/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 789us/step - accuracy: 0.8777 - auc_1: 0.8868 - f1_score: 0.4695 - loss: 25
1.6272 - val_accuracy: 0.8744 - val_auc_1: 0.8998 - val_f1_score: 0.4665 - val_loss: 236.9030
Epoch 68/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 675us/step - accuracy: 0.8777 - auc_1: 0.8888 - f1_score: 0.4693 - loss: 24
8.7715 - val_accuracy: 0.8744 - val_auc_1: 0.8998 - val_f1_score: 0.4665 - val_loss: 236.5314
Epoch 69/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 654us/step - accuracy: 0.8778 - auc_1: 0.8887 - f1_score: 0.4712 - loss: 24
8.8644 - val_accuracy: 0.8744 - val_auc_1: 0.9000 - val_f1_score: 0.4665 - val_loss: 236.1656
Epoch 70/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 601us/step - accuracy: 0.8776 - auc_1: 0.8866 - f1_score: 0.4699 - loss: 25
0.4906 - val_accuracy: 0.8744 - val_auc_1: 0.8999 - val_f1_score: 0.4665 - val_loss: 235.8376
Epoch 71/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 649us/step - accuracy: 0.8777 - auc_1: 0.8883 - f1_score: 0.4683 - loss: 24
8.6644 - val_accuracy: 0.8744 - val_auc_1: 0.8999 - val_f1_score: 0.4665 - val_loss: 235.4749
Epoch 72/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 630us/step - accuracy: 0.8775 - auc_1: 0.8880 - f1_score: 0.4692 - loss: 24
8.2770 - val_accuracy: 0.8744 - val_auc_1: 0.9004 - val_f1_score: 0.4665 - val_loss: 235.1111
Epoch 73/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 659us/step - accuracy: 0.8775 - auc_1: 0.8911 - f1_score: 0.4699 - loss: 24
5.8486 - val_accuracy: 0.8744 - val_auc_1: 0.9004 - val_f1_score: 0.4665 - val_loss: 234.7605
Epoch 74/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 675us/step - accuracy: 0.8778 - auc_1: 0.8882 - f1_score: 0.4694 - loss: 24
7.0561 - val_accuracy: 0.8744 - val_auc_1: 0.9003 - val_f1_score: 0.4665 - val_loss: 234.4345
Epoch 75/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 724us/step - accuracy: 0.8780 - auc_1: 0.8908 - f1_score: 0.4700 - loss: 24
5.5005 - val_accuracy: 0.8744 - val_auc_1: 0.9006 - val_f1_score: 0.4665 - val_loss: 234.0832
Epoch 76/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 702us/step - accuracy: 0.8781 - auc_1: 0.8890 - f1_score: 0.4701 - loss: 24
5.8316 - val_accuracy: 0.8744 - val_auc_1: 0.9006 - val_f1_score: 0.4665 - val_loss: 233.7584
Epoch 77/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 690us/step - accuracy: 0.8777 - auc_1: 0.8896 - f1_score: 0.4690 - loss: 24
5.7168 - val_accuracy: 0.8744 - val_auc_1: 0.9004 - val_f1_score: 0.4665 - val_loss: 233.4050
Epoch 78/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 712us/step - accuracy: 0.8781 - auc_1: 0.8917 - f1_score: 0.4706 - loss: 24
3.3171 - val_accuracy: 0.8744 - val_auc_1: 0.9005 - val_f1_score: 0.4665 - val_loss: 233.0739
Epoch 79/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 720us/step - accuracy: 0.8776 - auc_1: 0.8906 - f1_score: 0.4682 - loss: 24
3.9493 - val_accuracy: 0.8744 - val_auc_1: 0.9005 - val_f1_score: 0.4665 - val_loss: 232.7537
```

```
Epoch 80/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 710us/step - accuracy: 0.8778 - auc_1: 0.8874 - f1_score: 0.4693 - loss: 24
5.1088 - val_accuracy: 0.8744 - val_auc_1: 0.9007 - val_f1_score: 0.4665 - val_loss: 232.4242
Epoch 81/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 696us/step - accuracy: 0.8779 - auc_1: 0.8935 - f1_score: 0.4688 - loss: 24
0.8039 - val_accuracy: 0.8744 - val_auc_1: 0.9011 - val_f1_score: 0.4665 - val_loss: 232.0849
Epoch 82/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 682us/step - accuracy: 0.8781 - auc_1: 0.8903 - f1_score: 0.4704 - loss: 24
2.6351 - val_accuracy: 0.8744 - val_auc_1: 0.9013 - val_f1_score: 0.4665 - val_loss: 231.7302
Epoch 83/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 679us/step - accuracy: 0.8775 - auc_1: 0.8936 - f1_score: 0.4679 - loss: 24
0.3530 - val_accuracy: 0.8744 - val_auc_1: 0.9012 - val_f1_score: 0.4665 - val_loss: 231.4103
Epoch 84/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 722us/step - accuracy: 0.8778 - auc_1: 0.8913 - f1_score: 0.4694 - loss: 24
2.0833 - val_accuracy: 0.8744 - val_auc_1: 0.9011 - val_f1_score: 0.4665 - val_loss: 231.1329
Epoch 85/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 665us/step - accuracy: 0.8777 - auc_1: 0.8884 - f1_score: 0.4674 - loss: 24
3.4775 - val_accuracy: 0.8744 - val_auc_1: 0.9012 - val_f1_score: 0.4665 - val_loss: 230.8317
Epoch 86/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 785us/step - accuracy: 0.8777 - auc_1: 0.8917 - f1_score: 0.4674 - loss: 24
0.9331 - val_accuracy: 0.8744 - val_auc_1: 0.9010 - val_f1_score: 0.4665 - val_loss: 230.5242
Epoch 87/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 877us/step - accuracy: 0.8779 - auc_1: 0.8912 - f1_score: 0.4682 - loss: 24
0.5585 - val_accuracy: 0.8744 - val_auc_1: 0.9016 - val_f1_score: 0.4665 - val_loss: 230.2099
Epoch 88/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 920us/step - accuracy: 0.8778 - auc_1: 0.8902 - f1_score: 0.4680 - loss: 24
0.6687 - val_accuracy: 0.8744 - val_auc_1: 0.9016 - val_f1_score: 0.4665 - val_loss: 229.9123
Epoch 89/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 834us/step - accuracy: 0.8776 - auc_1: 0.8951 - f1_score: 0.4683 - loss: 23
7.3921 - val_accuracy: 0.8744 - val_auc_1: 0.9018 - val_f1_score: 0.4665 - val_loss: 229.5898
Epoch 90/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 678us/step - accuracy: 0.8777 - auc_1: 0.8897 - f1_score: 0.4683 - loss: 24
0.5728 - val_accuracy: 0.8744 - val_auc_1: 0.9017 - val_f1_score: 0.4665 - val_loss: 229.3317
Epoch 91/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 744us/step - accuracy: 0.8777 - auc_1: 0.8922 - f1_score: 0.4680 - loss: 23
8.2964 - val_accuracy: 0.8744 - val_auc_1: 0.9020 - val_f1_score: 0.4665 - val_loss: 229.0263
Epoch 92/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 716us/step - accuracy: 0.8780 - auc_1: 0.8916 - f1_score: 0.4695 - loss: 23
8.3285 - val_accuracy: 0.8744 - val_auc_1: 0.9018 - val_f1_score: 0.4665 - val_loss: 228.7428
Epoch 93/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 682us/step - accuracy: 0.8781 - auc_1: 0.8921 - f1_score: 0.4698 - loss: 23
7.8770 - val_accuracy: 0.8744 - val_auc_1: 0.9018 - val_f1_score: 0.4665 - val_loss: 228.4573
Epoch 94/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 707us/step - accuracy: 0.8778 - auc_1: 0.8917 - f1_score: 0.4676 - loss: 23
8.0952 - val_accuracy: 0.8744 - val_auc_1: 0.9017 - val_f1_score: 0.4665 - val_loss: 228.1876
Epoch 95/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 677us/step - accuracy: 0.8777 - auc_1: 0.8930 - f1_score: 0.4682 - loss: 23
6.6122 - val_accuracy: 0.8744 - val_auc_1: 0.9018 - val_f1_score: 0.4665 - val_loss: 227.9052
Epoch 96/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 733us/step - accuracy: 0.8778 - auc_1: 0.8934 - f1_score: 0.4675 - loss: 23
6.4809 - val_accuracy: 0.8744 - val_auc_1: 0.9018 - val_f1_score: 0.4665 - val_loss: 227.6341
Epoch 97/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 708us/step - accuracy: 0.8778 - auc_1: 0.8946 - f1_score: 0.4681 - loss: 23
5.0625 - val_accuracy: 0.8744 - val_auc_1: 0.9019 - val_f1_score: 0.4665 - val_loss: 227.3583
Epoch 98/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 674us/step - accuracy: 0.8778 - auc_1: 0.8915 - f1_score: 0.4676 - loss: 23
6.7674 - val_accuracy: 0.8744 - val_auc_1: 0.9019 - val_f1_score: 0.4665 - val_loss: 227.1115
Epoch 99/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 684us/step - accuracy: 0.8777 - auc_1: 0.8940 - f1_score: 0.4674 - loss: 23
5.1824 - val_accuracy: 0.8744 - val_auc_1: 0.9017 - val_f1_score: 0.4665 - val_loss: 226.8241
Epoch 100/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 725us/step - accuracy: 0.8778 - auc_1: 0.8945 - f1_score: 0.4682 - loss: 23
4.3376 - val_accuracy: 0.8744 - val_auc_1: 0.9019 - val_f1_score: 0.4665 - val_loss: 226.5514
Epoch 101/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 718us/step - accuracy: 0.8777 - auc_1: 0.8962 - f1_score: 0.4683 - loss: 23
3.1636 - val_accuracy: 0.8744 - val_auc_1: 0.9018 - val_f1_score: 0.4665 - val_loss: 226.2628
Epoch 102/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 892us/step - accuracy: 0.8779 - auc_1: 0.8935 - f1_score: 0.4690 - loss: 23
3.6058 - val_accuracy: 0.8744 - val_auc_1: 0.9018 - val_f1_score: 0.4665 - val_loss: 226.0149
Epoch 103/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 883us/step - accuracy: 0.8780 - auc_1: 0.8901 - f1_score: 0.4689 - loss: 23
6.1443 - val_accuracy: 0.8744 - val_auc_1: 0.9020 - val_f1_score: 0.4665 - val_loss: 225.7624
Epoch 104/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 885us/step - accuracy: 0.8778 - auc_1: 0.8933 - f1_score: 0.4675 - loss: 23
3.5368 - val_accuracy: 0.8744 - val_auc_1: 0.9018 - val_f1_score: 0.4665 - val_loss: 225.5280
Epoch 105/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 735us/step - accuracy: 0.8779 - auc_1: 0.8918 - f1_score: 0.4677 - loss: 23
4.1672 - val_accuracy: 0.8744 - val_auc_1: 0.9022 - val_f1_score: 0.4665 - val_loss: 225.3047
Epoch 106/300
```

```
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 680us/step - accuracy: 0.8778 - auc_1: 0.8913 - f1_score: 0.4681 - loss: 23
4.4978 - val_accuracy: 0.8744 - val_auc_1: 0.9021 - val_f1_score: 0.4665 - val_loss: 225.0623
Epoch 107/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 661us/step - accuracy: 0.8778 - auc_1: 0.8932 - f1_score: 0.4674 - loss: 23
3.2300 - val_accuracy: 0.8744 - val_auc_1: 0.9023 - val_f1_score: 0.4665 - val_loss: 224.8128
Epoch 108/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 715us/step - accuracy: 0.8779 - auc_1: 0.8926 - f1_score: 0.4679 - loss: 23
3.7906 - val_accuracy: 0.8744 - val_auc_1: 0.9022 - val_f1_score: 0.4665 - val_loss: 224.5777
Epoch 109/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 671us/step - accuracy: 0.8780 - auc_1: 0.8935 - f1_score: 0.4689 - loss: 23
2.0510 - val_accuracy: 0.8744 - val_auc_1: 0.9023 - val_f1_score: 0.4665 - val_loss: 224.3383
Epoch 110/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 708us/step - accuracy: 0.8778 - auc_1: 0.8943 - f1_score: 0.4674 - loss: 23
1.5306 - val_accuracy: 0.8744 - val_auc_1: 0.9024 - val_f1_score: 0.4665 - val_loss: 224.1097
Epoch 111/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 722us/step - accuracy: 0.8779 - auc_1: 0.8954 - f1_score: 0.4684 - loss: 23
0.0845 - val_accuracy: 0.8744 - val_auc_1: 0.9024 - val_f1_score: 0.4665 - val_loss: 223.8551
Epoch 112/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 690us/step - accuracy: 0.8778 - auc_1: 0.8940 - f1_score: 0.4674 - loss: 23
1.1389 - val_accuracy: 0.8744 - val_auc_1: 0.9024 - val_f1_score: 0.4665 - val_loss: 223.6440
Epoch 113/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 714us/step - accuracy: 0.8778 - auc_1: 0.8924 - f1_score: 0.4675 - loss: 23
1.8656 - val_accuracy: 0.8744 - val_auc_1: 0.9025 - val_f1_score: 0.4665 - val_loss: 223.4309
Epoch 114/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 730us/step - accuracy: 0.8778 - auc_1: 0.8931 - f1_score: 0.4675 - loss: 23
0.9336 - val_accuracy: 0.8744 - val_auc_1: 0.9026 - val_f1_score: 0.4665 - val_loss: 223.2133
Epoch 115/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 709us/step - accuracy: 0.8778 - auc_1: 0.8937 - f1_score: 0.4679 - loss: 23
0.2968 - val_accuracy: 0.8744 - val_auc_1: 0.9026 - val_f1_score: 0.4665 - val_loss: 223.0016
Epoch 116/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 932us/step - accuracy: 0.8778 - auc_1: 0.8941 - f1_score: 0.4675 - loss: 22
9.9728 - val_accuracy: 0.8744 - val_auc_1: 0.9025 - val_f1_score: 0.4665 - val_loss: 222.7746
Epoch 117/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 855us/step - accuracy: 0.8779 - auc_1: 0.8924 - f1_score: 0.4679 - loss: 23
1.1205 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 222.5906
Epoch 118/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 866us/step - accuracy: 0.8779 - auc_1: 0.8938 - f1_score: 0.4682 - loss: 22
9.9763 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 222.3900
Epoch 119/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 829us/step - accuracy: 0.8778 - auc_1: 0.8957 - f1_score: 0.4675 - loss: 22
8.1880 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 222.1814
Epoch 120/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 700us/step - accuracy: 0.8779 - auc_1: 0.8939 - f1_score: 0.4679 - loss: 22
9.5139 - val_accuracy: 0.8744 - val_auc_1: 0.9026 - val_f1_score: 0.4665 - val_loss: 221.9823
Epoch 121/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 704us/step - accuracy: 0.8779 - auc_1: 0.8966 - f1_score: 0.4679 - loss: 22
7.8776 - val_accuracy: 0.8744 - val_auc_1: 0.9026 - val_f1_score: 0.4665 - val_loss: 221.8023
Epoch 122/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 687us/step - accuracy: 0.8779 - auc_1: 0.8991 - f1_score: 0.4676 - loss: 22
5.8742 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 221.5708
Epoch 123/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 687us/step - accuracy: 0.8778 - auc_1: 0.8933 - f1_score: 0.4675 - loss: 22
9.1751 - val_accuracy: 0.8744 - val_auc_1: 0.9025 - val_f1_score: 0.4665 - val_loss: 221.4085
Epoch 124/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 683us/step - accuracy: 0.8778 - auc_1: 0.8957 - f1_score: 0.4675 - loss: 22
6.9142 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 221.2138
Epoch 125/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 734us/step - accuracy: 0.8778 - auc_1: 0.8953 - f1_score: 0.4675 - loss: 22
7.2702 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 221.0308
Epoch 126/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 730us/step - accuracy: 0.8779 - auc_1: 0.8988 - f1_score: 0.4683 - loss: 22
4.7628 - val_accuracy: 0.8744 - val_auc_1: 0.9030 - val_f1_score: 0.4665 - val_loss: 220.8235
Epoch 127/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 691us/step - accuracy: 0.8778 - auc_1: 0.8960 - f1_score: 0.4679 - loss: 22
6.6329 - val_accuracy: 0.8744 - val_auc_1: 0.9031 - val_f1_score: 0.4665 - val_loss: 220.6400
Epoch 128/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 704us/step - accuracy: 0.8778 - auc_1: 0.8945 - f1_score: 0.4675 - loss: 22
7.3389 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 220.4726
Epoch 129/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 697us/step - accuracy: 0.8778 - auc_1: 0.8965 - f1_score: 0.4674 - loss: 22
5.7496 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 220.3056
Epoch 130/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 838us/step - accuracy: 0.8778 - auc_1: 0.8964 - f1_score: 0.4675 - loss: 22
6.0429 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 220.1485
Epoch 131/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 870us/step - accuracy: 0.8778 - auc_1: 0.8957 - f1_score: 0.4675 - loss: 22
5.7683 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 219.9762
Epoch 132/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 827us/step - accuracy: 0.8778 - auc_1: 0.8939 - f1_score: 0.4675 - loss: 22
```

```
7.0512 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 219.8150
Epoch 133/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 783us/step - accuracy: 0.8780 - auc_1: 0.8964 - f1_score: 0.4687 - loss: 22
4.8206 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 219.6413
Epoch 134/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 735us/step - accuracy: 0.8778 - auc_1: 0.8967 - f1_score: 0.4675 - loss: 22
4.3064 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 219.4864
Epoch 135/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 703us/step - accuracy: 0.8779 - auc_1: 0.8961 - f1_score: 0.4679 - loss: 22
4.8927 - val_accuracy: 0.8744 - val_auc_1: 0.9024 - val_f1_score: 0.4665 - val_loss: 219.3442
Epoch 136/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 681us/step - accuracy: 0.8778 - auc_1: 0.8965 - f1_score: 0.4675 - loss: 22
4.4295 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 219.1734
Epoch 137/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 740us/step - accuracy: 0.8778 - auc_1: 0.8961 - f1_score: 0.4675 - loss: 22
4.7274 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 219.0006
Epoch 138/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 726us/step - accuracy: 0.8779 - auc_1: 0.8956 - f1_score: 0.4679 - loss: 22
4.4197 - val_accuracy: 0.8744 - val_auc_1: 0.9025 - val_f1_score: 0.4665 - val_loss: 218.8497
Epoch 139/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 658us/step - accuracy: 0.8778 - auc_1: 0.8976 - f1_score: 0.4675 - loss: 22
3.1997 - val_accuracy: 0.8744 - val_auc_1: 0.9026 - val_f1_score: 0.4665 - val_loss: 218.6906
Epoch 140/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 669us/step - accuracy: 0.8778 - auc_1: 0.8971 - f1_score: 0.4675 - loss: 22
3.4738 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 218.5186
Epoch 141/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 686us/step - accuracy: 0.8778 - auc_1: 0.8951 - f1_score: 0.4675 - loss: 22
3.6574 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 218.3763
Epoch 142/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 824us/step - accuracy: 0.8779 - auc_1: 0.8974 - f1_score: 0.4675 - loss: 22
2.5542 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 218.2296
Epoch 143/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 850us/step - accuracy: 0.8778 - auc_1: 0.8958 - f1_score: 0.4675 - loss: 22
3.4530 - val_accuracy: 0.8744 - val_auc_1: 0.9026 - val_f1_score: 0.4665 - val_loss: 218.0891
Epoch 144/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 911us/step - accuracy: 0.8778 - auc_1: 0.8976 - f1_score: 0.4675 - loss: 22
2.4699 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 217.9376
Epoch 145/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 817us/step - accuracy: 0.8778 - auc_1: 0.8983 - f1_score: 0.4675 - loss: 22
1.3305 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 217.7930
Epoch 146/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 682us/step - accuracy: 0.8778 - auc_1: 0.8954 - f1_score: 0.4675 - loss: 22
2.7410 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 217.6420
Epoch 147/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 679us/step - accuracy: 0.8778 - auc_1: 0.8965 - f1_score: 0.4675 - loss: 22
2.6530 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 217.4822
Epoch 148/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 636us/step - accuracy: 0.8778 - auc_1: 0.8943 - f1_score: 0.4675 - loss: 22
3.4030 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 217.3554
Epoch 149/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 750us/step - accuracy: 0.8779 - auc_1: 0.8971 - f1_score: 0.4683 - loss: 22
1.8311 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 217.2051
Epoch 150/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 668us/step - accuracy: 0.8778 - auc_1: 0.8949 - f1_score: 0.4675 - loss: 22
2.7792 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 217.0571
Epoch 151/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 684us/step - accuracy: 0.8778 - auc_1: 0.8987 - f1_score: 0.4675 - loss: 22
0.3582 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 216.8938
Epoch 152/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 653us/step - accuracy: 0.8779 - auc_1: 0.8947 - f1_score: 0.4679 - loss: 22
2.4491 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 216.7499
Epoch 153/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 676us/step - accuracy: 0.8778 - auc_1: 0.8964 - f1_score: 0.4675 - loss: 22
1.2161 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 216.6020
Epoch 154/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 665us/step - accuracy: 0.8779 - auc_1: 0.8958 - f1_score: 0.4679 - loss: 22
1.0316 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 216.4488
Epoch 155/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 675us/step - accuracy: 0.8779 - auc_1: 0.8974 - f1_score: 0.4683 - loss: 22
0.7567 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 216.3130
Epoch 156/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 639us/step - accuracy: 0.8778 - auc_1: 0.9008 - f1_score: 0.4675 - loss: 21
8.3030 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 216.1508
Epoch 157/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 800us/step - accuracy: 0.8778 - auc_1: 0.8982 - f1_score: 0.4675 - loss: 21
9.7637 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 216.0031
Epoch 158/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 762us/step - accuracy: 0.8779 - auc_1: 0.8984 - f1_score: 0.4678 - loss: 21
9.3228 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 215.8490
```

```
Epoch 159/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 812us/step - accuracy: 0.8778 - auc_1: 0.8959 - f1_score: 0.4675 - loss: 22
0.5470 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 215.7042
Epoch 160/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 775us/step - accuracy: 0.8778 - auc_1: 0.8996 - f1_score: 0.4675 - loss: 21
8.4277 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 215.5650
Epoch 161/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 651us/step - accuracy: 0.8778 - auc_1: 0.8956 - f1_score: 0.4675 - loss: 22
0.4581 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 215.4558
Epoch 162/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 722us/step - accuracy: 0.8779 - auc_1: 0.8978 - f1_score: 0.4679 - loss: 21
9.0653 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 215.3147
Epoch 163/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 686us/step - accuracy: 0.8778 - auc_1: 0.8962 - f1_score: 0.4675 - loss: 21
9.8069 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 215.1950
Epoch 164/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 704us/step - accuracy: 0.8778 - auc_1: 0.8969 - f1_score: 0.4675 - loss: 21
9.0839 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 215.0802
Epoch 165/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 722us/step - accuracy: 0.8778 - auc_1: 0.8968 - f1_score: 0.4675 - loss: 21
9.4819 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 214.9447
Epoch 166/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 729us/step - accuracy: 0.8778 - auc_1: 0.8976 - f1_score: 0.4675 - loss: 21
8.4928 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 214.8202
Epoch 167/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 717us/step - accuracy: 0.8778 - auc_1: 0.8983 - f1_score: 0.4675 - loss: 21
7.7543 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 214.6788
Epoch 168/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 704us/step - accuracy: 0.8778 - auc_1: 0.8970 - f1_score: 0.4675 - loss: 21
8.4714 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 214.5632
Epoch 169/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 738us/step - accuracy: 0.8778 - auc_1: 0.8964 - f1_score: 0.4675 - loss: 21
8.6245 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 214.4555
Epoch 170/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8779 - auc_1: 0.8990 - f1_score: 0.4679 - loss: 217.0
827 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 214.3207
Epoch 171/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 931us/step - accuracy: 0.8779 - auc_1: 0.8966 - f1_score: 0.4679 - loss: 21
8.3148 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 214.2292
Epoch 172/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 863us/step - accuracy: 0.8778 - auc_1: 0.8977 - f1_score: 0.4675 - loss: 21
7.5596 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 214.1124
Epoch 173/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 727us/step - accuracy: 0.8778 - auc_1: 0.8993 - f1_score: 0.4675 - loss: 21
6.4114 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 213.9738
Epoch 174/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 720us/step - accuracy: 0.8778 - auc_1: 0.9011 - f1_score: 0.4675 - loss: 21
5.5465 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 213.8448
Epoch 175/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 706us/step - accuracy: 0.8778 - auc_1: 0.8985 - f1_score: 0.4675 - loss: 21
6.7828 - val_accuracy: 0.8744 - val_auc_1: 0.9030 - val_f1_score: 0.4665 - val_loss: 213.7167
Epoch 176/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 722us/step - accuracy: 0.8778 - auc_1: 0.8984 - f1_score: 0.4675 - loss: 21
6.5472 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 213.5884
Epoch 177/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 704us/step - accuracy: 0.8779 - auc_1: 0.8973 - f1_score: 0.4679 - loss: 21
6.9902 - val_accuracy: 0.8744 - val_auc_1: 0.9031 - val_f1_score: 0.4665 - val_loss: 213.4628
Epoch 178/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 732us/step - accuracy: 0.8778 - auc_1: 0.8997 - f1_score: 0.4675 - loss: 21
5.7553 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 213.3493
Epoch 179/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 704us/step - accuracy: 0.8778 - auc_1: 0.8983 - f1_score: 0.4675 - loss: 21
6.2863 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 213.2455
Epoch 180/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 664us/step - accuracy: 0.8779 - auc_1: 0.8997 - f1_score: 0.4679 - loss: 21
5.2988 - val_accuracy: 0.8744 - val_auc_1: 0.9030 - val_f1_score: 0.4665 - val_loss: 213.1220
Epoch 181/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 660us/step - accuracy: 0.8778 - auc_1: 0.8990 - f1_score: 0.4675 - loss: 21
5.3542 - val_accuracy: 0.8744 - val_auc_1: 0.9030 - val_f1_score: 0.4665 - val_loss: 213.0030
Epoch 182/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 670us/step - accuracy: 0.8778 - auc_1: 0.8975 - f1_score: 0.4675 - loss: 21
6.4713 - val_accuracy: 0.8744 - val_auc_1: 0.9030 - val_f1_score: 0.4665 - val_loss: 212.8810
Epoch 183/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 747us/step - accuracy: 0.8779 - auc_1: 0.8982 - f1_score: 0.4679 - loss: 21
5.5240 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 212.7696
Epoch 184/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 805us/step - accuracy: 0.8779 - auc_1: 0.8960 - f1_score: 0.4679 - loss: 21
6.5542 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 212.6675
Epoch 185/300
```

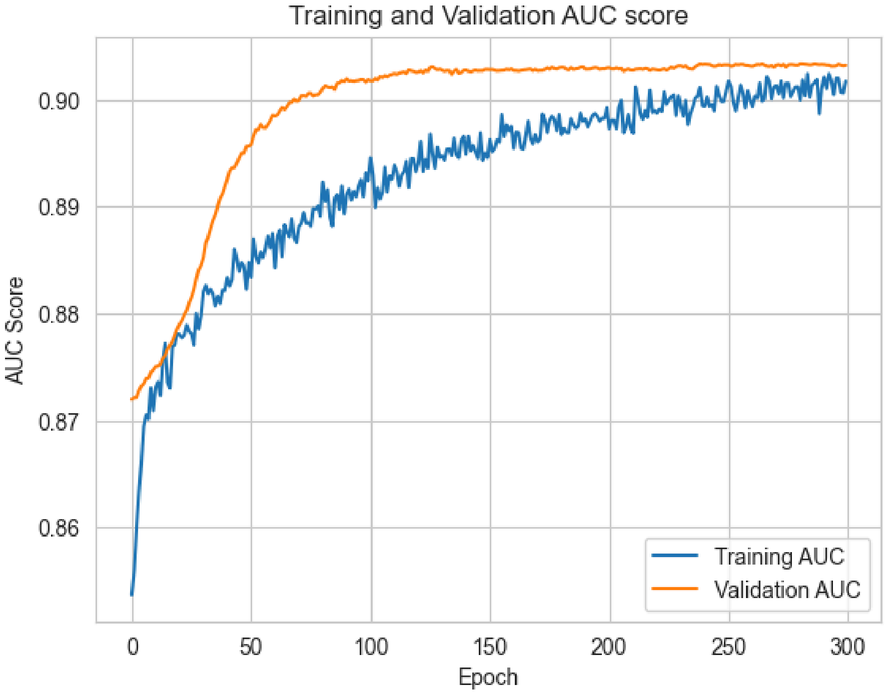
```
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 844us/step - accuracy: 0.8778 - auc_1: 0.8989 - f1_score: 0.4675 - loss: 21
4.9753 - val_accuracy: 0.8744 - val_auc_1: 0.9030 - val_f1_score: 0.4665 - val_loss: 212.5458
Epoch 186/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 790us/step - accuracy: 0.8778 - auc_1: 0.8973 - f1_score: 0.4675 - loss: 21
6.1735 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 212.4464
Epoch 187/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 660us/step - accuracy: 0.8778 - auc_1: 0.8982 - f1_score: 0.4675 - loss: 21
5.3155 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 212.3427
Epoch 188/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 649us/step - accuracy: 0.8778 - auc_1: 0.8986 - f1_score: 0.4675 - loss: 21
4.9396 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 212.2316
Epoch 189/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 679us/step - accuracy: 0.8778 - auc_1: 0.8989 - f1_score: 0.4675 - loss: 21
4.6176 - val_accuracy: 0.8744 - val_auc_1: 0.9031 - val_f1_score: 0.4665 - val_loss: 212.1066
Epoch 190/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 645us/step - accuracy: 0.8778 - auc_1: 0.9010 - f1_score: 0.4675 - loss: 21
3.2942 - val_accuracy: 0.8744 - val_auc_1: 0.9031 - val_f1_score: 0.4665 - val_loss: 211.9870
Epoch 191/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 716us/step - accuracy: 0.8778 - auc_1: 0.8992 - f1_score: 0.4675 - loss: 21
4.6067 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 211.9027
Epoch 192/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 678us/step - accuracy: 0.8778 - auc_1: 0.8985 - f1_score: 0.4675 - loss: 21
4.2854 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 211.7905
Epoch 193/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 732us/step - accuracy: 0.8778 - auc_1: 0.8978 - f1_score: 0.4675 - loss: 21
4.5126 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 211.6754
Epoch 194/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 686us/step - accuracy: 0.8778 - auc_1: 0.8998 - f1_score: 0.4675 - loss: 21
3.5922 - val_accuracy: 0.8744 - val_auc_1: 0.9030 - val_f1_score: 0.4665 - val_loss: 211.5625
Epoch 195/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 677us/step - accuracy: 0.8779 - auc_1: 0.8984 - f1_score: 0.4679 - loss: 21
4.2811 - val_accuracy: 0.8744 - val_auc_1: 0.9030 - val_f1_score: 0.4665 - val_loss: 211.4354
Epoch 196/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 706us/step - accuracy: 0.8778 - auc_1: 0.8994 - f1_score: 0.4675 - loss: 21
3.5934 - val_accuracy: 0.8744 - val_auc_1: 0.9030 - val_f1_score: 0.4665 - val_loss: 211.3269
Epoch 197/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 694us/step - accuracy: 0.8778 - auc_1: 0.8996 - f1_score: 0.4675 - loss: 21
3.1639 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 211.2166
Epoch 198/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 777us/step - accuracy: 0.8778 - auc_1: 0.9005 - f1_score: 0.4675 - loss: 21
2.6048 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 211.1162
Epoch 199/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 836us/step - accuracy: 0.8778 - auc_1: 0.9005 - f1_score: 0.4675 - loss: 21
2.0861 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 211.0186
Epoch 200/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 896us/step - accuracy: 0.8778 - auc_1: 0.8988 - f1_score: 0.4675 - loss: 21
3.1337 - val_accuracy: 0.8744 - val_auc_1: 0.9030 - val_f1_score: 0.4665 - val_loss: 210.9031
Epoch 201/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 842us/step - accuracy: 0.8778 - auc_1: 0.8984 - f1_score: 0.4675 - loss: 21
3.8622 - val_accuracy: 0.8744 - val_auc_1: 0.9030 - val_f1_score: 0.4665 - val_loss: 210.8030
Epoch 202/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 738us/step - accuracy: 0.8778 - auc_1: 0.8988 - f1_score: 0.4675 - loss: 21
2.7587 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 210.6979
Epoch 203/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 702us/step - accuracy: 0.8778 - auc_1: 0.9010 - f1_score: 0.4675 - loss: 21
1.4671 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 210.5835
Epoch 204/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 663us/step - accuracy: 0.8778 - auc_1: 0.8988 - f1_score: 0.4675 - loss: 21
2.3822 - val_accuracy: 0.8744 - val_auc_1: 0.9030 - val_f1_score: 0.4665 - val_loss: 210.4821
Epoch 205/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 695us/step - accuracy: 0.8778 - auc_1: 0.9013 - f1_score: 0.4675 - loss: 21
1.1624 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 210.3602
Epoch 206/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 659us/step - accuracy: 0.8778 - auc_1: 0.9008 - f1_score: 0.4675 - loss: 21
1.4044 - val_accuracy: 0.8744 - val_auc_1: 0.9030 - val_f1_score: 0.4665 - val_loss: 210.2519
Epoch 207/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 690us/step - accuracy: 0.8778 - auc_1: 0.8983 - f1_score: 0.4675 - loss: 21
2.4382 - val_accuracy: 0.8744 - val_auc_1: 0.9027 - val_f1_score: 0.4665 - val_loss: 210.1665
Epoch 208/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 655us/step - accuracy: 0.8779 - auc_1: 0.8984 - f1_score: 0.4679 - loss: 21
2.6429 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 210.0702
Epoch 209/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 678us/step - accuracy: 0.8778 - auc_1: 0.8997 - f1_score: 0.4675 - loss: 21
1.8054 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 209.9598
Epoch 210/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 662us/step - accuracy: 0.8778 - auc_1: 0.8977 - f1_score: 0.4675 - loss: 21
2.1513 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 209.8590
Epoch 211/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 694us/step - accuracy: 0.8778 - auc_1: 0.8978 - f1_score: 0.4675 - loss: 21
```

```
2.3967 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 209.7507
Epoch 212/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 700us/step - accuracy: 0.8778 - auc_1: 0.9018 - f1_score: 0.4675 - loss: 20
9.8564 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 209.6109
Epoch 213/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 821us/step - accuracy: 0.8779 - auc_1: 0.9006 - f1_score: 0.4679 - loss: 21
0.2755 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 209.4993
Epoch 214/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 751us/step - accuracy: 0.8779 - auc_1: 0.8995 - f1_score: 0.4679 - loss: 21
0.9921 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 209.4006
Epoch 215/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 767us/step - accuracy: 0.8778 - auc_1: 0.9004 - f1_score: 0.4675 - loss: 21
0.1337 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 209.3094
Epoch 216/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 763us/step - accuracy: 0.8778 - auc_1: 0.8995 - f1_score: 0.4675 - loss: 21
0.7101 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 209.2149
Epoch 217/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 657us/step - accuracy: 0.8778 - auc_1: 0.8989 - f1_score: 0.4675 - loss: 21
1.1577 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 209.1254
Epoch 218/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 673us/step - accuracy: 0.8778 - auc_1: 0.9009 - f1_score: 0.4675 - loss: 20
9.8326 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 209.0049
Epoch 219/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 659us/step - accuracy: 0.8778 - auc_1: 0.9002 - f1_score: 0.4675 - loss: 21
0.0791 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 208.9089
Epoch 220/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 712us/step - accuracy: 0.8778 - auc_1: 0.8993 - f1_score: 0.4675 - loss: 21
0.2959 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 208.8151
Epoch 221/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 679us/step - accuracy: 0.8778 - auc_1: 0.9001 - f1_score: 0.4675 - loss: 20
9.9283 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 208.7014
Epoch 222/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 711us/step - accuracy: 0.8778 - auc_1: 0.9012 - f1_score: 0.4675 - loss: 20
9.1002 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 208.6060
Epoch 223/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 681us/step - accuracy: 0.8778 - auc_1: 0.9017 - f1_score: 0.4675 - loss: 20
8.6421 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 208.5185
Epoch 224/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 693us/step - accuracy: 0.8778 - auc_1: 0.9028 - f1_score: 0.4675 - loss: 20
8.2277 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 208.4207
Epoch 225/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 696us/step - accuracy: 0.8778 - auc_1: 0.9006 - f1_score: 0.4675 - loss: 20
8.6857 - val_accuracy: 0.8744 - val_auc_1: 0.9028 - val_f1_score: 0.4665 - val_loss: 208.3222
Epoch 226/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 694us/step - accuracy: 0.8778 - auc_1: 0.9001 - f1_score: 0.4675 - loss: 20
9.0662 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 208.2201
Epoch 227/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 658us/step - accuracy: 0.8778 - auc_1: 0.9011 - f1_score: 0.4675 - loss: 20
8.3867 - val_accuracy: 0.8744 - val_auc_1: 0.9031 - val_f1_score: 0.4665 - val_loss: 208.1161
Epoch 228/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 829us/step - accuracy: 0.8778 - auc_1: 0.9004 - f1_score: 0.4675 - loss: 20
9.1249 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 208.0100
Epoch 229/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 802us/step - accuracy: 0.8778 - auc_1: 0.8995 - f1_score: 0.4675 - loss: 20
9.2182 - val_accuracy: 0.8744 - val_auc_1: 0.9031 - val_f1_score: 0.4665 - val_loss: 207.9221
Epoch 230/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 846us/step - accuracy: 0.8778 - auc_1: 0.9031 - f1_score: 0.4675 - loss: 20
7.3110 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 207.8188
Epoch 231/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 818us/step - accuracy: 0.8778 - auc_1: 0.8993 - f1_score: 0.4675 - loss: 20
9.1806 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 207.7227
Epoch 232/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 710us/step - accuracy: 0.8778 - auc_1: 0.8989 - f1_score: 0.4675 - loss: 20
9.1755 - val_accuracy: 0.8744 - val_auc_1: 0.9031 - val_f1_score: 0.4665 - val_loss: 207.6258
Epoch 233/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 779us/step - accuracy: 0.8778 - auc_1: 0.9004 - f1_score: 0.4675 - loss: 20
8.0528 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 207.5131
Epoch 234/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 704us/step - accuracy: 0.8778 - auc_1: 0.8982 - f1_score: 0.4675 - loss: 20
9.3469 - val_accuracy: 0.8744 - val_auc_1: 0.9030 - val_f1_score: 0.4665 - val_loss: 207.4317
Epoch 235/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 728us/step - accuracy: 0.8778 - auc_1: 0.8992 - f1_score: 0.4675 - loss: 20
8.8695 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 207.3442
Epoch 236/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 713us/step - accuracy: 0.8778 - auc_1: 0.8990 - f1_score: 0.4675 - loss: 20
8.8219 - val_accuracy: 0.8744 - val_auc_1: 0.9029 - val_f1_score: 0.4665 - val_loss: 207.2454
Epoch 237/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 682us/step - accuracy: 0.8778 - auc_1: 0.9016 - f1_score: 0.4675 - loss: 20
7.2224 - val_accuracy: 0.8744 - val_auc_1: 0.9031 - val_f1_score: 0.4665 - val_loss: 207.1370
```

```
Epoch 238/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 693us/step - accuracy: 0.8778 - auc_1: 0.9019 - f1_score: 0.4675 - loss: 20
6.9764 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 207.0307
Epoch 239/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 669us/step - accuracy: 0.8778 - auc_1: 0.9022 - f1_score: 0.4675 - loss: 20
6.7161 - val_accuracy: 0.8744 - val_auc_1: 0.9034 - val_f1_score: 0.4665 - val_loss: 206.9270
Epoch 240/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 732us/step - accuracy: 0.8778 - auc_1: 0.9009 - f1_score: 0.4675 - loss: 20
6.8746 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 206.8362
Epoch 241/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 742us/step - accuracy: 0.8778 - auc_1: 0.9011 - f1_score: 0.4675 - loss: 20
7.1205 - val_accuracy: 0.8744 - val_auc_1: 0.9034 - val_f1_score: 0.4665 - val_loss: 206.7483
Epoch 242/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 807us/step - accuracy: 0.8778 - auc_1: 0.9007 - f1_score: 0.4675 - loss: 20
7.3440 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 206.6562
Epoch 243/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 863us/step - accuracy: 0.8778 - auc_1: 0.9005 - f1_score: 0.4675 - loss: 20
7.3408 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 206.5594
Epoch 244/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 759us/step - accuracy: 0.8778 - auc_1: 0.8985 - f1_score: 0.4675 - loss: 20
7.8175 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 206.4693
Epoch 245/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 693us/step - accuracy: 0.8778 - auc_1: 0.8992 - f1_score: 0.4675 - loss: 20
7.7457 - val_accuracy: 0.8744 - val_auc_1: 0.9031 - val_f1_score: 0.4665 - val_loss: 206.3945
Epoch 246/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 669us/step - accuracy: 0.8778 - auc_1: 0.9021 - f1_score: 0.4675 - loss: 20
6.3295 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 206.2984
Epoch 247/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 696us/step - accuracy: 0.8778 - auc_1: 0.9019 - f1_score: 0.4675 - loss: 20
6.1877 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 206.2098
Epoch 248/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 656us/step - accuracy: 0.8778 - auc_1: 0.9011 - f1_score: 0.4675 - loss: 20
6.5966 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 206.1174
Epoch 249/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 678us/step - accuracy: 0.8778 - auc_1: 0.9017 - f1_score: 0.4675 - loss: 20
6.3457 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 206.0270
Epoch 250/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 661us/step - accuracy: 0.8778 - auc_1: 0.9022 - f1_score: 0.4675 - loss: 20
5.6619 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 205.9271
Epoch 251/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 679us/step - accuracy: 0.8778 - auc_1: 0.9030 - f1_score: 0.4675 - loss: 20
5.1954 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 205.8390
Epoch 252/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 658us/step - accuracy: 0.8778 - auc_1: 0.9016 - f1_score: 0.4675 - loss: 20
5.5876 - val_accuracy: 0.8744 - val_auc_1: 0.9031 - val_f1_score: 0.4665 - val_loss: 205.7489
Epoch 253/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 807us/step - accuracy: 0.8778 - auc_1: 0.9015 - f1_score: 0.4675 - loss: 20
5.5760 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 205.6696
Epoch 254/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 793us/step - accuracy: 0.8778 - auc_1: 0.9001 - f1_score: 0.4675 - loss: 20
6.3465 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 205.6000
Epoch 255/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 841us/step - accuracy: 0.8778 - auc_1: 0.9018 - f1_score: 0.4675 - loss: 20
5.6091 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 205.4923
Epoch 256/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 780us/step - accuracy: 0.8778 - auc_1: 0.9023 - f1_score: 0.4675 - loss: 20
4.7856 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 205.3956
Epoch 257/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 724us/step - accuracy: 0.8778 - auc_1: 0.9017 - f1_score: 0.4675 - loss: 20
5.2777 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 205.3179
Epoch 258/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 760us/step - accuracy: 0.8778 - auc_1: 0.8996 - f1_score: 0.4675 - loss: 20
6.0054 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 205.2310
Epoch 259/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 690us/step - accuracy: 0.8778 - auc_1: 0.9025 - f1_score: 0.4675 - loss: 20
4.7025 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 205.1477
Epoch 260/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 727us/step - accuracy: 0.8778 - auc_1: 0.8998 - f1_score: 0.4675 - loss: 20
6.0380 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 205.0644
Epoch 261/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 768us/step - accuracy: 0.8778 - auc_1: 0.9003 - f1_score: 0.4675 - loss: 20
5.5366 - val_accuracy: 0.8744 - val_auc_1: 0.9031 - val_f1_score: 0.4665 - val_loss: 204.9789
Epoch 262/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 698us/step - accuracy: 0.8778 - auc_1: 0.9015 - f1_score: 0.4675 - loss: 20
4.7909 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 204.8846
Epoch 263/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 649us/step - accuracy: 0.8778 - auc_1: 0.9020 - f1_score: 0.4675 - loss: 20
4.4743 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 204.8107
Epoch 264/300
```

```
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 679us/step - accuracy: 0.8778 - auc_1: 0.9029 - f1_score: 0.4675 - loss: 20
3.9232 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 204.7190
Epoch 265/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 662us/step - accuracy: 0.8778 - auc_1: 0.9020 - f1_score: 0.4675 - loss: 20
4.1087 - val_accuracy: 0.8744 - val_auc_1: 0.9034 - val_f1_score: 0.4665 - val_loss: 204.6296
Epoch 266/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 850us/step - accuracy: 0.8778 - auc_1: 0.8994 - f1_score: 0.4675 - loss: 20
5.0992 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 204.5428
Epoch 267/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 806us/step - accuracy: 0.8778 - auc_1: 0.9039 - f1_score: 0.4675 - loss: 20
3.5257 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 204.4396
Epoch 268/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 771us/step - accuracy: 0.8778 - auc_1: 0.9019 - f1_score: 0.4675 - loss: 20
4.1063 - val_accuracy: 0.8744 - val_auc_1: 0.9031 - val_f1_score: 0.4665 - val_loss: 204.3540
Epoch 269/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 805us/step - accuracy: 0.8778 - auc_1: 0.9007 - f1_score: 0.4675 - loss: 20
4.3012 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 204.2626
Epoch 270/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 720us/step - accuracy: 0.8778 - auc_1: 0.9018 - f1_score: 0.4675 - loss: 20
3.7831 - val_accuracy: 0.8744 - val_auc_1: 0.9031 - val_f1_score: 0.4665 - val_loss: 204.1726
Epoch 271/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 719us/step - accuracy: 0.8778 - auc_1: 0.9019 - f1_score: 0.4675 - loss: 20
3.8212 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 204.0948
Epoch 272/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 683us/step - accuracy: 0.8778 - auc_1: 0.9025 - f1_score: 0.4675 - loss: 20
3.5469 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 204.0101
Epoch 273/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 738us/step - accuracy: 0.8778 - auc_1: 0.9029 - f1_score: 0.4675 - loss: 20
2.7610 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 203.9174
Epoch 274/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 770us/step - accuracy: 0.8778 - auc_1: 0.9014 - f1_score: 0.4675 - loss: 20
3.6824 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 203.8495
Epoch 275/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 707us/step - accuracy: 0.8778 - auc_1: 0.9029 - f1_score: 0.4675 - loss: 20
2.9781 - val_accuracy: 0.8744 - val_auc_1: 0.9031 - val_f1_score: 0.4665 - val_loss: 203.7655
Epoch 276/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 699us/step - accuracy: 0.8778 - auc_1: 0.9019 - f1_score: 0.4675 - loss: 20
3.0142 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 203.6753
Epoch 277/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 690us/step - accuracy: 0.8778 - auc_1: 0.9014 - f1_score: 0.4675 - loss: 20
3.2569 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 203.6164
Epoch 278/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 858us/step - accuracy: 0.8778 - auc_1: 0.9009 - f1_score: 0.4675 - loss: 20
3.4170 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 203.5439
Epoch 279/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 749us/step - accuracy: 0.8778 - auc_1: 0.9031 - f1_score: 0.4675 - loss: 20
2.3165 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 203.4564
Epoch 280/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 920us/step - accuracy: 0.8778 - auc_1: 0.9031 - f1_score: 0.4675 - loss: 20
2.3595 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 203.3880
Epoch 281/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 737us/step - accuracy: 0.8778 - auc_1: 0.9009 - f1_score: 0.4675 - loss: 20
3.3910 - val_accuracy: 0.8744 - val_auc_1: 0.9034 - val_f1_score: 0.4665 - val_loss: 203.3309
Epoch 282/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 663us/step - accuracy: 0.8778 - auc_1: 0.9018 - f1_score: 0.4675 - loss: 20
2.6034 - val_accuracy: 0.8744 - val_auc_1: 0.9034 - val_f1_score: 0.4665 - val_loss: 203.2600
Epoch 283/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 713us/step - accuracy: 0.8778 - auc_1: 0.9014 - f1_score: 0.4675 - loss: 20
2.9577 - val_accuracy: 0.8744 - val_auc_1: 0.9034 - val_f1_score: 0.4665 - val_loss: 203.1984
Epoch 284/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 721us/step - accuracy: 0.8778 - auc_1: 0.9028 - f1_score: 0.4675 - loss: 20
2.3462 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 203.1224
Epoch 285/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 695us/step - accuracy: 0.8778 - auc_1: 0.9012 - f1_score: 0.4675 - loss: 20
2.5495 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 203.0687
Epoch 286/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 665us/step - accuracy: 0.8778 - auc_1: 0.9020 - f1_score: 0.4675 - loss: 20
2.4036 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 202.9883
Epoch 287/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 675us/step - accuracy: 0.8778 - auc_1: 0.9026 - f1_score: 0.4675 - loss: 20
2.3551 - val_accuracy: 0.8744 - val_auc_1: 0.9034 - val_f1_score: 0.4665 - val_loss: 202.9214
Epoch 288/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 666us/step - accuracy: 0.8778 - auc_1: 0.9028 - f1_score: 0.4675 - loss: 20
1.8435 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 202.8567
Epoch 289/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 682us/step - accuracy: 0.8778 - auc_1: 0.9004 - f1_score: 0.4675 - loss: 20
3.0456 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 202.8162
Epoch 290/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 669us/step - accuracy: 0.8778 - auc_1: 0.9014 - f1_score: 0.4675 - loss: 20
```

```
2.1904 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 202.7631
Epoch 291/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 792us/step - accuracy: 0.8778 - auc_1: 0.9029 - f1_score: 0.4675 - loss: 20
1.6490 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 202.6948
Epoch 292/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 816us/step - accuracy: 0.8778 - auc_1: 0.9005 - f1_score: 0.4675 - loss: 20
2.7788 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 202.6334
Epoch 293/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 921us/step - accuracy: 0.8778 - auc_1: 0.9028 - f1_score: 0.4675 - loss: 20
1.6376 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 202.5527
Epoch 294/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 910us/step - accuracy: 0.8778 - auc_1: 0.9037 - f1_score: 0.4675 - loss: 20
1.0339 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 202.5062
Epoch 295/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 728us/step - accuracy: 0.8778 - auc_1: 0.8999 - f1_score: 0.4675 - loss: 20
2.3806 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 202.4417
Epoch 296/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 719us/step - accuracy: 0.8778 - auc_1: 0.9037 - f1_score: 0.4675 - loss: 20
1.0287 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 202.3751
Epoch 297/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 700us/step - accuracy: 0.8778 - auc_1: 0.9016 - f1_score: 0.4675 - loss: 20
1.8615 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 202.3076
Epoch 298/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 714us/step - accuracy: 0.8778 - auc_1: 0.9000 - f1_score: 0.4675 - loss: 20
2.4212 - val_accuracy: 0.8744 - val_auc_1: 0.9033 - val_f1_score: 0.4665 - val_loss: 202.2474
Epoch 299/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 698us/step - accuracy: 0.8778 - auc_1: 0.9017 - f1_score: 0.4675 - loss: 20
1.6089 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 202.2003
Epoch 300/300
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 689us/step - accuracy: 0.8778 - auc_1: 0.9030 - f1_score: 0.4675 - loss: 20
0.9285 - val_accuracy: 0.8744 - val_auc_1: 0.9032 - val_f1_score: 0.4665 - val_loss: 202.1338
84/84 ━━━━━━━━━━━━━━━━━━━━ 0s 532us/step - accuracy: 0.8680 - auc_1: 0.8985 - f1_score: 0.4646 - loss: 208.0
001
```



In [36]:
```python
# update_summary_score(summary_df,
#                 'Softmax NN',
#                 simple_softmax_model.evaluate(X_train, y_train)[2],
#                 scores[2],
#                 simple_softmax_model.evaluate(X_test, y_test)[2])
               # simple_softmax_model.evaluate(X_train, y_train)[1],
               # scores[1],
               # simple_softmax_model.evaluate(X_test, y_test)[1],
               # simple_softmax_model.evaluate(X_train, y_train)[3].numpy()[1],
               # scores[3].numpy()[1],
               # simple_softmax_model.evaluate(X_test, y_test)[3].numpy()[1])
update_summary(summary_df,
               'Softmax NN',
               y_train_sigmoid,
               simple_softmax_model.predict(X_train),
               y_val_sigmoid,
               simple_softmax_model.predict(X_val),
               y_test_sigmoid,
```

```
                simple_softmax_model.predict(X_test))
summary_df
```

```
753/753 ━━━━━━━━━━━━━━━━━━━━ 0s 371us/step
84/84 ━━━━━━━━━━━━━━━━━━━━ 0s 318us/step
93/93 ━━━━━━━━━━━━━━━━━━━━ 0s 321us/step
```

Out[36]:

	Model	Train AUC	Val AUC	Test AUC
0	Logit as Benchmark	0.6707	0.6637	0.6637
1	LASSO Logit	0.6835	0.6791	0.6791
2	Random Forest CV	0.9686	0.7086	0.7086
3	GBM CV	0.8193	0.7133	0.7133
4	Sigmoid NN	0.6578	0.6321	0.6611
5	Softmax NN	0.6555	0.6314	0.6611

The softmax neural network performs comparable to the sigmoid neural network in predicting for popularity with the test set. However, the performance on the validation set is worse, which raise some concern how it performs on a different unseen dataset.

Neural network with SMOTE

I also experiment with oversampling since the poplar articles are rarer than the non-popular ones. The hypothesis is the model might nto be able to capture the patterns of the rare cases, hence oversampling the positive cases might help. This neural network share the same configuration with the sigmoid neural network and undergo the same optimizations, except for the adjustment in the oversampling strategy.

In [37]:
```
from imblearn.over_sampling import SMOTE, RandomOverSampler

smt = RandomOverSampler(sampling_strategy=0.5, random_state=prng)
X_smote, y_smote = smt.fit_resample(X_train, y_train_sigmoid)
# y_smote = to_categorical(y_smote, num_classes=num_classes)
```

In [38]:
```
def custom_loss_smote(y_true, y_pred):
    # Define weights
    false_positive_weight = 1.0
    false_negative_weight = 10000.0

    # Calculate binary cross entropy
    bce = tf.keras.losses.BinaryCrossentropy()

    # Calculate loss
    loss = bce(y_true, y_pred)

    # Calculate weighted loss
    weighted_loss = tf.where(tf.greater(y_true, y_pred), false_negative_weight * loss, false_positive_weigh

    return tf.reduce_mean(weighted_loss)
```

In [40]:
```
from sklearn.utils import compute_class_weight

# Build the simple fully connected single hidden layer network model
# smote_model = Sequential([
#     Input(shape=X_train.shape[1:]),
#     # Dense(22, activation='relu', kernel_regularizer=l1(0.5), kernel_initializer='glorot_normal'),
#     # Dropout(0.4),
#     Dense(22, activation='relu', kernel_regularizer=l1(0.5)),
#     Dropout(0.7),
#     # Dense(1, activation='sigmoid', kernel_regularizer=l1(0.5), kernel_initializer='glorot_normal')
#     Dense(1, activation='sigmoid', kernel_regularizer=l1(0.5), kernel_initializer='glorot_normal')
# ])
smote_model = Sequential([
    Input(shape=X_train.shape[1:]),
    # Normalization(axis=-1),
    Dense(256, activation='relu', kernel_regularizer=l1(0.5)),
    Dropout(0.4),
    Dense(1, activation='sigmoid', kernel_regularizer=l1(0.5), kernel_initializer='glorot_normal')
])

# Compile the model
opt = Adam(learning_rate=0.00001)
smote_model.compile(loss=custom_loss_smote, optimizer=opt, metrics=[AUC(), 'accuracy', F1Score()])
```

```python
# Fit the model
keras.utils.set_random_seed(42)  # for reproducibility
# smote_history = smote_model.fit(X_smote, y_smote, validation_data=(X_val, y_val_sigmoid), epochs=500, bat
smote_history = smote_model.fit(X_smote, y_smote, validation_data=(X_val, y_val_sigmoid), epochs=200, batch

plot_history(smote_history.history)
```

```
Epoch 1/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 894us/step - accuracy: 0.6478 - auc_3: 0.5224 - f1_score: 0.4939 - loss: 256
5.2551 - val_accuracy: 0.8726 - val_auc_3: 0.6188 - val_f1_score: 0.2231 - val_loss: 1042.6143
Epoch 2/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 829us/step - accuracy: 0.6739 - auc_3: 0.5809 - f1_score: 0.4939 - loss: 249
0.7603 - val_accuracy: 0.8636 - val_auc_3: 0.6284 - val_f1_score: 0.2231 - val_loss: 1028.4751
Epoch 3/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 742us/step - accuracy: 0.6728 - auc_3: 0.6013 - f1_score: 0.4939 - loss: 246
2.0571 - val_accuracy: 0.8498 - val_auc_3: 0.6335 - val_f1_score: 0.2231 - val_loss: 1014.7947
Epoch 4/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 680us/step - accuracy: 0.6708 - auc_3: 0.6183 - f1_score: 0.4939 - loss: 243
5.5093 - val_accuracy: 0.8363 - val_auc_3: 0.6353 - val_f1_score: 0.2231 - val_loss: 1002.3637
Epoch 5/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 702us/step - accuracy: 0.6752 - auc_3: 0.6237 - f1_score: 0.4939 - loss: 241
8.4424 - val_accuracy: 0.8341 - val_auc_3: 0.6362 - val_f1_score: 0.2231 - val_loss: 995.2867
Epoch 6/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 693us/step - accuracy: 0.6733 - auc_3: 0.6306 - f1_score: 0.4939 - loss: 240
0.5874 - val_accuracy: 0.8296 - val_auc_3: 0.6360 - val_f1_score: 0.2231 - val_loss: 985.0209
Epoch 7/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 690us/step - accuracy: 0.6761 - auc_3: 0.6357 - f1_score: 0.4939 - loss: 238
6.1846 - val_accuracy: 0.8285 - val_auc_3: 0.6365 - val_f1_score: 0.2231 - val_loss: 976.4055
Epoch 8/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 674us/step - accuracy: 0.6779 - auc_3: 0.6407 - f1_score: 0.4939 - loss: 237
3.9929 - val_accuracy: 0.8262 - val_auc_3: 0.6368 - val_f1_score: 0.2231 - val_loss: 968.0669
Epoch 9/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 695us/step - accuracy: 0.6808 - auc_3: 0.6404 - f1_score: 0.4939 - loss: 236
6.0818 - val_accuracy: 0.8236 - val_auc_3: 0.6363 - val_f1_score: 0.2231 - val_loss: 962.6808
Epoch 10/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 739us/step - accuracy: 0.6809 - auc_3: 0.6471 - f1_score: 0.4939 - loss: 234
8.3066 - val_accuracy: 0.8232 - val_auc_3: 0.6365 - val_f1_score: 0.2231 - val_loss: 953.6549
Epoch 11/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 843us/step - accuracy: 0.6791 - auc_3: 0.6486 - f1_score: 0.4939 - loss: 234
1.5945 - val_accuracy: 0.8229 - val_auc_3: 0.6361 - val_f1_score: 0.2231 - val_loss: 946.8198
Epoch 12/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 691us/step - accuracy: 0.6829 - auc_3: 0.6535 - f1_score: 0.4939 - loss: 232
5.3489 - val_accuracy: 0.8229 - val_auc_3: 0.6366 - val_f1_score: 0.2231 - val_loss: 942.0395
Epoch 13/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 704us/step - accuracy: 0.6800 - auc_3: 0.6545 - f1_score: 0.4939 - loss: 231
8.3796 - val_accuracy: 0.8225 - val_auc_3: 0.6367 - val_f1_score: 0.2231 - val_loss: 935.9697
Epoch 14/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 700us/step - accuracy: 0.6808 - auc_3: 0.6494 - f1_score: 0.4939 - loss: 232
2.1477 - val_accuracy: 0.8203 - val_auc_3: 0.6368 - val_f1_score: 0.2231 - val_loss: 930.9672
Epoch 15/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 709us/step - accuracy: 0.6781 - auc_3: 0.6508 - f1_score: 0.4939 - loss: 231
4.0659 - val_accuracy: 0.8203 - val_auc_3: 0.6370 - val_f1_score: 0.2231 - val_loss: 925.4462
Epoch 16/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 688us/step - accuracy: 0.6805 - auc_3: 0.6510 - f1_score: 0.4939 - loss: 231
0.2319 - val_accuracy: 0.8191 - val_auc_3: 0.6366 - val_f1_score: 0.2231 - val_loss: 922.4958
Epoch 17/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 688us/step - accuracy: 0.6804 - auc_3: 0.6547 - f1_score: 0.4939 - loss: 230
1.2209 - val_accuracy: 0.8173 - val_auc_3: 0.6369 - val_f1_score: 0.2231 - val_loss: 919.5569
Epoch 18/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 720us/step - accuracy: 0.6818 - auc_3: 0.6541 - f1_score: 0.4939 - loss: 229
8.1094 - val_accuracy: 0.8195 - val_auc_3: 0.6369 - val_f1_score: 0.2231 - val_loss: 915.0072
Epoch 19/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 857us/step - accuracy: 0.6808 - auc_3: 0.6587 - f1_score: 0.4939 - loss: 228
8.4224 - val_accuracy: 0.8191 - val_auc_3: 0.6370 - val_f1_score: 0.2231 - val_loss: 911.1318
Epoch 20/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 711us/step - accuracy: 0.6833 - auc_3: 0.6572 - f1_score: 0.4939 - loss: 228
5.6956 - val_accuracy: 0.8206 - val_auc_3: 0.6372 - val_f1_score: 0.2231 - val_loss: 907.3270
Epoch 21/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 681us/step - accuracy: 0.6805 - auc_3: 0.6620 - f1_score: 0.4939 - loss: 227
7.0833 - val_accuracy: 0.8203 - val_auc_3: 0.6374 - val_f1_score: 0.2231 - val_loss: 903.8914
Epoch 22/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 691us/step - accuracy: 0.6823 - auc_3: 0.6586 - f1_score: 0.4939 - loss: 227
5.2319 - val_accuracy: 0.8199 - val_auc_3: 0.6375 - val_f1_score: 0.2231 - val_loss: 900.3234
Epoch 23/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 736us/step - accuracy: 0.6849 - auc_3: 0.6612 - f1_score: 0.4939 - loss: 226
7.8237 - val_accuracy: 0.8214 - val_auc_3: 0.6375 - val_f1_score: 0.2231 - val_loss: 896.6500
Epoch 24/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 703us/step - accuracy: 0.6832 - auc_3: 0.6623 - f1_score: 0.4939 - loss: 226
5.7454 - val_accuracy: 0.8188 - val_auc_3: 0.6377 - val_f1_score: 0.2231 - val_loss: 895.4241
Epoch 25/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 675us/step - accuracy: 0.6827 - auc_3: 0.6600 - f1_score: 0.4939 - loss: 226
4.5203 - val_accuracy: 0.8217 - val_auc_3: 0.6375 - val_f1_score: 0.2231 - val_loss: 890.4979
Epoch 26/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 761us/step - accuracy: 0.6825 - auc_3: 0.6580 - f1_score: 0.4939 - loss: 226
3.0469 - val_accuracy: 0.8210 - val_auc_3: 0.6371 - val_f1_score: 0.2231 - val_loss: 887.8550
Epoch 27/200
```

```
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 888us/step - accuracy: 0.6854 - auc_3: 0.6638 - f1_score: 0.4939 - loss: 225
3.3777 - val_accuracy: 0.8206 - val_auc_3: 0.6382 - val_f1_score: 0.2231 - val_loss: 884.6188
Epoch 28/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 705us/step - accuracy: 0.6824 - auc_3: 0.6632 - f1_score: 0.4939 - loss: 225
1.8613 - val_accuracy: 0.8191 - val_auc_3: 0.6380 - val_f1_score: 0.2231 - val_loss: 882.1913
Epoch 29/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 753us/step - accuracy: 0.6837 - auc_3: 0.6623 - f1_score: 0.4939 - loss: 225
0.4260 - val_accuracy: 0.8184 - val_auc_3: 0.6380 - val_f1_score: 0.2231 - val_loss: 879.7242
Epoch 30/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 798us/step - accuracy: 0.6825 - auc_3: 0.6640 - f1_score: 0.4939 - loss: 224
6.7422 - val_accuracy: 0.8184 - val_auc_3: 0.6379 - val_f1_score: 0.2231 - val_loss: 876.7336
Epoch 31/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 704us/step - accuracy: 0.6847 - auc_3: 0.6634 - f1_score: 0.4939 - loss: 224
3.4526 - val_accuracy: 0.8184 - val_auc_3: 0.6380 - val_f1_score: 0.2231 - val_loss: 874.1961
Epoch 32/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 684us/step - accuracy: 0.6846 - auc_3: 0.6651 - f1_score: 0.4939 - loss: 223
7.8982 - val_accuracy: 0.8191 - val_auc_3: 0.6385 - val_f1_score: 0.2231 - val_loss: 872.2775
Epoch 33/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 676us/step - accuracy: 0.6865 - auc_3: 0.6621 - f1_score: 0.4939 - loss: 223
9.0540 - val_accuracy: 0.8173 - val_auc_3: 0.6385 - val_f1_score: 0.2231 - val_loss: 869.5399
Epoch 34/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 852us/step - accuracy: 0.6886 - auc_3: 0.6679 - f1_score: 0.4939 - loss: 222
8.4021 - val_accuracy: 0.8176 - val_auc_3: 0.6390 - val_f1_score: 0.2231 - val_loss: 868.0243
Epoch 35/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 788us/step - accuracy: 0.6875 - auc_3: 0.6647 - f1_score: 0.4939 - loss: 222
9.9585 - val_accuracy: 0.8180 - val_auc_3: 0.6390 - val_f1_score: 0.2231 - val_loss: 865.5854
Epoch 36/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 723us/step - accuracy: 0.6831 - auc_3: 0.6647 - f1_score: 0.4939 - loss: 223
0.2002 - val_accuracy: 0.8191 - val_auc_3: 0.6388 - val_f1_score: 0.2231 - val_loss: 862.7111
Epoch 37/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 697us/step - accuracy: 0.6848 - auc_3: 0.6652 - f1_score: 0.4939 - loss: 222
6.6157 - val_accuracy: 0.8184 - val_auc_3: 0.6391 - val_f1_score: 0.2231 - val_loss: 861.0197
Epoch 38/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 683us/step - accuracy: 0.6860 - auc_3: 0.6664 - f1_score: 0.4939 - loss: 222
3.0776 - val_accuracy: 0.8176 - val_auc_3: 0.6396 - val_f1_score: 0.2231 - val_loss: 859.3624
Epoch 39/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 693us/step - accuracy: 0.6865 - auc_3: 0.6669 - f1_score: 0.4939 - loss: 222
1.5420 - val_accuracy: 0.8188 - val_auc_3: 0.6399 - val_f1_score: 0.2231 - val_loss: 856.6826
Epoch 40/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 674us/step - accuracy: 0.6855 - auc_3: 0.6671 - f1_score: 0.4939 - loss: 221
8.7739 - val_accuracy: 0.8184 - val_auc_3: 0.6396 - val_f1_score: 0.2231 - val_loss: 854.5861
Epoch 41/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 738us/step - accuracy: 0.6874 - auc_3: 0.6674 - f1_score: 0.4939 - loss: 221
5.0764 - val_accuracy: 0.8180 - val_auc_3: 0.6394 - val_f1_score: 0.2231 - val_loss: 852.8400
Epoch 42/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 891us/step - accuracy: 0.6872 - auc_3: 0.6691 - f1_score: 0.4939 - loss: 221
1.9751 - val_accuracy: 0.8180 - val_auc_3: 0.6403 - val_f1_score: 0.2231 - val_loss: 851.5481
Epoch 43/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 775us/step - accuracy: 0.6870 - auc_3: 0.6664 - f1_score: 0.4939 - loss: 221
1.2058 - val_accuracy: 0.8180 - val_auc_3: 0.6403 - val_f1_score: 0.2231 - val_loss: 849.6806
Epoch 44/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 713us/step - accuracy: 0.6878 - auc_3: 0.6675 - f1_score: 0.4939 - loss: 221
0.9792 - val_accuracy: 0.8180 - val_auc_3: 0.6402 - val_f1_score: 0.2231 - val_loss: 848.4230
Epoch 45/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 724us/step - accuracy: 0.6843 - auc_3: 0.6676 - f1_score: 0.4939 - loss: 220
9.0640 - val_accuracy: 0.8176 - val_auc_3: 0.6406 - val_f1_score: 0.2231 - val_loss: 846.3737
Epoch 46/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 700us/step - accuracy: 0.6879 - auc_3: 0.6664 - f1_score: 0.4939 - loss: 220
8.5942 - val_accuracy: 0.8188 - val_auc_3: 0.6408 - val_f1_score: 0.2231 - val_loss: 843.8352
Epoch 47/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 684us/step - accuracy: 0.6891 - auc_3: 0.6703 - f1_score: 0.4939 - loss: 220
1.0637 - val_accuracy: 0.8191 - val_auc_3: 0.6407 - val_f1_score: 0.2231 - val_loss: 842.2385
Epoch 48/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 699us/step - accuracy: 0.6864 - auc_3: 0.6692 - f1_score: 0.4939 - loss: 220
1.6548 - val_accuracy: 0.8191 - val_auc_3: 0.6409 - val_f1_score: 0.2231 - val_loss: 840.4570
Epoch 49/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 762us/step - accuracy: 0.6888 - auc_3: 0.6697 - f1_score: 0.4939 - loss: 219
8.3218 - val_accuracy: 0.8191 - val_auc_3: 0.6411 - val_f1_score: 0.2231 - val_loss: 838.8671
Epoch 50/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 854us/step - accuracy: 0.6866 - auc_3: 0.6659 - f1_score: 0.4939 - loss: 220
1.5876 - val_accuracy: 0.8173 - val_auc_3: 0.6410 - val_f1_score: 0.2231 - val_loss: 838.4476
Epoch 51/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.6877 - auc_3: 0.6697 - f1_score: 0.4939 - loss: 2194.
4158 - val_accuracy: 0.8188 - val_auc_3: 0.6413 - val_f1_score: 0.2231 - val_loss: 836.0755
Epoch 52/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 704us/step - accuracy: 0.6854 - auc_3: 0.6705 - f1_score: 0.4939 - loss: 219
3.8276 - val_accuracy: 0.8188 - val_auc_3: 0.6413 - val_f1_score: 0.2231 - val_loss: 834.4206
Epoch 53/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 716us/step - accuracy: 0.6840 - auc_3: 0.6695 - f1_score: 0.4939 - loss: 219
```

```
3.7175 - val_accuracy: 0.8191 - val_auc_3: 0.6412 - val_f1_score: 0.2231 - val_loss: 833.1984
Epoch 54/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 756us/step - accuracy: 0.6850 - auc_3: 0.6694 - f1_score: 0.4939 - loss: 219
1.3376 - val_accuracy: 0.8191 - val_auc_3: 0.6416 - val_f1_score: 0.2231 - val_loss: 831.2665
Epoch 55/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 903us/step - accuracy: 0.6860 - auc_3: 0.6706 - f1_score: 0.4939 - loss: 218
8.7773 - val_accuracy: 0.8195 - val_auc_3: 0.6420 - val_f1_score: 0.2231 - val_loss: 829.9159
Epoch 56/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 751us/step - accuracy: 0.6869 - auc_3: 0.6694 - f1_score: 0.4939 - loss: 218
8.0515 - val_accuracy: 0.8191 - val_auc_3: 0.6422 - val_f1_score: 0.2231 - val_loss: 828.9991
Epoch 57/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 752us/step - accuracy: 0.6904 - auc_3: 0.6723 - f1_score: 0.4939 - loss: 218
2.4934 - val_accuracy: 0.8195 - val_auc_3: 0.6423 - val_f1_score: 0.2231 - val_loss: 826.5319
Epoch 58/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 735us/step - accuracy: 0.6863 - auc_3: 0.6687 - f1_score: 0.4939 - loss: 218
6.1597 - val_accuracy: 0.8195 - val_auc_3: 0.6423 - val_f1_score: 0.2231 - val_loss: 826.0394
Epoch 59/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 737us/step - accuracy: 0.6860 - auc_3: 0.6717 - f1_score: 0.4939 - loss: 218
1.3845 - val_accuracy: 0.8199 - val_auc_3: 0.6427 - val_f1_score: 0.2231 - val_loss: 824.4956
Epoch 60/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 804us/step - accuracy: 0.6868 - auc_3: 0.6730 - f1_score: 0.4939 - loss: 217
7.5901 - val_accuracy: 0.8203 - val_auc_3: 0.6426 - val_f1_score: 0.2231 - val_loss: 823.1273
Epoch 61/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 880us/step - accuracy: 0.6855 - auc_3: 0.6712 - f1_score: 0.4939 - loss: 218
0.2041 - val_accuracy: 0.8199 - val_auc_3: 0.6431 - val_f1_score: 0.2231 - val_loss: 821.6808
Epoch 62/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 723us/step - accuracy: 0.6874 - auc_3: 0.6727 - f1_score: 0.4939 - loss: 217
6.2090 - val_accuracy: 0.8199 - val_auc_3: 0.6430 - val_f1_score: 0.2231 - val_loss: 820.3537
Epoch 63/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 714us/step - accuracy: 0.6879 - auc_3: 0.6712 - f1_score: 0.4939 - loss: 217
6.4783 - val_accuracy: 0.8188 - val_auc_3: 0.6435 - val_f1_score: 0.2231 - val_loss: 819.0206
Epoch 64/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 707us/step - accuracy: 0.6877 - auc_3: 0.6731 - f1_score: 0.4939 - loss: 217
1.1484 - val_accuracy: 0.8199 - val_auc_3: 0.6437 - val_f1_score: 0.2231 - val_loss: 817.9370
Epoch 65/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 705us/step - accuracy: 0.6879 - auc_3: 0.6728 - f1_score: 0.4939 - loss: 217
1.6831 - val_accuracy: 0.8199 - val_auc_3: 0.6437 - val_f1_score: 0.2231 - val_loss: 816.2812
Epoch 66/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 789us/step - accuracy: 0.6883 - auc_3: 0.6729 - f1_score: 0.4939 - loss: 216
8.7676 - val_accuracy: 0.8199 - val_auc_3: 0.6437 - val_f1_score: 0.2231 - val_loss: 815.3873
Epoch 67/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 870us/step - accuracy: 0.6878 - auc_3: 0.6727 - f1_score: 0.4939 - loss: 216
8.9343 - val_accuracy: 0.8188 - val_auc_3: 0.6435 - val_f1_score: 0.2231 - val_loss: 813.8256
Epoch 68/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 707us/step - accuracy: 0.6895 - auc_3: 0.6721 - f1_score: 0.4939 - loss: 216
8.2893 - val_accuracy: 0.8188 - val_auc_3: 0.6439 - val_f1_score: 0.2231 - val_loss: 812.8397
Epoch 69/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 746us/step - accuracy: 0.6856 - auc_3: 0.6717 - f1_score: 0.4939 - loss: 216
8.6147 - val_accuracy: 0.8195 - val_auc_3: 0.6440 - val_f1_score: 0.2231 - val_loss: 810.9688
Epoch 70/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 705us/step - accuracy: 0.6884 - auc_3: 0.6730 - f1_score: 0.4939 - loss: 216
3.8818 - val_accuracy: 0.8188 - val_auc_3: 0.6440 - val_f1_score: 0.2231 - val_loss: 809.8965
Epoch 71/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 728us/step - accuracy: 0.6893 - auc_3: 0.6742 - f1_score: 0.4939 - loss: 216
1.5898 - val_accuracy: 0.8184 - val_auc_3: 0.6439 - val_f1_score: 0.2231 - val_loss: 808.8779
Epoch 72/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 700us/step - accuracy: 0.6881 - auc_3: 0.6728 - f1_score: 0.4939 - loss: 216
2.8982 - val_accuracy: 0.8184 - val_auc_3: 0.6442 - val_f1_score: 0.2231 - val_loss: 807.4987
Epoch 73/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 692us/step - accuracy: 0.6881 - auc_3: 0.6739 - f1_score: 0.4939 - loss: 215
9.4360 - val_accuracy: 0.8173 - val_auc_3: 0.6443 - val_f1_score: 0.2231 - val_loss: 806.6118
Epoch 74/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 892us/step - accuracy: 0.6858 - auc_3: 0.6733 - f1_score: 0.4939 - loss: 216
0.1626 - val_accuracy: 0.8180 - val_auc_3: 0.6448 - val_f1_score: 0.2231 - val_loss: 805.1989
Epoch 75/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 779us/step - accuracy: 0.6900 - auc_3: 0.6757 - f1_score: 0.4939 - loss: 215
5.8025 - val_accuracy: 0.8173 - val_auc_3: 0.6443 - val_f1_score: 0.2231 - val_loss: 804.9405
Epoch 76/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 765us/step - accuracy: 0.6907 - auc_3: 0.6755 - f1_score: 0.4939 - loss: 215
3.8794 - val_accuracy: 0.8180 - val_auc_3: 0.6449 - val_f1_score: 0.2231 - val_loss: 802.6245
Epoch 77/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 691us/step - accuracy: 0.6899 - auc_3: 0.6732 - f1_score: 0.4939 - loss: 215
6.6453 - val_accuracy: 0.8184 - val_auc_3: 0.6448 - val_f1_score: 0.2231 - val_loss: 801.8363
Epoch 78/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 704us/step - accuracy: 0.6868 - auc_3: 0.6749 - f1_score: 0.4939 - loss: 215
2.2415 - val_accuracy: 0.8188 - val_auc_3: 0.6449 - val_f1_score: 0.2231 - val_loss: 800.5794
Epoch 79/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 669us/step - accuracy: 0.6878 - auc_3: 0.6745 - f1_score: 0.4939 - loss: 215
1.4912 - val_accuracy: 0.8169 - val_auc_3: 0.6446 - val_f1_score: 0.2231 - val_loss: 800.7315
```

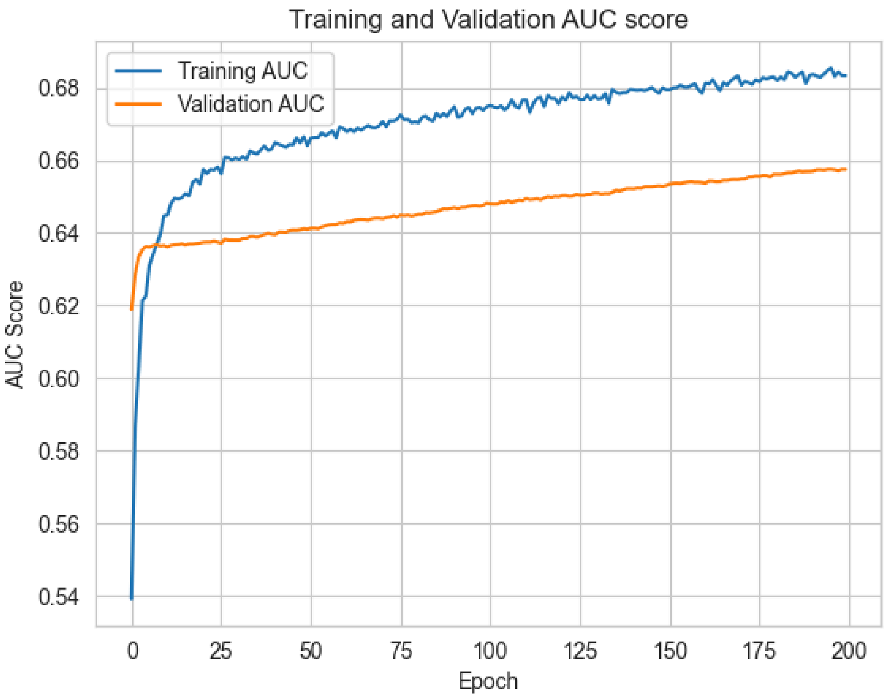
```
Epoch 80/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 690us/step - accuracy: 0.6882 - auc_3: 0.6747 - f1_score: 0.4939 - loss: 215
1.2698 - val_accuracy: 0.8165 - val_auc_3: 0.6449 - val_f1_score: 0.2231 - val_loss: 799.6884
Epoch 81/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 757us/step - accuracy: 0.6889 - auc_3: 0.6738 - f1_score: 0.4939 - loss: 215
1.8792 - val_accuracy: 0.8184 - val_auc_3: 0.6451 - val_f1_score: 0.2231 - val_loss: 798.3287
Epoch 82/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 819us/step - accuracy: 0.6886 - auc_3: 0.6749 - f1_score: 0.4939 - loss: 214
7.1523 - val_accuracy: 0.8180 - val_auc_3: 0.6451 - val_f1_score: 0.2231 - val_loss: 796.8068
Epoch 83/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 698us/step - accuracy: 0.6894 - auc_3: 0.6755 - f1_score: 0.4939 - loss: 214
6.8257 - val_accuracy: 0.8176 - val_auc_3: 0.6456 - val_f1_score: 0.2231 - val_loss: 796.1012
Epoch 84/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 699us/step - accuracy: 0.6902 - auc_3: 0.6744 - f1_score: 0.4939 - loss: 214
7.1133 - val_accuracy: 0.8195 - val_auc_3: 0.6455 - val_f1_score: 0.2231 - val_loss: 794.1171
Epoch 85/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 687us/step - accuracy: 0.6896 - auc_3: 0.6736 - f1_score: 0.4939 - loss: 214
8.3394 - val_accuracy: 0.8176 - val_auc_3: 0.6457 - val_f1_score: 0.2231 - val_loss: 793.9586
Epoch 86/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 694us/step - accuracy: 0.6869 - auc_3: 0.6765 - f1_score: 0.4939 - loss: 214
2.5173 - val_accuracy: 0.8184 - val_auc_3: 0.6459 - val_f1_score: 0.2231 - val_loss: 792.6176
Epoch 87/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 703us/step - accuracy: 0.6918 - auc_3: 0.6742 - f1_score: 0.4939 - loss: 214
4.1904 - val_accuracy: 0.8180 - val_auc_3: 0.6462 - val_f1_score: 0.2231 - val_loss: 791.9048
Epoch 88/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 668us/step - accuracy: 0.6871 - auc_3: 0.6766 - f1_score: 0.4939 - loss: 214
1.8384 - val_accuracy: 0.8191 - val_auc_3: 0.6467 - val_f1_score: 0.2231 - val_loss: 790.2050
Epoch 89/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 740us/step - accuracy: 0.6882 - auc_3: 0.6751 - f1_score: 0.4939 - loss: 214
2.2463 - val_accuracy: 0.8173 - val_auc_3: 0.6466 - val_f1_score: 0.2231 - val_loss: 789.8747
Epoch 90/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 844us/step - accuracy: 0.6883 - auc_3: 0.6761 - f1_score: 0.4939 - loss: 213
9.4565 - val_accuracy: 0.8184 - val_auc_3: 0.6468 - val_f1_score: 0.2231 - val_loss: 788.6678
Epoch 91/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 704us/step - accuracy: 0.6889 - auc_3: 0.6773 - f1_score: 0.4939 - loss: 213
8.2244 - val_accuracy: 0.8176 - val_auc_3: 0.6470 - val_f1_score: 0.2231 - val_loss: 787.7002
Epoch 92/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 697us/step - accuracy: 0.6875 - auc_3: 0.6744 - f1_score: 0.4939 - loss: 214
0.9678 - val_accuracy: 0.8180 - val_auc_3: 0.6467 - val_f1_score: 0.2231 - val_loss: 786.7294
Epoch 93/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 694us/step - accuracy: 0.6902 - auc_3: 0.6754 - f1_score: 0.4939 - loss: 213
6.2581 - val_accuracy: 0.8180 - val_auc_3: 0.6470 - val_f1_score: 0.2231 - val_loss: 786.0438
Epoch 94/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 660us/step - accuracy: 0.6893 - auc_3: 0.6777 - f1_score: 0.4939 - loss: 213
3.3257 - val_accuracy: 0.8176 - val_auc_3: 0.6470 - val_f1_score: 0.2231 - val_loss: 785.7266
Epoch 95/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 685us/step - accuracy: 0.6905 - auc_3: 0.6777 - f1_score: 0.4939 - loss: 213
1.4763 - val_accuracy: 0.8191 - val_auc_3: 0.6473 - val_f1_score: 0.2231 - val_loss: 783.3875
Epoch 96/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 661us/step - accuracy: 0.6877 - auc_3: 0.6765 - f1_score: 0.4939 - loss: 213
4.4099 - val_accuracy: 0.8173 - val_auc_3: 0.6475 - val_f1_score: 0.2231 - val_loss: 783.5164
Epoch 97/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 811us/step - accuracy: 0.6885 - auc_3: 0.6764 - f1_score: 0.4939 - loss: 213
4.8218 - val_accuracy: 0.8184 - val_auc_3: 0.6475 - val_f1_score: 0.2231 - val_loss: 782.1556
Epoch 98/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 776us/step - accuracy: 0.6899 - auc_3: 0.6795 - f1_score: 0.4939 - loss: 212
7.9729 - val_accuracy: 0.8180 - val_auc_3: 0.6474 - val_f1_score: 0.2231 - val_loss: 781.0989
Epoch 99/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 691us/step - accuracy: 0.6906 - auc_3: 0.6768 - f1_score: 0.4939 - loss: 212
9.0554 - val_accuracy: 0.8180 - val_auc_3: 0.6476 - val_f1_score: 0.2231 - val_loss: 780.4418
Epoch 100/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 667us/step - accuracy: 0.6914 - auc_3: 0.6789 - f1_score: 0.4939 - loss: 212
6.3318 - val_accuracy: 0.8180 - val_auc_3: 0.6482 - val_f1_score: 0.2231 - val_loss: 779.7672
Epoch 101/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 678us/step - accuracy: 0.6888 - auc_3: 0.6796 - f1_score: 0.4939 - loss: 212
6.1538 - val_accuracy: 0.8176 - val_auc_3: 0.6480 - val_f1_score: 0.2231 - val_loss: 778.7658
Epoch 102/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 679us/step - accuracy: 0.6900 - auc_3: 0.6775 - f1_score: 0.4939 - loss: 212
7.9062 - val_accuracy: 0.8176 - val_auc_3: 0.6480 - val_f1_score: 0.2231 - val_loss: 779.1167
Epoch 103/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 689us/step - accuracy: 0.6895 - auc_3: 0.6770 - f1_score: 0.4939 - loss: 212
6.9111 - val_accuracy: 0.8188 - val_auc_3: 0.6480 - val_f1_score: 0.2231 - val_loss: 777.7007
Epoch 104/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 729us/step - accuracy: 0.6909 - auc_3: 0.6780 - f1_score: 0.4939 - loss: 212
5.0618 - val_accuracy: 0.8176 - val_auc_3: 0.6485 - val_f1_score: 0.2231 - val_loss: 776.8036
Epoch 105/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 817us/step - accuracy: 0.6897 - auc_3: 0.6769 - f1_score: 0.4939 - loss: 212
6.0693 - val_accuracy: 0.8176 - val_auc_3: 0.6484 - val_f1_score: 0.2231 - val_loss: 775.7505
Epoch 106/200
```

```
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 686us/step - accuracy: 0.6877 - auc_3: 0.6768 - f1_score: 0.4939 - loss: 2125.5273 - val_accuracy: 0.8176 - val_auc_3: 0.6488 - val_f1_score: 0.2231 - val_loss: 774.4695
Epoch 107/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 670us/step - accuracy: 0.6908 - auc_3: 0.6795 - f1_score: 0.4939 - loss: 2120.2368 - val_accuracy: 0.8169 - val_auc_3: 0.6484 - val_f1_score: 0.2231 - val_loss: 774.0714
Epoch 108/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 726us/step - accuracy: 0.6904 - auc_3: 0.6795 - f1_score: 0.4939 - loss: 2119.4558 - val_accuracy: 0.8165 - val_auc_3: 0.6488 - val_f1_score: 0.2231 - val_loss: 774.7234
Epoch 109/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 660us/step - accuracy: 0.6902 - auc_3: 0.6783 - f1_score: 0.4939 - loss: 2121.4978 - val_accuracy: 0.8173 - val_auc_3: 0.6490 - val_f1_score: 0.2231 - val_loss: 773.2806
Epoch 110/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 698us/step - accuracy: 0.6899 - auc_3: 0.6791 - f1_score: 0.4939 - loss: 2120.2607 - val_accuracy: 0.8173 - val_auc_3: 0.6489 - val_f1_score: 0.2231 - val_loss: 772.2445
Epoch 111/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 729us/step - accuracy: 0.6904 - auc_3: 0.6800 - f1_score: 0.4939 - loss: 2115.9368 - val_accuracy: 0.8173 - val_auc_3: 0.6495 - val_f1_score: 0.2231 - val_loss: 770.7770
Epoch 112/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 806us/step - accuracy: 0.6880 - auc_3: 0.6771 - f1_score: 0.4939 - loss: 2119.9812 - val_accuracy: 0.8173 - val_auc_3: 0.6491 - val_f1_score: 0.2231 - val_loss: 771.9280
Epoch 113/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 781us/step - accuracy: 0.6908 - auc_3: 0.6797 - f1_score: 0.4939 - loss: 2115.7400 - val_accuracy: 0.8180 - val_auc_3: 0.6492 - val_f1_score: 0.2231 - val_loss: 770.7756
Epoch 114/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 696us/step - accuracy: 0.6923 - auc_3: 0.6807 - f1_score: 0.4939 - loss: 2112.9194 - val_accuracy: 0.8180 - val_auc_3: 0.6494 - val_f1_score: 0.2231 - val_loss: 769.0343
Epoch 115/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 708us/step - accuracy: 0.6929 - auc_3: 0.6815 - f1_score: 0.4939 - loss: 2111.7788 - val_accuracy: 0.8169 - val_auc_3: 0.6491 - val_f1_score: 0.2231 - val_loss: 769.1921
Epoch 116/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 725us/step - accuracy: 0.6911 - auc_3: 0.6781 - f1_score: 0.4939 - loss: 2115.7727 - val_accuracy: 0.8165 - val_auc_3: 0.6496 - val_f1_score: 0.2231 - val_loss: 768.2681
Epoch 117/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 723us/step - accuracy: 0.6917 - auc_3: 0.6816 - f1_score: 0.4939 - loss: 2109.3398 - val_accuracy: 0.8165 - val_auc_3: 0.6500 - val_f1_score: 0.2231 - val_loss: 767.2527
Epoch 118/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 887us/step - accuracy: 0.6900 - auc_3: 0.6818 - f1_score: 0.4939 - loss: 2109.2964 - val_accuracy: 0.8169 - val_auc_3: 0.6498 - val_f1_score: 0.2231 - val_loss: 767.0977
Epoch 119/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 776us/step - accuracy: 0.6896 - auc_3: 0.6813 - f1_score: 0.4939 - loss: 2109.3130 - val_accuracy: 0.8169 - val_auc_3: 0.6501 - val_f1_score: 0.2231 - val_loss: 766.4718
Epoch 120/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 862us/step - accuracy: 0.6898 - auc_3: 0.6801 - f1_score: 0.4939 - loss: 2110.2271 - val_accuracy: 0.8173 - val_auc_3: 0.6500 - val_f1_score: 0.2231 - val_loss: 764.8976
Epoch 121/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 745us/step - accuracy: 0.6912 - auc_3: 0.6805 - f1_score: 0.4939 - loss: 2108.9014 - val_accuracy: 0.8161 - val_auc_3: 0.6503 - val_f1_score: 0.2231 - val_loss: 764.5753
Epoch 122/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 747us/step - accuracy: 0.6927 - auc_3: 0.6804 - f1_score: 0.4939 - loss: 2109.4609 - val_accuracy: 0.8165 - val_auc_3: 0.6503 - val_f1_score: 0.2231 - val_loss: 763.7055
Epoch 123/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 771us/step - accuracy: 0.6935 - auc_3: 0.6838 - f1_score: 0.4939 - loss: 2103.4893 - val_accuracy: 0.8154 - val_auc_3: 0.6501 - val_f1_score: 0.2231 - val_loss: 763.7889
Epoch 124/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 856us/step - accuracy: 0.6898 - auc_3: 0.6803 - f1_score: 0.4939 - loss: 2107.7117 - val_accuracy: 0.8158 - val_auc_3: 0.6502 - val_f1_score: 0.2231 - val_loss: 762.4655
Epoch 125/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 740us/step - accuracy: 0.6917 - auc_3: 0.6829 - f1_score: 0.4939 - loss: 2103.0767 - val_accuracy: 0.8173 - val_auc_3: 0.6506 - val_f1_score: 0.2231 - val_loss: 761.8130
Epoch 126/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 707us/step - accuracy: 0.6906 - auc_3: 0.6800 - f1_score: 0.4939 - loss: 2106.2437 - val_accuracy: 0.8146 - val_auc_3: 0.6504 - val_f1_score: 0.2231 - val_loss: 762.3754
Epoch 127/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 703us/step - accuracy: 0.6900 - auc_3: 0.6799 - f1_score: 0.4939 - loss: 2106.6228 - val_accuracy: 0.8154 - val_auc_3: 0.6504 - val_f1_score: 0.2231 - val_loss: 760.6650
Epoch 128/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 739us/step - accuracy: 0.6912 - auc_3: 0.6811 - f1_score: 0.4939 - loss: 2103.7290 - val_accuracy: 0.8165 - val_auc_3: 0.6507 - val_f1_score: 0.2231 - val_loss: 759.7440
Epoch 129/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 697us/step - accuracy: 0.6923 - auc_3: 0.6815 - f1_score: 0.4939 - loss: 2102.1819 - val_accuracy: 0.8150 - val_auc_3: 0.6506 - val_f1_score: 0.2231 - val_loss: 760.3443
Epoch 130/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 787us/step - accuracy: 0.6888 - auc_3: 0.6794 - f1_score: 0.4939 - loss: 2103.1589 - val_accuracy: 0.8161 - val_auc_3: 0.6510 - val_f1_score: 0.2231 - val_loss: 758.7997
Epoch 131/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 901us/step - accuracy: 0.6902 - auc_3: 0.6807 - f1_score: 0.4939 - loss: 2103.2627 - val_accuracy: 0.8150 - val_auc_3: 0.6510 - val_f1_score: 0.2231 - val_loss: 759.2770
Epoch 132/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 713us/step - accuracy: 0.6928 - auc_3: 0.6810 - f1_score: 0.4939 - loss: 210
```

```
0.8376 - val_accuracy: 0.8158 - val_auc_3: 0.6507 - val_f1_score: 0.2231 - val_loss: 758.3180
Epoch 133/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 679us/step - accuracy: 0.6900 - auc_3: 0.6817 - f1_score: 0.4939 - loss: 210
1.0461 - val_accuracy: 0.8161 - val_auc_3: 0.6509 - val_f1_score: 0.2231 - val_loss: 757.5227
Epoch 134/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 684us/step - accuracy: 0.6913 - auc_3: 0.6789 - f1_score: 0.4939 - loss: 210
4.2805 - val_accuracy: 0.8161 - val_auc_3: 0.6508 - val_f1_score: 0.2231 - val_loss: 757.1846
Epoch 135/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 687us/step - accuracy: 0.6907 - auc_3: 0.6836 - f1_score: 0.4939 - loss: 209
6.0752 - val_accuracy: 0.8150 - val_auc_3: 0.6511 - val_f1_score: 0.2231 - val_loss: 756.3127
Epoch 136/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 653us/step - accuracy: 0.6918 - auc_3: 0.6817 - f1_score: 0.4939 - loss: 209
8.1990 - val_accuracy: 0.8169 - val_auc_3: 0.6518 - val_f1_score: 0.2231 - val_loss: 755.5259
Epoch 137/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 736us/step - accuracy: 0.6924 - auc_3: 0.6827 - f1_score: 0.4939 - loss: 209
6.3582 - val_accuracy: 0.8161 - val_auc_3: 0.6515 - val_f1_score: 0.2231 - val_loss: 755.5557
Epoch 138/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 739us/step - accuracy: 0.6904 - auc_3: 0.6819 - f1_score: 0.4939 - loss: 209
6.7625 - val_accuracy: 0.8165 - val_auc_3: 0.6519 - val_f1_score: 0.2231 - val_loss: 754.2095
Epoch 139/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 755us/step - accuracy: 0.6914 - auc_3: 0.6809 - f1_score: 0.4939 - loss: 209
9.5054 - val_accuracy: 0.8158 - val_auc_3: 0.6522 - val_f1_score: 0.2231 - val_loss: 754.0060
Epoch 140/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 769us/step - accuracy: 0.6919 - auc_3: 0.6824 - f1_score: 0.4939 - loss: 209
4.8264 - val_accuracy: 0.8173 - val_auc_3: 0.6522 - val_f1_score: 0.2231 - val_loss: 752.6600
Epoch 141/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 750us/step - accuracy: 0.6905 - auc_3: 0.6812 - f1_score: 0.4939 - loss: 209
5.7515 - val_accuracy: 0.8165 - val_auc_3: 0.6521 - val_f1_score: 0.2231 - val_loss: 752.3420
Epoch 142/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 757us/step - accuracy: 0.6902 - auc_3: 0.6827 - f1_score: 0.4939 - loss: 209
3.5623 - val_accuracy: 0.8165 - val_auc_3: 0.6524 - val_f1_score: 0.2231 - val_loss: 751.6852
Epoch 143/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 714us/step - accuracy: 0.6913 - auc_3: 0.6812 - f1_score: 0.4939 - loss: 209
5.9246 - val_accuracy: 0.8165 - val_auc_3: 0.6523 - val_f1_score: 0.2231 - val_loss: 751.5945
Epoch 144/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 712us/step - accuracy: 0.6910 - auc_3: 0.6828 - f1_score: 0.4939 - loss: 209
1.8796 - val_accuracy: 0.8158 - val_auc_3: 0.6527 - val_f1_score: 0.2231 - val_loss: 751.6364
Epoch 145/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 949us/step - accuracy: 0.6908 - auc_3: 0.6813 - f1_score: 0.4939 - loss: 209
3.7834 - val_accuracy: 0.8161 - val_auc_3: 0.6527 - val_f1_score: 0.2231 - val_loss: 750.8409
Epoch 146/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 728us/step - accuracy: 0.6899 - auc_3: 0.6829 - f1_score: 0.4939 - loss: 209
2.4287 - val_accuracy: 0.8161 - val_auc_3: 0.6527 - val_f1_score: 0.2231 - val_loss: 750.2346
Epoch 147/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 758us/step - accuracy: 0.6916 - auc_3: 0.6835 - f1_score: 0.4939 - loss: 208
9.6750 - val_accuracy: 0.8154 - val_auc_3: 0.6529 - val_f1_score: 0.2231 - val_loss: 749.5161
Epoch 148/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 740us/step - accuracy: 0.6923 - auc_3: 0.6816 - f1_score: 0.4939 - loss: 209
2.3074 - val_accuracy: 0.8146 - val_auc_3: 0.6530 - val_f1_score: 0.2231 - val_loss: 749.5339
Epoch 149/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 743us/step - accuracy: 0.6909 - auc_3: 0.6825 - f1_score: 0.4939 - loss: 209
1.2629 - val_accuracy: 0.8150 - val_auc_3: 0.6528 - val_f1_score: 0.2231 - val_loss: 748.7080
Epoch 150/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 682us/step - accuracy: 0.6930 - auc_3: 0.6834 - f1_score: 0.4939 - loss: 208
9.3379 - val_accuracy: 0.8150 - val_auc_3: 0.6531 - val_f1_score: 0.2231 - val_loss: 748.4804
Epoch 151/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 715us/step - accuracy: 0.6931 - auc_3: 0.6836 - f1_score: 0.4939 - loss: 208
8.9373 - val_accuracy: 0.8143 - val_auc_3: 0.6534 - val_f1_score: 0.2231 - val_loss: 747.5773
Epoch 152/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 808us/step - accuracy: 0.6941 - auc_3: 0.6839 - f1_score: 0.4939 - loss: 208
7.7092 - val_accuracy: 0.8143 - val_auc_3: 0.6536 - val_f1_score: 0.2231 - val_loss: 747.0736
Epoch 153/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 689us/step - accuracy: 0.6928 - auc_3: 0.6827 - f1_score: 0.4939 - loss: 208
8.2463 - val_accuracy: 0.8143 - val_auc_3: 0.6537 - val_f1_score: 0.2231 - val_loss: 746.9031
Epoch 154/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 730us/step - accuracy: 0.6933 - auc_3: 0.6832 - f1_score: 0.4939 - loss: 208
6.9546 - val_accuracy: 0.8154 - val_auc_3: 0.6536 - val_f1_score: 0.2231 - val_loss: 746.0906
Epoch 155/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 694us/step - accuracy: 0.6924 - auc_3: 0.6837 - f1_score: 0.4939 - loss: 208
5.5266 - val_accuracy: 0.8146 - val_auc_3: 0.6538 - val_f1_score: 0.2231 - val_loss: 745.9182
Epoch 156/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 715us/step - accuracy: 0.6921 - auc_3: 0.6841 - f1_score: 0.4939 - loss: 208
4.4963 - val_accuracy: 0.8158 - val_auc_3: 0.6540 - val_f1_score: 0.2231 - val_loss: 745.1731
Epoch 157/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 704us/step - accuracy: 0.6922 - auc_3: 0.6825 - f1_score: 0.4939 - loss: 208
5.8650 - val_accuracy: 0.8150 - val_auc_3: 0.6541 - val_f1_score: 0.2231 - val_loss: 744.6752
Epoch 158/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 736us/step - accuracy: 0.6937 - auc_3: 0.6845 - f1_score: 0.4939 - loss: 208
4.3140 - val_accuracy: 0.8150 - val_auc_3: 0.6540 - val_f1_score: 0.2231 - val_loss: 744.2276
```

```
Epoch 159/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 746us/step - accuracy: 0.6929 - auc_3: 0.6830 - f1_score: 0.4939 - loss: 208
2.6958 - val_accuracy: 0.8150 - val_auc_3: 0.6540 - val_f1_score: 0.2231 - val_loss: 744.0947
Epoch 160/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 688us/step - accuracy: 0.6911 - auc_3: 0.6807 - f1_score: 0.4939 - loss: 208
7.2473 - val_accuracy: 0.8150 - val_auc_3: 0.6539 - val_f1_score: 0.2231 - val_loss: 743.6872
Epoch 161/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 674us/step - accuracy: 0.6915 - auc_3: 0.6841 - f1_score: 0.4939 - loss: 208
3.7012 - val_accuracy: 0.8143 - val_auc_3: 0.6537 - val_f1_score: 0.2231 - val_loss: 743.4077
Epoch 162/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 641us/step - accuracy: 0.6929 - auc_3: 0.6837 - f1_score: 0.4939 - loss: 208
2.1597 - val_accuracy: 0.8158 - val_auc_3: 0.6543 - val_f1_score: 0.2231 - val_loss: 741.9312
Epoch 163/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 684us/step - accuracy: 0.6921 - auc_3: 0.6855 - f1_score: 0.4939 - loss: 208
0.1670 - val_accuracy: 0.8143 - val_auc_3: 0.6543 - val_f1_score: 0.2231 - val_loss: 742.2313
Epoch 164/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 681us/step - accuracy: 0.6945 - auc_3: 0.6844 - f1_score: 0.4939 - loss: 207
8.8362 - val_accuracy: 0.8154 - val_auc_3: 0.6541 - val_f1_score: 0.2231 - val_loss: 741.5776
Epoch 165/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 689us/step - accuracy: 0.6934 - auc_3: 0.6824 - f1_score: 0.4939 - loss: 208
3.4480 - val_accuracy: 0.8143 - val_auc_3: 0.6541 - val_f1_score: 0.2231 - val_loss: 742.0972
Epoch 166/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 815us/step - accuracy: 0.6918 - auc_3: 0.6847 - f1_score: 0.4939 - loss: 207
9.7808 - val_accuracy: 0.8154 - val_auc_3: 0.6545 - val_f1_score: 0.2231 - val_loss: 740.8836
Epoch 167/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 736us/step - accuracy: 0.6945 - auc_3: 0.6831 - f1_score: 0.4939 - loss: 208
0.3994 - val_accuracy: 0.8135 - val_auc_3: 0.6544 - val_f1_score: 0.2231 - val_loss: 740.7405
Epoch 168/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 699us/step - accuracy: 0.6957 - auc_3: 0.6862 - f1_score: 0.4939 - loss: 207
6.3818 - val_accuracy: 0.8143 - val_auc_3: 0.6546 - val_f1_score: 0.2231 - val_loss: 740.4163
Epoch 169/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 675us/step - accuracy: 0.6915 - auc_3: 0.6850 - f1_score: 0.4939 - loss: 207
7.9370 - val_accuracy: 0.8154 - val_auc_3: 0.6547 - val_f1_score: 0.2231 - val_loss: 739.2770
Epoch 170/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 701us/step - accuracy: 0.6951 - auc_3: 0.6884 - f1_score: 0.4939 - loss: 207
3.0508 - val_accuracy: 0.8161 - val_auc_3: 0.6547 - val_f1_score: 0.2231 - val_loss: 739.1093
Epoch 171/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 716us/step - accuracy: 0.6920 - auc_3: 0.6827 - f1_score: 0.4939 - loss: 208
1.9160 - val_accuracy: 0.8161 - val_auc_3: 0.6549 - val_f1_score: 0.2231 - val_loss: 738.9121
Epoch 172/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 722us/step - accuracy: 0.6924 - auc_3: 0.6833 - f1_score: 0.4939 - loss: 207
8.2607 - val_accuracy: 0.8146 - val_auc_3: 0.6549 - val_f1_score: 0.2231 - val_loss: 738.4277
Epoch 173/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 939us/step - accuracy: 0.6947 - auc_3: 0.6867 - f1_score: 0.4939 - loss: 207
2.7400 - val_accuracy: 0.8143 - val_auc_3: 0.6554 - val_f1_score: 0.2231 - val_loss: 738.3849
Epoch 174/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 702us/step - accuracy: 0.6903 - auc_3: 0.6846 - f1_score: 0.4939 - loss: 207
6.5325 - val_accuracy: 0.8143 - val_auc_3: 0.6554 - val_f1_score: 0.2231 - val_loss: 737.6655
Epoch 175/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 718us/step - accuracy: 0.6944 - auc_3: 0.6867 - f1_score: 0.4939 - loss: 207
3.6716 - val_accuracy: 0.8154 - val_auc_3: 0.6556 - val_f1_score: 0.2231 - val_loss: 737.3701
Epoch 176/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 704us/step - accuracy: 0.6936 - auc_3: 0.6853 - f1_score: 0.4939 - loss: 207
4.3950 - val_accuracy: 0.8143 - val_auc_3: 0.6556 - val_f1_score: 0.2231 - val_loss: 737.6127
Epoch 177/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 696us/step - accuracy: 0.6952 - auc_3: 0.6882 - f1_score: 0.4939 - loss: 207
0.4893 - val_accuracy: 0.8150 - val_auc_3: 0.6558 - val_f1_score: 0.2231 - val_loss: 736.3100
Epoch 178/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 681us/step - accuracy: 0.6945 - auc_3: 0.6867 - f1_score: 0.4939 - loss: 207
2.1895 - val_accuracy: 0.8146 - val_auc_3: 0.6558 - val_f1_score: 0.2231 - val_loss: 736.9030
Epoch 179/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 773us/step - accuracy: 0.6918 - auc_3: 0.6851 - f1_score: 0.4939 - loss: 207
3.8020 - val_accuracy: 0.8143 - val_auc_3: 0.6554 - val_f1_score: 0.2231 - val_loss: 736.2632
Epoch 180/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 796us/step - accuracy: 0.6955 - auc_3: 0.6867 - f1_score: 0.4939 - loss: 207
1.1414 - val_accuracy: 0.8146 - val_auc_3: 0.6562 - val_f1_score: 0.2231 - val_loss: 735.0018
Epoch 181/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 710us/step - accuracy: 0.6940 - auc_3: 0.6844 - f1_score: 0.4939 - loss: 207
3.4524 - val_accuracy: 0.8135 - val_auc_3: 0.6561 - val_f1_score: 0.2231 - val_loss: 735.6100
Epoch 182/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 716us/step - accuracy: 0.6941 - auc_3: 0.6875 - f1_score: 0.4939 - loss: 207
0.5229 - val_accuracy: 0.8135 - val_auc_3: 0.6561 - val_f1_score: 0.2231 - val_loss: 735.4761
Epoch 183/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 670us/step - accuracy: 0.6923 - auc_3: 0.6849 - f1_score: 0.4939 - loss: 207
1.7368 - val_accuracy: 0.8132 - val_auc_3: 0.6563 - val_f1_score: 0.2231 - val_loss: 735.3262
Epoch 184/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 684us/step - accuracy: 0.6939 - auc_3: 0.6869 - f1_score: 0.4939 - loss: 206
9.9504 - val_accuracy: 0.8146 - val_auc_3: 0.6566 - val_f1_score: 0.2231 - val_loss: 733.9603
Epoch 185/200
```

```
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 751us/step - accuracy: 0.6937 - auc_3: 0.6859 - f1_score: 0.4939 - loss: 206
9.3342 - val_accuracy: 0.8146 - val_auc_3: 0.6566 - val_f1_score: 0.2231 - val_loss: 733.6747
Epoch 186/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 766us/step - accuracy: 0.6944 - auc_3: 0.6857 - f1_score: 0.4939 - loss: 206
9.3828 - val_accuracy: 0.8135 - val_auc_3: 0.6568 - val_f1_score: 0.2231 - val_loss: 734.0336
Epoch 187/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 672us/step - accuracy: 0.6920 - auc_3: 0.6878 - f1_score: 0.4939 - loss: 206
6.5139 - val_accuracy: 0.8143 - val_auc_3: 0.6570 - val_f1_score: 0.2231 - val_loss: 732.8321
Epoch 188/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 693us/step - accuracy: 0.6941 - auc_3: 0.6885 - f1_score: 0.4939 - loss: 206
5.1406 - val_accuracy: 0.8143 - val_auc_3: 0.6568 - val_f1_score: 0.2231 - val_loss: 732.7299
Epoch 189/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 700us/step - accuracy: 0.6930 - auc_3: 0.6845 - f1_score: 0.4939 - loss: 207
1.5432 - val_accuracy: 0.8143 - val_auc_3: 0.6569 - val_f1_score: 0.2231 - val_loss: 732.3109
Epoch 190/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 666us/step - accuracy: 0.6946 - auc_3: 0.6861 - f1_score: 0.4939 - loss: 206
7.6934 - val_accuracy: 0.8143 - val_auc_3: 0.6569 - val_f1_score: 0.2231 - val_loss: 732.5128
Epoch 191/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 676us/step - accuracy: 0.6946 - auc_3: 0.6885 - f1_score: 0.4939 - loss: 206
3.1323 - val_accuracy: 0.8146 - val_auc_3: 0.6570 - val_f1_score: 0.2231 - val_loss: 731.8049
Epoch 192/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 759us/step - accuracy: 0.6921 - auc_3: 0.6861 - f1_score: 0.4939 - loss: 206
8.4553 - val_accuracy: 0.8146 - val_auc_3: 0.6574 - val_f1_score: 0.2231 - val_loss: 730.8094
Epoch 193/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 780us/step - accuracy: 0.6940 - auc_3: 0.6858 - f1_score: 0.4939 - loss: 206
6.6428 - val_accuracy: 0.8150 - val_auc_3: 0.6574 - val_f1_score: 0.2231 - val_loss: 730.7613
Epoch 194/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 686us/step - accuracy: 0.6937 - auc_3: 0.6872 - f1_score: 0.4939 - loss: 206
6.1494 - val_accuracy: 0.8146 - val_auc_3: 0.6573 - val_f1_score: 0.2231 - val_loss: 730.8159
Epoch 195/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 731us/step - accuracy: 0.6934 - auc_3: 0.6868 - f1_score: 0.4939 - loss: 206
6.3591 - val_accuracy: 0.8150 - val_auc_3: 0.6575 - val_f1_score: 0.2231 - val_loss: 729.9662
Epoch 196/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 696us/step - accuracy: 0.6949 - auc_3: 0.6877 - f1_score: 0.4939 - loss: 206
4.1123 - val_accuracy: 0.8150 - val_auc_3: 0.6575 - val_f1_score: 0.2231 - val_loss: 729.7014
Epoch 197/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 756us/step - accuracy: 0.6922 - auc_3: 0.6863 - f1_score: 0.4939 - loss: 206
6.5059 - val_accuracy: 0.8158 - val_auc_3: 0.6574 - val_f1_score: 0.2231 - val_loss: 729.1404
Epoch 198/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 870us/step - accuracy: 0.6941 - auc_3: 0.6884 - f1_score: 0.4939 - loss: 206
2.9297 - val_accuracy: 0.8139 - val_auc_3: 0.6571 - val_f1_score: 0.2231 - val_loss: 729.5283
Epoch 199/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 742us/step - accuracy: 0.6936 - auc_3: 0.6851 - f1_score: 0.4939 - loss: 206
6.9243 - val_accuracy: 0.8143 - val_auc_3: 0.6575 - val_f1_score: 0.2231 - val_loss: 728.7767
Epoch 200/200
991/991 ━━━━━━━━━━━━━━━━━━━━ 1s 713us/step - accuracy: 0.6945 - auc_3: 0.6875 - f1_score: 0.4939 - loss: 206
3.8174 - val_accuracy: 0.8146 - val_auc_3: 0.6575 - val_f1_score: 0.2231 - val_loss: 728.6266
```



In [41]:
```
update_summary(summary_df,
               'SMOTE NN',
               y_train_sigmoid,
               smote_model.predict(X_train),
```

```
                y_val_sigmoid,
                smote_model.predict(X_val),
                y_test_sigmoid,
                smote_model.predict(X_test),
                class1_only=True)
summary_df
```

```
753/753 ━━━━━━━━━━━━━━━━━━━━ 0s 329us/step
84/84 ━━━━━━━━━━━━━━━━━━━━ 0s 301us/step
93/93 ━━━━━━━━━━━━━━━━━━━━ 0s 391us/step
```

Out[41]:

	Model	Train AUC	Val AUC	Test AUC
0	Logit as Benchmark	0.6707	0.6637	0.6637
1	LASSO Logit	0.6835	0.6791	0.6791
2	Random Forest CV	0.9686	0.7086	0.7086
3	GBM CV	0.8193	0.7133	0.7133
4	Sigmoid NN	0.6578	0.6321	0.6611
5	Softmax NN	0.6555	0.6314	0.6611
6	SMOTE NN	0.6879	0.6579	0.6740

Surprisingly, with oversampling, the neural network performs much better. This result gives confidence that the model capture the patterns of popular articles better via oversampling.

Stacked Model

This model is a neural network similar to the sigmoid neural network, with a twist that it includes the predictions of the gradient boosting model as an input. The idea is to take advantages of the gradient boosting great performance to improve the neural network performance.

In [42]:

```
# split train, val, test again with engineered features
outcome = news_df["is_popular"]
# features = news_df[high_performance_predictors]
features = news_df.drop(columns=exclude_cols)
# features = news_df.drop(columns=['timedelta', 'is_popular', 'article_id'])
prng = np.random.RandomState(42)
X_train, X_test, y_train, y_test = train_test_split(features, outcome, test_size=0.1, random_state=prng)
X_val, y_val = X_test, y_test
# X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.1, random_state=prng)
X_train_no_scale, X_val_no_scale, X_test_no_scale = X_train.copy()[gbm_high_perm], X_val.copy()[gbm_high_pe

# normalize data
scaler = MinMaxScaler(feature_range=(-1, 1))
# scaler = StandardScaler()
# scaler.fit(features)
columns_not_to_scale = [col for col in X_train.columns if col not in binary_cols]
scaler.fit(X_train[columns_not_to_scale])

X_train[columns_not_to_scale] = scaler.transform(X_train[columns_not_to_scale])
X_val[columns_not_to_scale] = scaler.transform(X_val[columns_not_to_scale])
X_test[columns_not_to_scale] = scaler.transform(X_test[columns_not_to_scale])

gbm_pred_train, gbm_pred_val, gbm_pred_test = gbm_model.predict_proba(X_train_no_scale), gbm_model.predict_

X_train = np.hstack((X_train, gbm_pred_train))
X_val = np.hstack((X_val, gbm_pred_val))
X_test = np.hstack((X_test, gbm_pred_test))
# X_train = np.hstack((X_train, gbm_pred_train[:,1].reshape(-1, 1)))
# X_val = np.hstack((X_val, gbm_pred_val[:,1].reshape(-1, 1)))
# X_test = np.hstack((X_test, gbm_pred_test[:,1].reshape(-1, 1)))
```

In [43]:

```
from keras.metrics import AUC, F1Score
from keras.models import Sequential
from keras.layers import Input, Dense, Normalization, Dropout, BatchNormalization
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
from keras.regularizers import l1
import keras

l1_reg = 0.5
```

```
# Build the simple fully connected single hidden layer network model
stacked_model = Sequential([
    Input(shape=X_train.shape[1:]),
    # Normalization(axis=-1),
    # Dense(256, activation='relu', kernel_regularizer=l1(0.5)),
    Dense(256, activation='relu', kernel_regularizer=l1(l1_reg)),
    # Dropout(0.4),
    Dense(1, activation='sigmoid', kernel_regularizer=l1(l1_reg))
])

# Compile the model
opt = Adam(learning_rate=0.00001)
stacked_model.compile(loss=custom_loss, optimizer=opt, metrics=[AUC(), 'accuracy', F1Score()])

# Fit the model
keras.utils.set_random_seed(42)  # for reproducibility
stacked_history = stacked_model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=200, batch_siz

plot_history(stacked_history.history)
```

```
Epoch 1/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.7849 - auc_4: 0.5201 - f1_score: 0.2223 - loss: 1157.
2595 - val_accuracy: 0.8911 - val_auc_4: 0.5272 - val_f1_score: 0.1965 - val_loss: 856.8764
Epoch 2/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 746us/step - accuracy: 0.8750 - auc_4: 0.5539 - f1_score: 0.2223 - loss: 95
4.2651 - val_accuracy: 0.8911 - val_auc_4: 0.5547 - val_f1_score: 0.1965 - val_loss: 942.5197
Epoch 3/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 729us/step - accuracy: 0.8750 - auc_4: 0.5749 - f1_score: 0.2223 - loss: 91
1.5139 - val_accuracy: 0.8911 - val_auc_4: 0.5690 - val_f1_score: 0.1965 - val_loss: 977.7761
Epoch 4/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 750us/step - accuracy: 0.8750 - auc_4: 0.5908 - f1_score: 0.2223 - loss: 89
6.9622 - val_accuracy: 0.8911 - val_auc_4: 0.5842 - val_f1_score: 0.1965 - val_loss: 980.3577
Epoch 5/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 720us/step - accuracy: 0.8750 - auc_4: 0.6038 - f1_score: 0.2223 - loss: 88
5.4709 - val_accuracy: 0.8911 - val_auc_4: 0.6047 - val_f1_score: 0.1965 - val_loss: 971.7957
Epoch 6/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 739us/step - accuracy: 0.8750 - auc_4: 0.6159 - f1_score: 0.2223 - loss: 87
4.2792 - val_accuracy: 0.8911 - val_auc_4: 0.6118 - val_f1_score: 0.1965 - val_loss: 960.7734
Epoch 7/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 698us/step - accuracy: 0.8750 - auc_4: 0.6270 - f1_score: 0.2223 - loss: 86
3.2770 - val_accuracy: 0.8911 - val_auc_4: 0.6203 - val_f1_score: 0.1965 - val_loss: 949.4267
Epoch 8/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 720us/step - accuracy: 0.8750 - auc_4: 0.6361 - f1_score: 0.2223 - loss: 85
2.4893 - val_accuracy: 0.8911 - val_auc_4: 0.6310 - val_f1_score: 0.1965 - val_loss: 938.3997
Epoch 9/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 914us/step - accuracy: 0.8750 - auc_4: 0.6439 - f1_score: 0.2223 - loss: 84
1.9200 - val_accuracy: 0.8911 - val_auc_4: 0.6397 - val_f1_score: 0.1965 - val_loss: 927.5270
Epoch 10/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 990us/step - accuracy: 0.8750 - auc_4: 0.6510 - f1_score: 0.2223 - loss: 83
1.5875 - val_accuracy: 0.8911 - val_auc_4: 0.6453 - val_f1_score: 0.1965 - val_loss: 917.1147
Epoch 11/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 931us/step - accuracy: 0.8750 - auc_4: 0.6569 - f1_score: 0.2223 - loss: 82
1.5290 - val_accuracy: 0.8911 - val_auc_4: 0.6441 - val_f1_score: 0.1965 - val_loss: 907.0573
Epoch 12/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 862us/step - accuracy: 0.8750 - auc_4: 0.6612 - f1_score: 0.2223 - loss: 81
1.7304 - val_accuracy: 0.8911 - val_auc_4: 0.6444 - val_f1_score: 0.1965 - val_loss: 897.2170
Epoch 13/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 782us/step - accuracy: 0.8750 - auc_4: 0.6653 - f1_score: 0.2223 - loss: 80
2.1935 - val_accuracy: 0.8911 - val_auc_4: 0.6492 - val_f1_score: 0.1965 - val_loss: 887.6238
Epoch 14/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 767us/step - accuracy: 0.8750 - auc_4: 0.6691 - f1_score: 0.2223 - loss: 79
2.9376 - val_accuracy: 0.8911 - val_auc_4: 0.6502 - val_f1_score: 0.1965 - val_loss: 878.4175
Epoch 15/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 740us/step - accuracy: 0.8750 - auc_4: 0.6715 - f1_score: 0.2223 - loss: 78
3.9577 - val_accuracy: 0.8911 - val_auc_4: 0.6536 - val_f1_score: 0.1965 - val_loss: 869.5032
Epoch 16/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 727us/step - accuracy: 0.8750 - auc_4: 0.6736 - f1_score: 0.2223 - loss: 77
5.2225 - val_accuracy: 0.8911 - val_auc_4: 0.6525 - val_f1_score: 0.1965 - val_loss: 860.7593
Epoch 17/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 761us/step - accuracy: 0.8750 - auc_4: 0.6760 - f1_score: 0.2223 - loss: 76
6.7442 - val_accuracy: 0.8911 - val_auc_4: 0.6543 - val_f1_score: 0.1965 - val_loss: 852.2482
Epoch 18/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 697us/step - accuracy: 0.8750 - auc_4: 0.6779 - f1_score: 0.2223 - loss: 75
8.4991 - val_accuracy: 0.8911 - val_auc_4: 0.6568 - val_f1_score: 0.1965 - val_loss: 843.8710
Epoch 19/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 741us/step - accuracy: 0.8750 - auc_4: 0.6798 - f1_score: 0.2223 - loss: 75
0.4719 - val_accuracy: 0.8911 - val_auc_4: 0.6557 - val_f1_score: 0.1965 - val_loss: 835.7199
Epoch 20/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 731us/step - accuracy: 0.8750 - auc_4: 0.6812 - f1_score: 0.2223 - loss: 74
2.6714 - val_accuracy: 0.8911 - val_auc_4: 0.6536 - val_f1_score: 0.1965 - val_loss: 827.6408
Epoch 21/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 773us/step - accuracy: 0.8750 - auc_4: 0.6824 - f1_score: 0.2223 - loss: 73
5.1001 - val_accuracy: 0.8911 - val_auc_4: 0.6526 - val_f1_score: 0.1965 - val_loss: 819.6082
Epoch 22/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 784us/step - accuracy: 0.8750 - auc_4: 0.6839 - f1_score: 0.2223 - loss: 72
7.7625 - val_accuracy: 0.8911 - val_auc_4: 0.6543 - val_f1_score: 0.1965 - val_loss: 811.6664
Epoch 23/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 793us/step - accuracy: 0.8750 - auc_4: 0.6853 - f1_score: 0.2223 - loss: 72
0.6577 - val_accuracy: 0.8911 - val_auc_4: 0.6568 - val_f1_score: 0.1965 - val_loss: 804.0173
Epoch 24/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8750 - auc_4: 0.6868 - f1_score: 0.2223 - loss: 713.7
670 - val_accuracy: 0.8911 - val_auc_4: 0.6539 - val_f1_score: 0.1965 - val_loss: 796.3500
Epoch 25/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 768us/step - accuracy: 0.8750 - auc_4: 0.6879 - f1_score: 0.2223 - loss: 70
7.0864 - val_accuracy: 0.8911 - val_auc_4: 0.6539 - val_f1_score: 0.1965 - val_loss: 788.7993
Epoch 26/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 721us/step - accuracy: 0.8750 - auc_4: 0.6894 - f1_score: 0.2223 - loss: 70
0.6521 - val_accuracy: 0.8911 - val_auc_4: 0.6563 - val_f1_score: 0.1965 - val_loss: 781.4426
Epoch 27/200
```

```
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 702us/step - accuracy: 0.8750 - auc_4: 0.6904 - f1_score: 0.2223 - loss: 69
4.4930 - val_accuracy: 0.8911 - val_auc_4: 0.6556 - val_f1_score: 0.1965 - val_loss: 774.3315
Epoch 28/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 721us/step - accuracy: 0.8750 - auc_4: 0.6913 - f1_score: 0.2223 - loss: 68
8.5939 - val_accuracy: 0.8911 - val_auc_4: 0.6594 - val_f1_score: 0.1965 - val_loss: 767.5202
Epoch 29/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 730us/step - accuracy: 0.8750 - auc_4: 0.6924 - f1_score: 0.2223 - loss: 68
2.9404 - val_accuracy: 0.8911 - val_auc_4: 0.6589 - val_f1_score: 0.1965 - val_loss: 761.0078
Epoch 30/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 718us/step - accuracy: 0.8750 - auc_4: 0.6935 - f1_score: 0.2223 - loss: 67
7.5295 - val_accuracy: 0.8911 - val_auc_4: 0.6602 - val_f1_score: 0.1965 - val_loss: 754.5483
Epoch 31/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 709us/step - accuracy: 0.8750 - auc_4: 0.6948 - f1_score: 0.2223 - loss: 67
2.3798 - val_accuracy: 0.8911 - val_auc_4: 0.6614 - val_f1_score: 0.1965 - val_loss: 748.4949
Epoch 32/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 775us/step - accuracy: 0.8750 - auc_4: 0.6958 - f1_score: 0.2223 - loss: 66
7.4724 - val_accuracy: 0.8911 - val_auc_4: 0.6631 - val_f1_score: 0.1965 - val_loss: 742.7950
Epoch 33/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 703us/step - accuracy: 0.8750 - auc_4: 0.6972 - f1_score: 0.2223 - loss: 66
2.8398 - val_accuracy: 0.8911 - val_auc_4: 0.6620 - val_f1_score: 0.1965 - val_loss: 737.4274
Epoch 34/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 698us/step - accuracy: 0.8750 - auc_4: 0.6982 - f1_score: 0.2223 - loss: 65
8.4838 - val_accuracy: 0.8911 - val_auc_4: 0.6604 - val_f1_score: 0.1965 - val_loss: 732.2485
Epoch 35/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 736us/step - accuracy: 0.8750 - auc_4: 0.6993 - f1_score: 0.2223 - loss: 65
4.3916 - val_accuracy: 0.8911 - val_auc_4: 0.6608 - val_f1_score: 0.1965 - val_loss: 727.1181
Epoch 36/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 720us/step - accuracy: 0.8750 - auc_4: 0.7001 - f1_score: 0.2223 - loss: 65
0.4872 - val_accuracy: 0.8911 - val_auc_4: 0.6602 - val_f1_score: 0.1965 - val_loss: 722.0392
Epoch 37/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 707us/step - accuracy: 0.8750 - auc_4: 0.7013 - f1_score: 0.2223 - loss: 64
6.7524 - val_accuracy: 0.8911 - val_auc_4: 0.6632 - val_f1_score: 0.1965 - val_loss: 717.0428
Epoch 38/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 722us/step - accuracy: 0.8750 - auc_4: 0.7021 - f1_score: 0.2223 - loss: 64
3.1577 - val_accuracy: 0.8911 - val_auc_4: 0.6632 - val_f1_score: 0.1965 - val_loss: 712.3145
Epoch 39/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 737us/step - accuracy: 0.8750 - auc_4: 0.7029 - f1_score: 0.2223 - loss: 63
9.7089 - val_accuracy: 0.8911 - val_auc_4: 0.6647 - val_f1_score: 0.1965 - val_loss: 707.7613
Epoch 40/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 797us/step - accuracy: 0.8750 - auc_4: 0.7037 - f1_score: 0.2223 - loss: 63
6.4146 - val_accuracy: 0.8911 - val_auc_4: 0.6629 - val_f1_score: 0.1965 - val_loss: 703.2966
Epoch 41/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 815us/step - accuracy: 0.8750 - auc_4: 0.7046 - f1_score: 0.2223 - loss: 63
3.2495 - val_accuracy: 0.8911 - val_auc_4: 0.6611 - val_f1_score: 0.1965 - val_loss: 698.8715
Epoch 42/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8750 - auc_4: 0.7055 - f1_score: 0.2223 - loss: 630.1
923 - val_accuracy: 0.8911 - val_auc_4: 0.6603 - val_f1_score: 0.1965 - val_loss: 694.5441
Epoch 43/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 865us/step - accuracy: 0.8750 - auc_4: 0.7066 - f1_score: 0.2223 - loss: 62
7.2326 - val_accuracy: 0.8911 - val_auc_4: 0.6622 - val_f1_score: 0.1965 - val_loss: 690.2726
Epoch 44/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 694us/step - accuracy: 0.8750 - auc_4: 0.7074 - f1_score: 0.2223 - loss: 62
4.3751 - val_accuracy: 0.8911 - val_auc_4: 0.6669 - val_f1_score: 0.1965 - val_loss: 685.9086
Epoch 45/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 768us/step - accuracy: 0.8750 - auc_4: 0.7085 - f1_score: 0.2223 - loss: 62
1.6097 - val_accuracy: 0.8911 - val_auc_4: 0.6635 - val_f1_score: 0.1965 - val_loss: 681.4852
Epoch 46/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 728us/step - accuracy: 0.8750 - auc_4: 0.7095 - f1_score: 0.2223 - loss: 61
8.9382 - val_accuracy: 0.8911 - val_auc_4: 0.6632 - val_f1_score: 0.1965 - val_loss: 677.1259
Epoch 47/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 743us/step - accuracy: 0.8750 - auc_4: 0.7103 - f1_score: 0.2223 - loss: 61
6.3609 - val_accuracy: 0.8911 - val_auc_4: 0.6651 - val_f1_score: 0.1965 - val_loss: 672.8178
Epoch 48/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 788us/step - accuracy: 0.8750 - auc_4: 0.7113 - f1_score: 0.2223 - loss: 61
3.8810 - val_accuracy: 0.8911 - val_auc_4: 0.6656 - val_f1_score: 0.1965 - val_loss: 668.6255
Epoch 49/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 738us/step - accuracy: 0.8750 - auc_4: 0.7122 - f1_score: 0.2223 - loss: 61
1.4866 - val_accuracy: 0.8911 - val_auc_4: 0.6641 - val_f1_score: 0.1965 - val_loss: 664.5699
Epoch 50/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 810us/step - accuracy: 0.8750 - auc_4: 0.7132 - f1_score: 0.2223 - loss: 60
9.1716 - val_accuracy: 0.8911 - val_auc_4: 0.6676 - val_f1_score: 0.1965 - val_loss: 660.4726
Epoch 51/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 755us/step - accuracy: 0.8750 - auc_4: 0.7137 - f1_score: 0.2223 - loss: 60
6.9265 - val_accuracy: 0.8911 - val_auc_4: 0.6672 - val_f1_score: 0.1965 - val_loss: 656.4532
Epoch 52/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 717us/step - accuracy: 0.8750 - auc_4: 0.7145 - f1_score: 0.2223 - loss: 60
4.7449 - val_accuracy: 0.8911 - val_auc_4: 0.6666 - val_f1_score: 0.1965 - val_loss: 652.4829
Epoch 53/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 742us/step - accuracy: 0.8750 - auc_4: 0.7154 - f1_score: 0.2223 - loss: 60
```

```
2.6267 - val_accuracy: 0.8911 - val_auc_4: 0.6681 - val_f1_score: 0.1965 - val_loss: 648.5151
Epoch 54/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 729us/step - accuracy: 0.8750 - auc_4: 0.7159 - f1_score: 0.2223 - loss: 60
0.5743 - val_accuracy: 0.8911 - val_auc_4: 0.6709 - val_f1_score: 0.1965 - val_loss: 644.5064
Epoch 55/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 744us/step - accuracy: 0.8750 - auc_4: 0.7168 - f1_score: 0.2223 - loss: 59
8.5731 - val_accuracy: 0.8911 - val_auc_4: 0.6702 - val_f1_score: 0.1965 - val_loss: 640.5411
Epoch 56/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8750 - auc_4: 0.7175 - f1_score: 0.2223 - loss: 596.6
133 - val_accuracy: 0.8911 - val_auc_4: 0.6713 - val_f1_score: 0.1965 - val_loss: 636.7484
Epoch 57/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 895us/step - accuracy: 0.8750 - auc_4: 0.7183 - f1_score: 0.2223 - loss: 59
4.6987 - val_accuracy: 0.8911 - val_auc_4: 0.6694 - val_f1_score: 0.1965 - val_loss: 633.0470
Epoch 58/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 885us/step - accuracy: 0.8750 - auc_4: 0.7190 - f1_score: 0.2223 - loss: 59
2.8211 - val_accuracy: 0.8911 - val_auc_4: 0.6702 - val_f1_score: 0.1965 - val_loss: 629.4686
Epoch 59/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 766us/step - accuracy: 0.8750 - auc_4: 0.7197 - f1_score: 0.2223 - loss: 59
0.9929 - val_accuracy: 0.8911 - val_auc_4: 0.6677 - val_f1_score: 0.1965 - val_loss: 625.9331
Epoch 60/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 754us/step - accuracy: 0.8750 - auc_4: 0.7204 - f1_score: 0.2223 - loss: 58
9.2036 - val_accuracy: 0.8911 - val_auc_4: 0.6677 - val_f1_score: 0.1965 - val_loss: 622.4139
Epoch 61/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 770us/step - accuracy: 0.8750 - auc_4: 0.7211 - f1_score: 0.2223 - loss: 58
7.4493 - val_accuracy: 0.8911 - val_auc_4: 0.6675 - val_f1_score: 0.1965 - val_loss: 619.0092
Epoch 62/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 746us/step - accuracy: 0.8750 - auc_4: 0.7218 - f1_score: 0.2223 - loss: 58
5.7355 - val_accuracy: 0.8911 - val_auc_4: 0.6669 - val_f1_score: 0.1965 - val_loss: 615.5430
Epoch 63/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 745us/step - accuracy: 0.8750 - auc_4: 0.7225 - f1_score: 0.2223 - loss: 58
4.0563 - val_accuracy: 0.8911 - val_auc_4: 0.6657 - val_f1_score: 0.1965 - val_loss: 612.1453
Epoch 64/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 725us/step - accuracy: 0.8750 - auc_4: 0.7233 - f1_score: 0.2223 - loss: 58
2.4098 - val_accuracy: 0.8911 - val_auc_4: 0.6682 - val_f1_score: 0.1965 - val_loss: 608.8386
Epoch 65/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 748us/step - accuracy: 0.8750 - auc_4: 0.7240 - f1_score: 0.2223 - loss: 58
0.7983 - val_accuracy: 0.8911 - val_auc_4: 0.6695 - val_f1_score: 0.1965 - val_loss: 605.6873
Epoch 66/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 839us/step - accuracy: 0.8750 - auc_4: 0.7250 - f1_score: 0.2223 - loss: 57
9.2125 - val_accuracy: 0.8911 - val_auc_4: 0.6722 - val_f1_score: 0.1965 - val_loss: 602.6767
Epoch 67/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 995us/step - accuracy: 0.8750 - auc_4: 0.7257 - f1_score: 0.2223 - loss: 57
7.6552 - val_accuracy: 0.8911 - val_auc_4: 0.6709 - val_f1_score: 0.1965 - val_loss: 599.7070
Epoch 68/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 894us/step - accuracy: 0.8750 - auc_4: 0.7269 - f1_score: 0.2223 - loss: 57
6.1352 - val_accuracy: 0.8911 - val_auc_4: 0.6710 - val_f1_score: 0.1965 - val_loss: 596.8080
Epoch 69/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 858us/step - accuracy: 0.8750 - auc_4: 0.7277 - f1_score: 0.2223 - loss: 57
4.6472 - val_accuracy: 0.8911 - val_auc_4: 0.6721 - val_f1_score: 0.1965 - val_loss: 593.9603
Epoch 70/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 768us/step - accuracy: 0.8750 - auc_4: 0.7284 - f1_score: 0.2223 - loss: 57
3.1807 - val_accuracy: 0.8911 - val_auc_4: 0.6727 - val_f1_score: 0.1965 - val_loss: 591.1071
Epoch 71/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 757us/step - accuracy: 0.8750 - auc_4: 0.7293 - f1_score: 0.2223 - loss: 57
1.7453 - val_accuracy: 0.8911 - val_auc_4: 0.6714 - val_f1_score: 0.1965 - val_loss: 588.3195
Epoch 72/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 765us/step - accuracy: 0.8750 - auc_4: 0.7300 - f1_score: 0.2223 - loss: 57
0.3401 - val_accuracy: 0.8911 - val_auc_4: 0.6714 - val_f1_score: 0.1965 - val_loss: 585.6163
Epoch 73/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 747us/step - accuracy: 0.8750 - auc_4: 0.7310 - f1_score: 0.2223 - loss: 56
8.9551 - val_accuracy: 0.8911 - val_auc_4: 0.6716 - val_f1_score: 0.1965 - val_loss: 582.9995
Epoch 74/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 729us/step - accuracy: 0.8750 - auc_4: 0.7320 - f1_score: 0.2223 - loss: 56
7.6018 - val_accuracy: 0.8911 - val_auc_4: 0.6724 - val_f1_score: 0.1965 - val_loss: 580.3709
Epoch 75/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 726us/step - accuracy: 0.8750 - auc_4: 0.7329 - f1_score: 0.2223 - loss: 56
6.2757 - val_accuracy: 0.8911 - val_auc_4: 0.6731 - val_f1_score: 0.1965 - val_loss: 577.7936
Epoch 76/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 699us/step - accuracy: 0.8750 - auc_4: 0.7335 - f1_score: 0.2223 - loss: 56
4.9736 - val_accuracy: 0.8911 - val_auc_4: 0.6735 - val_f1_score: 0.1965 - val_loss: 575.2816
Epoch 77/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 724us/step - accuracy: 0.8750 - auc_4: 0.7340 - f1_score: 0.2223 - loss: 56
3.6965 - val_accuracy: 0.8911 - val_auc_4: 0.6719 - val_f1_score: 0.1965 - val_loss: 572.8641
Epoch 78/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 960us/step - accuracy: 0.8750 - auc_4: 0.7349 - f1_score: 0.2223 - loss: 56
2.4411 - val_accuracy: 0.8911 - val_auc_4: 0.6745 - val_f1_score: 0.1965 - val_loss: 570.5334
Epoch 79/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 794us/step - accuracy: 0.8750 - auc_4: 0.7355 - f1_score: 0.2223 - loss: 56
1.2058 - val_accuracy: 0.8911 - val_auc_4: 0.6743 - val_f1_score: 0.1965 - val_loss: 568.3072
```

```
Epoch 80/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 844us/step - accuracy: 0.8750 - auc_4: 0.7361 - f1_score: 0.2223 - loss: 55
9.9832 - val_accuracy: 0.8911 - val_auc_4: 0.6730 - val_f1_score: 0.1965 - val_loss: 566.1485
Epoch 81/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 826us/step - accuracy: 0.8750 - auc_4: 0.7367 - f1_score: 0.2223 - loss: 55
8.7756 - val_accuracy: 0.8911 - val_auc_4: 0.6709 - val_f1_score: 0.1965 - val_loss: 564.0312
Epoch 82/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 708us/step - accuracy: 0.8750 - auc_4: 0.7376 - f1_score: 0.2223 - loss: 55
7.5821 - val_accuracy: 0.8911 - val_auc_4: 0.6722 - val_f1_score: 0.1965 - val_loss: 561.9153
Epoch 83/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 664us/step - accuracy: 0.8750 - auc_4: 0.7384 - f1_score: 0.2223 - loss: 55
6.3986 - val_accuracy: 0.8911 - val_auc_4: 0.6716 - val_f1_score: 0.1965 - val_loss: 559.9485
Epoch 84/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 838us/step - accuracy: 0.8750 - auc_4: 0.7391 - f1_score: 0.2223 - loss: 55
5.2306 - val_accuracy: 0.8911 - val_auc_4: 0.6724 - val_f1_score: 0.1965 - val_loss: 558.0300
Epoch 85/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 726us/step - accuracy: 0.8750 - auc_4: 0.7402 - f1_score: 0.2223 - loss: 55
4.0790 - val_accuracy: 0.8911 - val_auc_4: 0.6760 - val_f1_score: 0.1965 - val_loss: 556.0998
Epoch 86/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 732us/step - accuracy: 0.8750 - auc_4: 0.7408 - f1_score: 0.2223 - loss: 55
2.9378 - val_accuracy: 0.8911 - val_auc_4: 0.6757 - val_f1_score: 0.1965 - val_loss: 554.2283
Epoch 87/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 724us/step - accuracy: 0.8750 - auc_4: 0.7415 - f1_score: 0.2223 - loss: 55
1.8156 - val_accuracy: 0.8911 - val_auc_4: 0.6767 - val_f1_score: 0.1965 - val_loss: 552.3509
Epoch 88/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 719us/step - accuracy: 0.8750 - auc_4: 0.7423 - f1_score: 0.2223 - loss: 55
0.7051 - val_accuracy: 0.8911 - val_auc_4: 0.6774 - val_f1_score: 0.1965 - val_loss: 550.5029
Epoch 89/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 687us/step - accuracy: 0.8750 - auc_4: 0.7431 - f1_score: 0.2223 - loss: 54
9.6049 - val_accuracy: 0.8911 - val_auc_4: 0.6775 - val_f1_score: 0.1965 - val_loss: 548.7502
Epoch 90/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 733us/step - accuracy: 0.8750 - auc_4: 0.7439 - f1_score: 0.2223 - loss: 54
8.5195 - val_accuracy: 0.8911 - val_auc_4: 0.6778 - val_f1_score: 0.1965 - val_loss: 547.0430
Epoch 91/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 699us/step - accuracy: 0.8750 - auc_4: 0.7447 - f1_score: 0.2223 - loss: 54
7.4476 - val_accuracy: 0.8911 - val_auc_4: 0.6788 - val_f1_score: 0.1965 - val_loss: 545.3835
Epoch 92/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 733us/step - accuracy: 0.8750 - auc_4: 0.7453 - f1_score: 0.2223 - loss: 54
6.3885 - val_accuracy: 0.8911 - val_auc_4: 0.6785 - val_f1_score: 0.1965 - val_loss: 543.6931
Epoch 93/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 778us/step - accuracy: 0.8751 - auc_4: 0.7460 - f1_score: 0.2223 - loss: 54
5.3415 - val_accuracy: 0.8911 - val_auc_4: 0.6785 - val_f1_score: 0.1965 - val_loss: 542.0182
Epoch 94/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 908us/step - accuracy: 0.8751 - auc_4: 0.7467 - f1_score: 0.2223 - loss: 54
4.3083 - val_accuracy: 0.8911 - val_auc_4: 0.6769 - val_f1_score: 0.1965 - val_loss: 540.2875
Epoch 95/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 944us/step - accuracy: 0.8751 - auc_4: 0.7474 - f1_score: 0.2223 - loss: 54
3.2853 - val_accuracy: 0.8911 - val_auc_4: 0.6758 - val_f1_score: 0.1965 - val_loss: 538.5621
Epoch 96/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 981us/step - accuracy: 0.8751 - auc_4: 0.7482 - f1_score: 0.2223 - loss: 54
2.2791 - val_accuracy: 0.8911 - val_auc_4: 0.6759 - val_f1_score: 0.1965 - val_loss: 536.8636
Epoch 97/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 823us/step - accuracy: 0.8751 - auc_4: 0.7491 - f1_score: 0.2223 - loss: 54
1.2900 - val_accuracy: 0.8911 - val_auc_4: 0.6767 - val_f1_score: 0.1965 - val_loss: 535.1469
Epoch 98/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 795us/step - accuracy: 0.8751 - auc_4: 0.7499 - f1_score: 0.2223 - loss: 54
0.3122 - val_accuracy: 0.8911 - val_auc_4: 0.6768 - val_f1_score: 0.1965 - val_loss: 533.4470
Epoch 99/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 763us/step - accuracy: 0.8751 - auc_4: 0.7506 - f1_score: 0.2223 - loss: 53
9.3460 - val_accuracy: 0.8911 - val_auc_4: 0.6777 - val_f1_score: 0.1965 - val_loss: 531.7258
Epoch 100/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 733us/step - accuracy: 0.8751 - auc_4: 0.7515 - f1_score: 0.2223 - loss: 53
8.3873 - val_accuracy: 0.8911 - val_auc_4: 0.6779 - val_f1_score: 0.1965 - val_loss: 530.0265
Epoch 101/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 703us/step - accuracy: 0.8751 - auc_4: 0.7521 - f1_score: 0.2223 - loss: 53
7.4406 - val_accuracy: 0.8911 - val_auc_4: 0.6774 - val_f1_score: 0.1965 - val_loss: 528.3421
Epoch 102/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 766us/step - accuracy: 0.8751 - auc_4: 0.7529 - f1_score: 0.2223 - loss: 53
6.5018 - val_accuracy: 0.8911 - val_auc_4: 0.6777 - val_f1_score: 0.1965 - val_loss: 526.6680
Epoch 103/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 706us/step - accuracy: 0.8751 - auc_4: 0.7536 - f1_score: 0.2223 - loss: 53
5.5745 - val_accuracy: 0.8911 - val_auc_4: 0.6786 - val_f1_score: 0.1965 - val_loss: 524.9967
Epoch 104/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 676us/step - accuracy: 0.8751 - auc_4: 0.7544 - f1_score: 0.2223 - loss: 53
4.6587 - val_accuracy: 0.8911 - val_auc_4: 0.6818 - val_f1_score: 0.1965 - val_loss: 523.3652
Epoch 105/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 701us/step - accuracy: 0.8751 - auc_4: 0.7552 - f1_score: 0.2223 - loss: 53
3.7543 - val_accuracy: 0.8911 - val_auc_4: 0.6812 - val_f1_score: 0.1965 - val_loss: 521.7517
Epoch 106/200
```

```
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 874us/step - accuracy: 0.8751 - auc_4: 0.7558 - f1_score: 0.2223 - loss: 53
2.8565 - val_accuracy: 0.8911 - val_auc_4: 0.6822 - val_f1_score: 0.1965 - val_loss: 520.1176
Epoch 107/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 879us/step - accuracy: 0.8751 - auc_4: 0.7565 - f1_score: 0.2223 - loss: 53
1.9703 - val_accuracy: 0.8911 - val_auc_4: 0.6827 - val_f1_score: 0.1965 - val_loss: 518.5396
Epoch 108/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 922us/step - accuracy: 0.8751 - auc_4: 0.7573 - f1_score: 0.2223 - loss: 53
1.0963 - val_accuracy: 0.8911 - val_auc_4: 0.6819 - val_f1_score: 0.1965 - val_loss: 516.9844
Epoch 109/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 770us/step - accuracy: 0.8753 - auc_4: 0.7579 - f1_score: 0.2223 - loss: 53
0.2303 - val_accuracy: 0.8911 - val_auc_4: 0.6819 - val_f1_score: 0.1965 - val_loss: 515.4613
Epoch 110/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 728us/step - accuracy: 0.8753 - auc_4: 0.7587 - f1_score: 0.2223 - loss: 52
9.3700 - val_accuracy: 0.8911 - val_auc_4: 0.6832 - val_f1_score: 0.1965 - val_loss: 513.9733
Epoch 111/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 701us/step - accuracy: 0.8753 - auc_4: 0.7594 - f1_score: 0.2223 - loss: 52
8.5153 - val_accuracy: 0.8911 - val_auc_4: 0.6840 - val_f1_score: 0.1965 - val_loss: 512.5312
Epoch 112/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 723us/step - accuracy: 0.8755 - auc_4: 0.7602 - f1_score: 0.2223 - loss: 52
7.6667 - val_accuracy: 0.8911 - val_auc_4: 0.6823 - val_f1_score: 0.1965 - val_loss: 511.1270
Epoch 113/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 696us/step - accuracy: 0.8756 - auc_4: 0.7610 - f1_score: 0.2223 - loss: 52
6.8254 - val_accuracy: 0.8911 - val_auc_4: 0.6820 - val_f1_score: 0.1965 - val_loss: 509.7452
Epoch 114/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 746us/step - accuracy: 0.8756 - auc_4: 0.7616 - f1_score: 0.2223 - loss: 52
5.9894 - val_accuracy: 0.8911 - val_auc_4: 0.6820 - val_f1_score: 0.1965 - val_loss: 508.3095
Epoch 115/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 715us/step - accuracy: 0.8757 - auc_4: 0.7625 - f1_score: 0.2223 - loss: 52
5.1589 - val_accuracy: 0.8911 - val_auc_4: 0.6818 - val_f1_score: 0.1965 - val_loss: 506.8862
Epoch 116/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 716us/step - accuracy: 0.8757 - auc_4: 0.7632 - f1_score: 0.2223 - loss: 52
4.3339 - val_accuracy: 0.8911 - val_auc_4: 0.6827 - val_f1_score: 0.1965 - val_loss: 505.4649
Epoch 117/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 759us/step - accuracy: 0.8757 - auc_4: 0.7639 - f1_score: 0.2223 - loss: 52
3.5137 - val_accuracy: 0.8911 - val_auc_4: 0.6839 - val_f1_score: 0.1965 - val_loss: 504.0258
Epoch 118/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 679us/step - accuracy: 0.8757 - auc_4: 0.7646 - f1_score: 0.2223 - loss: 52
2.6963 - val_accuracy: 0.8911 - val_auc_4: 0.6845 - val_f1_score: 0.1965 - val_loss: 502.5916
Epoch 119/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 766us/step - accuracy: 0.8757 - auc_4: 0.7653 - f1_score: 0.2223 - loss: 52
1.8833 - val_accuracy: 0.8911 - val_auc_4: 0.6853 - val_f1_score: 0.1965 - val_loss: 501.1460
Epoch 120/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 911us/step - accuracy: 0.8757 - auc_4: 0.7660 - f1_score: 0.2223 - loss: 52
1.0762 - val_accuracy: 0.8911 - val_auc_4: 0.6845 - val_f1_score: 0.1965 - val_loss: 499.7105
Epoch 121/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 895us/step - accuracy: 0.8758 - auc_4: 0.7666 - f1_score: 0.2223 - loss: 52
0.2744 - val_accuracy: 0.8911 - val_auc_4: 0.6837 - val_f1_score: 0.1965 - val_loss: 498.2331
Epoch 122/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 916us/step - accuracy: 0.8758 - auc_4: 0.7675 - f1_score: 0.2223 - loss: 51
9.4793 - val_accuracy: 0.8911 - val_auc_4: 0.6841 - val_f1_score: 0.1965 - val_loss: 496.7709
Epoch 123/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 835us/step - accuracy: 0.8758 - auc_4: 0.7684 - f1_score: 0.2223 - loss: 51
8.6899 - val_accuracy: 0.8911 - val_auc_4: 0.6835 - val_f1_score: 0.1965 - val_loss: 495.3561
Epoch 124/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 770us/step - accuracy: 0.8759 - auc_4: 0.7692 - f1_score: 0.2223 - loss: 51
7.9068 - val_accuracy: 0.8911 - val_auc_4: 0.6839 - val_f1_score: 0.1965 - val_loss: 493.9756
Epoch 125/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 751us/step - accuracy: 0.8760 - auc_4: 0.7698 - f1_score: 0.2223 - loss: 51
7.1296 - val_accuracy: 0.8911 - val_auc_4: 0.6853 - val_f1_score: 0.1965 - val_loss: 492.5779
Epoch 126/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 756us/step - accuracy: 0.8760 - auc_4: 0.7705 - f1_score: 0.2223 - loss: 51
6.3582 - val_accuracy: 0.8911 - val_auc_4: 0.6859 - val_f1_score: 0.1965 - val_loss: 491.2317
Epoch 127/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 791us/step - accuracy: 0.8760 - auc_4: 0.7714 - f1_score: 0.2223 - loss: 51
5.5905 - val_accuracy: 0.8911 - val_auc_4: 0.6870 - val_f1_score: 0.1965 - val_loss: 489.8914
Epoch 128/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 731us/step - accuracy: 0.8761 - auc_4: 0.7721 - f1_score: 0.2223 - loss: 51
4.8262 - val_accuracy: 0.8911 - val_auc_4: 0.6901 - val_f1_score: 0.1965 - val_loss: 488.6101
Epoch 129/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 722us/step - accuracy: 0.8763 - auc_4: 0.7726 - f1_score: 0.2223 - loss: 51
4.0675 - val_accuracy: 0.8911 - val_auc_4: 0.6907 - val_f1_score: 0.1965 - val_loss: 487.3263
Epoch 130/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 714us/step - accuracy: 0.8764 - auc_4: 0.7734 - f1_score: 0.2223 - loss: 51
3.3159 - val_accuracy: 0.8911 - val_auc_4: 0.6894 - val_f1_score: 0.1965 - val_loss: 486.0417
Epoch 131/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 723us/step - accuracy: 0.8765 - auc_4: 0.7740 - f1_score: 0.2223 - loss: 51
2.5686 - val_accuracy: 0.8911 - val_auc_4: 0.6892 - val_f1_score: 0.1965 - val_loss: 484.7619
Epoch 132/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 787us/step - accuracy: 0.8765 - auc_4: 0.7746 - f1_score: 0.2223 - loss: 51
```

```
1.8248 - val_accuracy: 0.8911 - val_auc_4: 0.6899 - val_f1_score: 0.1965 - val_loss: 483.5048
Epoch 133/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 834us/step - accuracy: 0.8768 - auc_4: 0.7754 - f1_score: 0.2223 - loss: 51
1.0856 - val_accuracy: 0.8911 - val_auc_4: 0.6886 - val_f1_score: 0.1965 - val_loss: 482.2527
Epoch 134/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 911us/step - accuracy: 0.8768 - auc_4: 0.7762 - f1_score: 0.2223 - loss: 51
0.3516 - val_accuracy: 0.8911 - val_auc_4: 0.6874 - val_f1_score: 0.1965 - val_loss: 481.0274
Epoch 135/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 837us/step - accuracy: 0.8769 - auc_4: 0.7770 - f1_score: 0.2223 - loss: 50
9.6215 - val_accuracy: 0.8911 - val_auc_4: 0.6875 - val_f1_score: 0.1965 - val_loss: 479.7998
Epoch 136/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 688us/step - accuracy: 0.8769 - auc_4: 0.7774 - f1_score: 0.2223 - loss: 50
8.8944 - val_accuracy: 0.8911 - val_auc_4: 0.6887 - val_f1_score: 0.1965 - val_loss: 478.5957
Epoch 137/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 742us/step - accuracy: 0.8769 - auc_4: 0.7781 - f1_score: 0.2223 - loss: 50
8.1693 - val_accuracy: 0.8911 - val_auc_4: 0.6901 - val_f1_score: 0.1965 - val_loss: 477.4048
Epoch 138/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 772us/step - accuracy: 0.8769 - auc_4: 0.7788 - f1_score: 0.2223 - loss: 50
7.4474 - val_accuracy: 0.8911 - val_auc_4: 0.6902 - val_f1_score: 0.1965 - val_loss: 476.2098
Epoch 139/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 711us/step - accuracy: 0.8770 - auc_4: 0.7797 - f1_score: 0.2223 - loss: 50
6.7299 - val_accuracy: 0.8911 - val_auc_4: 0.6916 - val_f1_score: 0.1965 - val_loss: 475.0405
Epoch 140/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 745us/step - accuracy: 0.8770 - auc_4: 0.7804 - f1_score: 0.2223 - loss: 50
6.0157 - val_accuracy: 0.8911 - val_auc_4: 0.6939 - val_f1_score: 0.1965 - val_loss: 473.8836
Epoch 141/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 745us/step - accuracy: 0.8773 - auc_4: 0.7809 - f1_score: 0.2223 - loss: 50
5.3062 - val_accuracy: 0.8911 - val_auc_4: 0.6932 - val_f1_score: 0.1965 - val_loss: 472.6931
Epoch 142/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 751us/step - accuracy: 0.8774 - auc_4: 0.7815 - f1_score: 0.2223 - loss: 50
4.6022 - val_accuracy: 0.8911 - val_auc_4: 0.6936 - val_f1_score: 0.1965 - val_loss: 471.5343
Epoch 143/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 716us/step - accuracy: 0.8776 - auc_4: 0.7824 - f1_score: 0.2223 - loss: 50
3.9036 - val_accuracy: 0.8911 - val_auc_4: 0.6926 - val_f1_score: 0.1965 - val_loss: 470.4199
Epoch 144/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 703us/step - accuracy: 0.8777 - auc_4: 0.7832 - f1_score: 0.2223 - loss: 50
3.2133 - val_accuracy: 0.8911 - val_auc_4: 0.6924 - val_f1_score: 0.1965 - val_loss: 469.3541
Epoch 145/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 714us/step - accuracy: 0.8777 - auc_4: 0.7837 - f1_score: 0.2223 - loss: 50
2.5296 - val_accuracy: 0.8911 - val_auc_4: 0.6923 - val_f1_score: 0.1965 - val_loss: 468.2781
Epoch 146/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 747us/step - accuracy: 0.8777 - auc_4: 0.7843 - f1_score: 0.2223 - loss: 50
1.8504 - val_accuracy: 0.8911 - val_auc_4: 0.6933 - val_f1_score: 0.1965 - val_loss: 467.2048
Epoch 147/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 763us/step - accuracy: 0.8779 - auc_4: 0.7848 - f1_score: 0.2223 - loss: 50
1.1753 - val_accuracy: 0.8911 - val_auc_4: 0.6948 - val_f1_score: 0.1965 - val_loss: 466.1560
Epoch 148/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 913us/step - accuracy: 0.8779 - auc_4: 0.7855 - f1_score: 0.2223 - loss: 50
0.5045 - val_accuracy: 0.8911 - val_auc_4: 0.6943 - val_f1_score: 0.1965 - val_loss: 465.1253
Epoch 149/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 862us/step - accuracy: 0.8780 - auc_4: 0.7863 - f1_score: 0.2223 - loss: 49
9.8403 - val_accuracy: 0.8911 - val_auc_4: 0.6950 - val_f1_score: 0.1965 - val_loss: 464.0952
Epoch 150/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 732us/step - accuracy: 0.8780 - auc_4: 0.7870 - f1_score: 0.2223 - loss: 49
9.1810 - val_accuracy: 0.8911 - val_auc_4: 0.6937 - val_f1_score: 0.1965 - val_loss: 463.0412
Epoch 151/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 721us/step - accuracy: 0.8781 - auc_4: 0.7876 - f1_score: 0.2223 - loss: 49
8.5278 - val_accuracy: 0.8911 - val_auc_4: 0.6928 - val_f1_score: 0.1965 - val_loss: 461.9830
Epoch 152/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 740us/step - accuracy: 0.8780 - auc_4: 0.7883 - f1_score: 0.2223 - loss: 49
7.8779 - val_accuracy: 0.8911 - val_auc_4: 0.6940 - val_f1_score: 0.1965 - val_loss: 460.9261
Epoch 153/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 715us/step - accuracy: 0.8781 - auc_4: 0.7889 - f1_score: 0.2223 - loss: 49
7.2327 - val_accuracy: 0.8911 - val_auc_4: 0.6946 - val_f1_score: 0.1965 - val_loss: 459.8964
Epoch 154/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 742us/step - accuracy: 0.8781 - auc_4: 0.7894 - f1_score: 0.2223 - loss: 49
6.5919 - val_accuracy: 0.8911 - val_auc_4: 0.6952 - val_f1_score: 0.1965 - val_loss: 458.8661
Epoch 155/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 715us/step - accuracy: 0.8782 - auc_4: 0.7899 - f1_score: 0.2223 - loss: 49
5.9534 - val_accuracy: 0.8911 - val_auc_4: 0.6950 - val_f1_score: 0.1965 - val_loss: 457.8288
Epoch 156/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 858us/step - accuracy: 0.8783 - auc_4: 0.7904 - f1_score: 0.2223 - loss: 49
5.3203 - val_accuracy: 0.8911 - val_auc_4: 0.6942 - val_f1_score: 0.1965 - val_loss: 456.8294
Epoch 157/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 733us/step - accuracy: 0.8783 - auc_4: 0.7911 - f1_score: 0.2223 - loss: 49
4.6945 - val_accuracy: 0.8911 - val_auc_4: 0.6957 - val_f1_score: 0.1965 - val_loss: 455.8544
Epoch 158/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 682us/step - accuracy: 0.8783 - auc_4: 0.7916 - f1_score: 0.2223 - loss: 49
4.0734 - val_accuracy: 0.8911 - val_auc_4: 0.6967 - val_f1_score: 0.1965 - val_loss: 454.8779
```

```
Epoch 159/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 704us/step - accuracy: 0.8784 - auc_4: 0.7923 - f1_score: 0.2223 - loss: 49
3.4548 - val_accuracy: 0.8911 - val_auc_4: 0.6965 - val_f1_score: 0.1965 - val_loss: 453.9065
Epoch 160/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 689us/step - accuracy: 0.8786 - auc_4: 0.7929 - f1_score: 0.2223 - loss: 49
2.8388 - val_accuracy: 0.8911 - val_auc_4: 0.6980 - val_f1_score: 0.1965 - val_loss: 452.9412
Epoch 161/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 701us/step - accuracy: 0.8786 - auc_4: 0.7934 - f1_score: 0.2223 - loss: 49
2.2263 - val_accuracy: 0.8911 - val_auc_4: 0.6979 - val_f1_score: 0.1965 - val_loss: 451.9995
Epoch 162/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 782us/step - accuracy: 0.8787 - auc_4: 0.7941 - f1_score: 0.2223 - loss: 49
1.6163 - val_accuracy: 0.8911 - val_auc_4: 0.6986 - val_f1_score: 0.1965 - val_loss: 451.0758
Epoch 163/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 824us/step - accuracy: 0.8788 - auc_4: 0.7946 - f1_score: 0.2223 - loss: 49
1.0078 - val_accuracy: 0.8911 - val_auc_4: 0.6972 - val_f1_score: 0.1965 - val_loss: 450.1538
Epoch 164/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 869us/step - accuracy: 0.8788 - auc_4: 0.7952 - f1_score: 0.2223 - loss: 49
0.4018 - val_accuracy: 0.8911 - val_auc_4: 0.6972 - val_f1_score: 0.1965 - val_loss: 449.2050
Epoch 165/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 847us/step - accuracy: 0.8791 - auc_4: 0.7957 - f1_score: 0.2223 - loss: 48
9.7994 - val_accuracy: 0.8911 - val_auc_4: 0.6981 - val_f1_score: 0.1965 - val_loss: 448.2885
Epoch 166/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 672us/step - accuracy: 0.8791 - auc_4: 0.7962 - f1_score: 0.2223 - loss: 48
9.2017 - val_accuracy: 0.8911 - val_auc_4: 0.6980 - val_f1_score: 0.1965 - val_loss: 447.3829
Epoch 167/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 695us/step - accuracy: 0.8792 - auc_4: 0.7969 - f1_score: 0.2223 - loss: 48
8.6079 - val_accuracy: 0.8911 - val_auc_4: 0.6987 - val_f1_score: 0.1965 - val_loss: 446.4762
Epoch 168/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 687us/step - accuracy: 0.8793 - auc_4: 0.7976 - f1_score: 0.2223 - loss: 48
8.0173 - val_accuracy: 0.8911 - val_auc_4: 0.6988 - val_f1_score: 0.1965 - val_loss: 445.6349
Epoch 169/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 665us/step - accuracy: 0.8793 - auc_4: 0.7982 - f1_score: 0.2223 - loss: 48
7.4306 - val_accuracy: 0.8911 - val_auc_4: 0.6987 - val_f1_score: 0.1965 - val_loss: 444.7992
Epoch 170/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 704us/step - accuracy: 0.8794 - auc_4: 0.7987 - f1_score: 0.2223 - loss: 48
6.8464 - val_accuracy: 0.8911 - val_auc_4: 0.6980 - val_f1_score: 0.1965 - val_loss: 443.9853
Epoch 171/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 674us/step - accuracy: 0.8794 - auc_4: 0.7992 - f1_score: 0.2223 - loss: 48
6.2657 - val_accuracy: 0.8911 - val_auc_4: 0.6998 - val_f1_score: 0.1965 - val_loss: 443.1895
Epoch 172/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 728us/step - accuracy: 0.8795 - auc_4: 0.7998 - f1_score: 0.2223 - loss: 48
5.6914 - val_accuracy: 0.8911 - val_auc_4: 0.6996 - val_f1_score: 0.1965 - val_loss: 442.4209
Epoch 173/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 725us/step - accuracy: 0.8792 - auc_4: 0.8004 - f1_score: 0.2223 - loss: 48
5.1237 - val_accuracy: 0.8911 - val_auc_4: 0.6995 - val_f1_score: 0.1965 - val_loss: 441.6616
Epoch 174/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 709us/step - accuracy: 0.8793 - auc_4: 0.8009 - f1_score: 0.2223 - loss: 48
4.5595 - val_accuracy: 0.8911 - val_auc_4: 0.6994 - val_f1_score: 0.1965 - val_loss: 440.9122
Epoch 175/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 762us/step - accuracy: 0.8794 - auc_4: 0.8015 - f1_score: 0.2223 - loss: 48
3.9978 - val_accuracy: 0.8911 - val_auc_4: 0.7004 - val_f1_score: 0.1965 - val_loss: 440.1702
Epoch 176/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 707us/step - accuracy: 0.8796 - auc_4: 0.8020 - f1_score: 0.2223 - loss: 48
3.4404 - val_accuracy: 0.8911 - val_auc_4: 0.7007 - val_f1_score: 0.1965 - val_loss: 439.4500
Epoch 177/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 821us/step - accuracy: 0.8798 - auc_4: 0.8024 - f1_score: 0.2223 - loss: 48
2.8857 - val_accuracy: 0.8911 - val_auc_4: 0.7014 - val_f1_score: 0.1965 - val_loss: 438.7498
Epoch 178/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 792us/step - accuracy: 0.8798 - auc_4: 0.8029 - f1_score: 0.2223 - loss: 48
2.3343 - val_accuracy: 0.8911 - val_auc_4: 0.7003 - val_f1_score: 0.1965 - val_loss: 438.0641
Epoch 179/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 862us/step - accuracy: 0.8800 - auc_4: 0.8034 - f1_score: 0.2223 - loss: 48
1.7875 - val_accuracy: 0.8911 - val_auc_4: 0.7007 - val_f1_score: 0.1965 - val_loss: 437.3773
Epoch 180/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 865us/step - accuracy: 0.8802 - auc_4: 0.8037 - f1_score: 0.2223 - loss: 48
1.2446 - val_accuracy: 0.8911 - val_auc_4: 0.7000 - val_f1_score: 0.1965 - val_loss: 436.7099
Epoch 181/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 719us/step - accuracy: 0.8804 - auc_4: 0.8041 - f1_score: 0.2223 - loss: 48
0.7070 - val_accuracy: 0.8911 - val_auc_4: 0.7013 - val_f1_score: 0.1965 - val_loss: 436.0567
Epoch 182/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 723us/step - accuracy: 0.8808 - auc_4: 0.8046 - f1_score: 0.2223 - loss: 48
0.1716 - val_accuracy: 0.8911 - val_auc_4: 0.7016 - val_f1_score: 0.1965 - val_loss: 435.4322
Epoch 183/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 754us/step - accuracy: 0.8808 - auc_4: 0.8052 - f1_score: 0.2223 - loss: 47
9.6404 - val_accuracy: 0.8911 - val_auc_4: 0.7022 - val_f1_score: 0.1965 - val_loss: 434.8204
Epoch 184/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 772us/step - accuracy: 0.8809 - auc_4: 0.8056 - f1_score: 0.2223 - loss: 47
9.1115 - val_accuracy: 0.8911 - val_auc_4: 0.7024 - val_f1_score: 0.1965 - val_loss: 434.2200
Epoch 185/200
```

```
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 707us/step - accuracy: 0.8813 - auc_4: 0.8061 - f1_score: 0.2223 - loss: 47
8.5851 - val_accuracy: 0.8911 - val_auc_4: 0.7021 - val_f1_score: 0.1965 - val_loss: 433.6268
Epoch 186/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 701us/step - accuracy: 0.8815 - auc_4: 0.8066 - f1_score: 0.2223 - loss: 47
8.0610 - val_accuracy: 0.8911 - val_auc_4: 0.7013 - val_f1_score: 0.1965 - val_loss: 433.0467
Epoch 187/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 691us/step - accuracy: 0.8817 - auc_4: 0.8072 - f1_score: 0.2223 - loss: 47
7.5401 - val_accuracy: 0.8911 - val_auc_4: 0.7015 - val_f1_score: 0.1965 - val_loss: 432.4832
Epoch 188/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 718us/step - accuracy: 0.8817 - auc_4: 0.8076 - f1_score: 0.2223 - loss: 47
7.0223 - val_accuracy: 0.8911 - val_auc_4: 0.7022 - val_f1_score: 0.1965 - val_loss: 431.9301
Epoch 189/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 755us/step - accuracy: 0.8818 - auc_4: 0.8081 - f1_score: 0.2223 - loss: 47
6.5076 - val_accuracy: 0.8911 - val_auc_4: 0.7017 - val_f1_score: 0.1965 - val_loss: 431.3957
Epoch 190/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 827us/step - accuracy: 0.8818 - auc_4: 0.8085 - f1_score: 0.2223 - loss: 47
5.9964 - val_accuracy: 0.8911 - val_auc_4: 0.7023 - val_f1_score: 0.1965 - val_loss: 430.8746
Epoch 191/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 921us/step - accuracy: 0.8818 - auc_4: 0.8088 - f1_score: 0.2223 - loss: 47
5.4896 - val_accuracy: 0.8911 - val_auc_4: 0.7018 - val_f1_score: 0.1965 - val_loss: 430.3592
Epoch 192/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 845us/step - accuracy: 0.8819 - auc_4: 0.8092 - f1_score: 0.2223 - loss: 47
4.9857 - val_accuracy: 0.8911 - val_auc_4: 0.7019 - val_f1_score: 0.1965 - val_loss: 429.8518
Epoch 193/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 724us/step - accuracy: 0.8819 - auc_4: 0.8096 - f1_score: 0.2223 - loss: 47
4.4852 - val_accuracy: 0.8911 - val_auc_4: 0.7030 - val_f1_score: 0.1965 - val_loss: 429.3652
Epoch 194/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 738us/step - accuracy: 0.8822 - auc_4: 0.8101 - f1_score: 0.2223 - loss: 47
3.9878 - val_accuracy: 0.8911 - val_auc_4: 0.7033 - val_f1_score: 0.1965 - val_loss: 428.9048
Epoch 195/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 727us/step - accuracy: 0.8823 - auc_4: 0.8105 - f1_score: 0.2223 - loss: 47
3.4945 - val_accuracy: 0.8911 - val_auc_4: 0.7036 - val_f1_score: 0.1965 - val_loss: 428.4672
Epoch 196/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 732us/step - accuracy: 0.8823 - auc_4: 0.8110 - f1_score: 0.2223 - loss: 47
3.0092 - val_accuracy: 0.8907 - val_auc_4: 0.7031 - val_f1_score: 0.1965 - val_loss: 428.0544
Epoch 197/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 729us/step - accuracy: 0.8824 - auc_4: 0.8113 - f1_score: 0.2223 - loss: 47
2.5286 - val_accuracy: 0.8907 - val_auc_4: 0.7029 - val_f1_score: 0.1965 - val_loss: 427.6608
Epoch 198/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 710us/step - accuracy: 0.8823 - auc_4: 0.8118 - f1_score: 0.2223 - loss: 47
2.0521 - val_accuracy: 0.8907 - val_auc_4: 0.7025 - val_f1_score: 0.1965 - val_loss: 427.2706
Epoch 199/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 732us/step - accuracy: 0.8824 - auc_4: 0.8122 - f1_score: 0.2223 - loss: 47
1.5804 - val_accuracy: 0.8907 - val_auc_4: 0.7023 - val_f1_score: 0.1965 - val_loss: 426.8867
Epoch 200/200
419/419 ━━━━━━━━━━━━━━━━━━━━ 0s 692us/step - accuracy: 0.8824 - auc_4: 0.8128 - f1_score: 0.2223 - loss: 47
1.1126 - val_accuracy: 0.8907 - val_auc_4: 0.7027 - val_f1_score: 0.1965 - val_loss: 426.5211
```

In [44]:
```
update_summary(summary_df,
               'Stacked NN',
               y_train,
               stacked_model.predict(X_train),
```

```
                y_val,
                stacked_model.predict(X_val),
                y_test,
                stacked_model.predict(X_test),
                class1_only=True)
summary_df
```

```
837/837 ━━━━━━━━━━━━━━━━━━━━ 0s 320us/step
93/93 ━━━━━━━━━━━━━━━━━━━━ 0s 283us/step
93/93 ━━━━━━━━━━━━━━━━━━━━ 0s 347us/step
```

Out[44]:

	Model	Train AUC	Val AUC	Test AUC
0	Logit as Benchmark	0.6707	0.6637	0.6637
1	LASSO Logit	0.6835	0.6791	0.6791
2	Random Forest CV	0.9686	0.7086	0.7086
3	GBM CV	0.8193	0.7133	0.7133
4	Sigmoid NN	0.6578	0.6321	0.6611
5	Softmax NN	0.6555	0.6314	0.6611
6	SMOTE NN	0.6879	0.6579	0.6740
7	Stacked NN	0.8086	0.7031	0.7031

Fortunately, it seems like the combination works. The stacked model improves upon the other networks quite significantly. However, its performance is still worse than the normal gradient boosting model.

Hybrid Model

This model is also a combination of the gradient boosting model and the oversampling with smote using sigmoid activation neural network. The configuration here is a bit more intricate. The main model itself is actually a logit model, but the predictions from the gradient boosting and sigmoid neural network are used as input. The logit model is cross validated to get the best regularization parameter.

In [45]:
```
# split train, val, test again with engineered features
outcome = news_df["is_popular"]
# features = news_df[high_performance_predictors]
features = news_df.drop(columns=exclude_cols)
# features = news_df.drop(columns=['timedelta', 'is_popular', 'article_id'])
prng = np.random.RandomState(42)
X_train, X_test, y_train, y_test = train_test_split(features, outcome, test_size=0.1, random_state=prng)
X_val, y_val = X_test, y_test
# X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.1, random_state=prng)

# normalize data
scaler = MinMaxScaler(feature_range=(-1, 1))
# scaler = StandardScaler()
# scaler.fit(features)
columns_not_to_scale = [col for col in X_train.columns if col not in binary_cols]
scaler.fit(X_train[columns_not_to_scale])

X_train[columns_not_to_scale] = scaler.transform(X_train[columns_not_to_scale])
X_val[columns_not_to_scale] = scaler.transform(X_val[columns_not_to_scale])
X_test[columns_not_to_scale] = scaler.transform(X_test[columns_not_to_scale])
```

In [46]:
```
def get_hybrid_data(ml_model, dl_model, ml_data, dl_data):
    ml_pred = ml_model.predict_proba(ml_data)
    dl_pred = dl_model.predict(dl_data)
    if dl_model.layers[-1].units == 1:
        df = pd.DataFrame({
            'ml_model_0': ml_pred[:,0],
            'ml_model_1': ml_pred[:,1],
            'dl_model': dl_pred.flatten()
        })
    else:
        df = pd.DataFrame({
            'ml_model_0': ml_pred[:,0],
            'ml_model_1': ml_pred[:,1],
            'dl_model_0': dl_pred[:,0],
            'dl_model_1': dl_pred[:,1]
        })
    return df
```

In [47]:
```python
X_train_hybrid, X_val_hybrid, X_test_hybrid = get_hybrid_data(gbm_model, smote_model, X_ori_sets[0][gbm_hig

# no regularisation needed so setting the parameter to very high value
lambdas = list(10**np.arange(-1, -3.01, -1/3))
n_obs = len(X_train_hybrid)
Cs_values = [1/(l*n_obs) for l in lambdas]
scoring='roc_auc'

# hybrid_model = LogisticRegressionCV(
#         Cs=Cs_values,
#         penalty='elasticnet',
#         l1_ratios=[0, 0.3, 0.5],
#         refit=True,
#         scoring=scoring,
#         solver="saga",
#         tol=1e-7,
#         random_state=prng,
#         class_weight=None
#     )

hybrid_model = LogisticRegressionCV(
        Cs=Cs_values,
        refit=True,
        scoring=scoring,
        solver="newton-cg",
        tol=1e-7,
        random_state=prng,
        class_weight=None
    )

# hybrid_model = LogisticRegressionCV(
#         Cs=Cs_values,
#         penalty='l2',
#         refit=True,
#         scoring=scoring,
#         solver="liblinear",
#         tol=1e-7,
#         random_state=prng,
#         class_weight=None
#     )

hybrid_model.fit(X_train_hybrid, y_ori_sets[0])

update_summary(summary_df,
               'Hybrid Model',
               y_ori_sets[0],
               hybrid_model.predict_proba(X_train_hybrid),
               y_ori_sets[1],
               hybrid_model.predict_proba(X_val_hybrid),
               y_ori_sets[2],
               hybrid_model.predict_proba(X_test_hybrid))
summary_df
```

```
837/837 ━━━━━━━━━━━━━━━━━━━━ 0s 452us/step
93/93 ━━━━━━━━━━━━━━━━━━━━ 0s 358us/step
93/93 ━━━━━━━━━━━━━━━━━━━━ 0s 315us/step
```

Out[47]:

	Model	Train AUC	Val AUC	Test AUC
0	Logit as Benchmark	0.6707	0.6637	0.6637
1	LASSO Logit	0.6835	0.6791	0.6791
2	Random Forest CV	0.9686	0.7086	0.7086
3	GBM CV	0.8193	0.7133	0.7133
4	Sigmoid NN	0.6578	0.6321	0.6611
5	Softmax NN	0.6555	0.6314	0.6611
6	SMOTE NN	0.6879	0.6579	0.6740
7	Stacked NN	0.8086	0.7031	0.7031
8	Hybrid Model	0.8161	0.7132	0.7132

In [48]:
```python
hybrid_model.coef_
```

Out[48]:
```
array([[-4.18936505,  4.18936505,  0.29459347]])
```

The hybrid model's performance are very close to the performance of the gradient boosting model. However, it seems like the inclusion of the neural network actually cost this hybrid model. Hence, it is still better to use the gradient boosting model for prediction.

Voting Classifier

This is another composite model, inspired by the Voting Classifier. The implementation here is a customized work to adapt the models in this notebook. Essentially, the input for this model is the predictions from the random forest, gradient boosting and the oversampling neural network. The prediction is then calculated by averaging all the probabilities predicted by the input models.

In [49]:
```
ml_models = {
    'Random Forest CV': rf_model,
    'GBM CV': gbm_model,
}
dl_models = {
    'SMOTE NN': smote_model
}

class CustomVotingClassifier:

    def __init__(self, ml_models: dict, dl_models: dict, voting='soft'):
        self.ml_models = ml_models
        self.dl_models = dl_models
        self.voting = voting

    def predict_proba(self, ml_data, dl_data):
        if 'Sigmoid NN' in self.dl_models.keys() or 'SMOTE NN' in self.dl_models.keys():
            return self.predict_proba_class1(ml_data, dl_data)
        else:
            class_0_pred = {}
            class_1_pred = {}
            for model_name, model in self.ml_models.items():
                if model_name == 'GBM CV':
                    model_pred = model.predict_proba(ml_data[gbm_high_perm])
                    class_0_pred[model_name] = model_pred[:,0]
                    class_1_pred[model_name] = model_pred[:,1]
                elif model_name == 'Random Forest CV':
                    model_pred = model.predict_proba(ml_data[rf_high_perm])
                    class_0_pred[model_name] = model_pred[:,0]
                    class_1_pred[model_name] = model_pred[:,1]
                else:
                    model_pred = model.predict_proba(ml_data)
                    class_0_pred[model_name] = model_pred[:,0]
                    class_1_pred[model_name] = model_pred[:,1]
            for model_name, model in self.dl_models.items():
                model_pred = model.predict(dl_data)
                class_0_pred[model_name] = model_pred[:,0]
                class_1_pred[model_name] = model_pred[:,1]

            pred_0_df = pd.DataFrame.from_dict(class_0_pred)
            pred_1_df = pd.DataFrame.from_dict(class_1_pred)

            # pred_df = pd.DataFrame({
            #     0: pred_0_df.mean(axis=1).to_numpy(),
            #     1: pred_1_df.mean(axis=1).to_numpy()
            # })
            return np.vstack((pred_0_df.mean(axis=1).to_numpy(), pred_1_df.mean(axis=1).to_numpy())).T

    def predict_proba_class1(self, ml_data, dl_data):
        class_1_pred = {}
        for model_name, model in self.ml_models.items():
            if model_name == 'GBM CV':
                model_pred = model.predict_proba(ml_data[gbm_high_perm])
                class_1_pred[model_name] = model_pred[:,1]
            elif model_name == 'Random Forest CV':
                model_pred = model.predict_proba(ml_data[rf_high_perm])
                class_1_pred[model_name] = model_pred[:,1]
            else:
                model_pred = model.predict_proba(ml_data)
                class_1_pred[model_name] = model_pred[:,1]
        for model_name, model in self.dl_models.items():
            model_pred = model.predict(dl_data)
            if model.layers[-1].units == 1:
```

```python
                class_1_pred[model_name] = model_pred.flatten()
            else:
                class_1_pred[model_name] = model_pred[:,1]

        pred_1_df = pd.DataFrame.from_dict(class_1_pred)

        # pred_df = pd.DataFrame({
        #     0: pred_0_df.mean(axis=1).to_numpy(),
        #     1: pred_1_df.mean(axis=1).to_numpy()
        # })
        return pred_1_df.mean(axis=1).to_numpy()

    def predict(self, ml_data, dl_data):
        model_pred = {}
        if self.voting == 'soft':
            pred_df = self.predict_proba(ml_data, dl_data)
            pred_df = pd.DataFrame({
                0: pred_df[:,0],
                1: pred_df[:,1]
            })
            pred_df = pred_df.idxmax(axis=1)
            return pred_df.to_numpy()
        elif self.voting == 'hard':
            for model_name, model in self.ml_models.items():
                model_pred[model_name] = model.predict(ml_data)
            for model_name, model in self.dl_models.items():
                model_pred[model_name] = np.argmax(model.predict(dl_data), axis=1)
            pred_df = pd.DataFrame.from_dict(model_pred)
            pred_df = pred_df.mode(axis=1)
            return pred_df.to_numpy()
        else:
            print('Invalid voting param!')
```

In [50]:

```python
soft_voting_clf = CustomVotingClassifier(ml_models, dl_models)
# hard_voting_clf = CustomVotingClassifier(ml_models, dl_models, 'hard')
update_summary(summary_df,
               'Soft Voting Classifier',
               y_ori_sets[0],
               soft_voting_clf.predict_proba(X_ori_sets[0], X_train),
               y_ori_sets[1],
               soft_voting_clf.predict_proba(X_ori_sets[1], X_val),
               y_ori_sets[2],
               soft_voting_clf.predict_proba(X_ori_sets[2], X_test),
               class1_only=True)
# update_summary(summary_df,
#                'Hard Voting Classifier',
#                y_ori_sets[0],
#                hard_voting_clf.predict_proba(X_ori_sets[0], X_train),
#                y_ori_sets[1],
#                hard_voting_clf.predict_proba(X_ori_sets[1], X_val),
#                y_ori_sets[2],
#                hard_voting_clf.predict_proba(X_ori_sets[2], X_test))
summary_df
```

```
837/837 ━━━━━━━━━━━━━━━━━━━━ 0s 332us/step
93/93 ━━━━━━━━━━━━━━━━━━━━ 0s 347us/step
93/93 ━━━━━━━━━━━━━━━━━━━━ 0s 296us/step
```

Out[50]:

	Model	Train AUC	Val AUC	Test AUC
0	Logit as Benchmark	0.6707	0.6637	0.6637
1	LASSO Logit	0.6835	0.6791	0.6791
2	Random Forest CV	0.9686	0.7086	0.7086
3	GBM CV	0.8193	0.7133	0.7133
4	Sigmoid NN	0.6578	0.6321	0.6611
5	Softmax NN	0.6555	0.6314	0.6611
6	SMOTE NN	0.6879	0.6579	0.6740
7	Stacked NN	0.8086	0.7031	0.7031
8	Hybrid Model	0.8161	0.7132	0.7132
9	Soft Voting Classifier	0.8616	0.6982	0.6982

Once again, like the hybrid model, although the performance of this model is better than all the single neural networks, it is not as good as the gradient boosting model. The average of all the best models' predictions in this notebook does not help improve the voting classifier performance.

Predict for the unseen data

Across all the models in this notebook, it seems like the gradient boosting model still reigns as the best performing model on the unseen data. Although there might be some other method or techniques that can be done to improve the neural network performance, I probably need more time and research to keep experimenting with them. Hence, for this competition, I choose the gradient boosting model's prediction as the main submission to compete.

In [51]:
```
unseen_df = pd.read_csv('online-news-popularity-ceu-ml-2024/test.csv')
```

In [52]:
```
unseen_df, _ = feature_engineer(unseen_df)
```

In [53]:
```
exclude_cols_test = exclude_cols.copy()
```

In [54]:
```
if 'is_popular' in exclude_cols_test:
    exclude_cols_test.remove('is_popular')
unseen_features = unseen_df.drop(columns=exclude_cols_test)
# unseen_features = unseen_df.drop(columns=['timedelta', 'article_id'])
predictions = gbm_model.predict_proba(unseen_features[gbm_high_perm])[:,1]
predictions
```

Out[54]:
```
array([0.18574096, 0.27410923, 0.08226444, ..., 0.07287442, 0.10838069,
       0.05157307])
```

In [56]:
```
if 'is_popular' in exclude_cols_test:
    exclude_cols_test.remove('is_popular')
unseen_features = unseen_df.drop(columns=exclude_cols_test)
# unseen_features = unseen_df.drop(columns=['timedelta', 'article_id'])
unseen_features[columns_not_to_scale] = scaler.transform(unseen_features[columns_not_to_scale])
predictions = smote_model.predict(unseen_features)
# predictions = weighted_deep3_model.predict(unseen_features)
# predictions = predictions[:, 1]
# predictions[:30]
predictions
```

```
310/310 ━━━━━━━━━━━━━━━━━━━━ 0s 311us/step
```

Out[56]:
```
array([[0.3461162 ],
       [0.6753606 ],
       [0.32546496],
       ...,
       [0.25698286],
       [0.29663742],
       [0.24091348]], dtype=float32)
```

In [57]:
```
if 'is_popular' in exclude_cols_test:
    exclude_cols_test.remove('is_popular')
unseen_features = unseen_df.drop(columns=exclude_cols_test)
# unseen_features = unseen_df.drop(columns=['timedelta', 'article_id'])
unseen_features[columns_not_to_scale] = scaler.transform(unseen_features[columns_not_to_scale])
unseen_features = np.hstack((unseen_features, gbm_model.predict_proba(unseen_features[gbm_high_perm])))
predictions = stacked_model.predict(unseen_features)
# predictions = weighted_deep3_model.predict(unseen_features)
# predictions = predictions[:, 1]
# predictions[:30]
predictions
```

```
310/310 ━━━━━━━━━━━━━━━━━━━━ 0s 318us/step
```

Out[57]:
```
array([[0.09388109],
       [0.17497267],
       [0.09604242],
       ...,
       [0.08925827],
       [0.1022398 ],
       [0.0962301 ]], dtype=float32)
```

In [58]:
```
if 'is_popular' in exclude_cols_test:
    exclude_cols_test.remove('is_popular')
unseen_features = unseen_df.drop(columns=exclude_cols_test)
# unseen_features = unseen_df.drop(columns=['timedelta', 'article_id'])
X_ml_unseen = unseen_features.copy()
X_dl_unseen = unseen_features.copy()
```

```
X_dl_unseen[columns_not_to_scale] = scaler.transform(X_dl_unseen[columns_not_to_scale])
predictions = hybrid_model.predict_proba(get_hybrid_data(gbm_model, smote_model, X_ml_unseen[gbm_high_perm]
# predictions = soft_voting_clf.predict_proba(X_ml_unseen, X_dl_unseen)
predictions
```

```
310/310 ━━━━━━━━━━━━━━━━━━━━ 0s 306us/step
```

Out[58]:
```
array([0.16378751, 0.31154511, 0.07562028, ..., 0.06899381, 0.09169855,
       0.05811489])
```

In [55]:
```
prediction_df = pd.DataFrame({
    'article_id': unseen_df.article_id,
    'score': predictions.flatten()
})
# prediction_df[(prediction_df.score > 0.6)]
prediction_df.to_csv('submission.csv', index=False)
```