competition_code_final

April 1, 2024

0.1 CS506 Midterm Kaggle Competition Final Code

0.1.1 Name: Youxuan Ma, U23330522

0.1.2 Kaggle Username: MarkMa316

[1]:
```
from sklearn.preprocessing import LabelEncoder
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from geopy.distance import geodesic
from sklearn.cluster import KMeans

# Calculate distance from home function
def calculate_distance(row):
    home_location = (row['lat'], row['long'])
    merch_location = (row['merch_lat'], row['merch_long'])
    return geodesic(home_location, merch_location).miles

# Function to calculate distance between two points
def calculate_distance2(row1, row2):
    point1 = (row1['lat'], row1['long'])
    point2 = (row2['lat'], row2['long'])
    return geodesic(point1, point2).miles

def calculate_similarity_score(amount, fraud_mean, fraud_std, normal_mean,␣
 ↪normal_std):
    # Calculate Z-scores for fraud and normal
    z_score_fraud = abs((amount - fraud_mean) / fraud_std)
    z_score_normal = abs((amount - normal_mean) / normal_std)

    # Invert the Z-scores to get similarity scores
    fraud_similarity = 1 / (1 + z_score_fraud)
    normal_similarity = 1 / (1 + z_score_normal)

    return fraud_similarity, normal_similarity

def process(df):
```

```
    # Rearrange the rows
    df['original_order'] = range(df.shape[0])

    df['trans_date_trans_time'] = pd.to_datetime(df['trans_date_trans_time'],␣
↪format='%d/%m/%Y %H:%M')
    df['dob'] = pd.to_datetime(df['dob'], format='%d/%m/%Y')

    df.sort_values(by=['cc_num', 'trans_date_trans_time'], inplace=True)

    # Calculate the time difference between transactions
    df['Time_Delta'] = df.groupby('cc_num')['trans_date_trans_time'].diff().dt.
↪total_seconds() / 60.0  # Time delta in minutes
    df['Time_Delta'] = df['Time_Delta'].fillna(value=0)

    # Shift the latitude and longitude to get the previous transaction's␣
↪location
    df['prev_lat'] = df.groupby('cc_num')['merch_lat'].shift(1)
    df['prev_long'] = df.groupby('cc_num')['merch_long'].shift(1)

    # Calculate the distance to the previous transaction
    df['distance_to_prev'] = df.apply(
        lambda row: calculate_distance2(
            {'lat': row['merch_lat'], 'long': row['merch_long']},
            {'lat': row['prev_lat'], 'long': row['prev_long']}
        ) if not pd.isnull(row['prev_lat']) else None,
        axis=1
    )
    df['distance_to_prev'] = df['distance_to_prev'].fillna(value=0)

    # Calculate location consistency as the inverse of the average distance to␣
↪previous transactions (higher value means more consistency)
    df['location_consistency'] = 100 / df.groupby('cc_num')['distance_to_prev'].
↪transform('mean')

    # Time-based features
    df['hour'] = df['trans_date_trans_time'].dt.hour
    df['day_of_week'] = df['trans_date_trans_time'].dt.dayofweek
    df['month'] = df['trans_date_trans_time'].dt.month
    df['day_of_month'] = df['trans_date_trans_time'].dt.day

    # Age of the account holder
    df['age'] = (df['trans_date_trans_time'] - df['dob']).dt.days // 365

    # Calculate the transaction distance from home
    df['dist_to_home'] = df.apply(calculate_distance, axis=1)
```

```
    # Group by category and calculate the mean and standard deviation of␣
↪transaction amounts
    category_stats = df.groupby('category')['amt'].agg(['mean', 'std']).
↪reset_index()
    # Merge these stats back into the main dataframe
    df = df.merge(category_stats, on='category', how='left')
    # Calculate z-score for each transaction amount within its category
    df['amt_anomaly_score_cat'] = ((df['amt'] - df['mean']) / df['std'])
    df.drop(columns=['mean', 'std'], inplace=True)
    # Group by merchant and calculate the mean and standard deviation of␣
↪transaction amounts
    merchant_stats = df.groupby('merchant')['amt'].agg(['mean', 'std']).
↪reset_index()
    # Merge these stats back into the main dataframe
    df = df.merge(merchant_stats, on='merchant', how='left')
    # Calculate z-score for each transaction amount within its merchant
    df['amt_anomaly_score_merch'] = ((df['amt'] - df['mean']) / df['std'])
    df.drop(columns=['mean', 'std'], inplace=True)

    # Calculate the historical average transaction amount for each user
    avg_amt_per_user = df.groupby('cc_num')['amt'].transform('mean').
↪rename('avg_amt_per_user')

    # Append this feature to the dataset
    df['amt_relative_avg'] = (abs(df['amt'] - avg_amt_per_user) /␣
↪avg_amt_per_user)

    kmeans = KMeans(n_clusters=12, random_state=42)

    # Create a new column for the city pop cluster labels
    df['city_pop_cluster'] = kmeans.fit_predict(df[['city_pop']])

    df.drop(columns=['trans_date_trans_time', 'lat', 'long', 'merch_lat',␣
↪'merch_long'], inplace=True)
    df.drop(columns=['prev_lat', 'prev_long'], inplace=True)

    # Identify categorical columns to encode
    categorical_cols = ['merchant', 'category', 'gender', 'city', 'state',␣
↪'job']

    mappings = {}

    label_encoder = LabelEncoder()
    for col in categorical_cols:
        df[col] = label_encoder.fit_transform(df[col])
```

```
        mappings[col] = {label: index for index, label in␣
↪enumerate(label_encoder.classes_)}

    # Calculate the fraud rate by category
    fraud_rate_by_category = df.groupby('category')['is_fraud'].mean().
↪reset_index()
    fraud_rate_by_category.rename(columns={'is_fraud': 'fraud_rate_cat'},␣
↪inplace=True)

    # Merge the fraud rate back into the main DataFrame
    df = pd.merge(df, fraud_rate_by_category[['category', 'fraud_rate_cat']],␣
↪on='category', how='left')

    # Calculate the fraud rate by merchant
    fraud_rate_by_merchant = df.groupby('merchant')['is_fraud'].mean().
↪reset_index()
    fraud_rate_by_merchant.rename(columns={'is_fraud': 'fraud_rate_merch'},␣
↪inplace=True)

    # Merge the fraud rate back into the main DataFrame
    df = pd.merge(df, fraud_rate_by_merchant[['merchant', 'fraud_rate_merch']],␣
↪on='merchant', how='left')

    # Separate the transactions by fraud and normal
    fraud_trans = df[df['is_fraud'] == 1]['amt']
    normal_trans = df[df['is_fraud'] == 0]['amt']

    # Calculate statistics
    fraud_mean, fraud_std = fraud_trans.mean(), fraud_trans.std()
    normal_mean, normal_std = normal_trans.mean(), normal_trans.std()

    v_calculate_similarity_score = np.vectorize(calculate_similarity_score)

    # Apply the function
    df['fraud_similarity'], df['normal_similarity'] =␣
↪v_calculate_similarity_score(
        df['amt'],
        fraud_mean, fraud_std,
        normal_mean, normal_std
    )

    # Sort the dataset back to its original order
    df.sort_values(by='original_order', inplace=True)
    df.drop(columns='original_order', inplace=True)
    df.drop(columns='cc_num', inplace=True)
```

```
    return df, mappings

trainingSet = pd.read_csv("./data/train.csv")
submissionSet = pd.read_csv("./data/test.csv")
train_processed, cat_map = process(trainingSet)
train_processed.drop(columns=['first', 'last', 'street', 'dob', 'zip',␣
 ↪'trans_num', 'unix_time'], inplace=True)

# Merge on Id so that the test set can have feature columns as well
test_df = pd.merge(train_processed, submissionSet, left_on='Id', right_on='Id')
test_df = test_df.drop(columns=['is_fraud_x'])
test_df = test_df.rename(columns={'is_fraud_y': 'is_fraud'})

# The training set is where the score is not null
train_df = train_processed[train_processed['is_fraud'].notnull()]

# Save the datasets with the new features
test_df.to_csv("./data/final_processed_test.csv", index=False)
train_df.to_csv("./data/final_processed_train.csv", index=False)
```

[2]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Turned out the 'city_pop_cluster' feature was not useful
X = train_df.drop(['is_fraud', 'Id', 'city_pop_cluster'], axis=1)
y = train_df['is_fraud']

num_cols = ['amt', 'Time_Delta', 'distance_to_prev', 'location_consistency',␣
 ↪'dist_to_home', 'amt_anomaly_score_cat', 'amt_anomaly_score_merch',␣
 ↪'amt_relative_avg', 'fraud_rate_cat', 'fraud_rate_merch',␣
 ↪'fraud_similarity', 'normal_similarity']

scaler = StandardScaler()
X[num_cols] = scaler.fit_transform(X[num_cols])
test_df[num_cols] = scaler.transform(test_df[num_cols])

# Split the data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,␣
 ↪stratify=y, random_state=42)
```

[ ]:
```
from sklearn.model_selection import GridSearchCV
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report, f1_score,␣
 ↪make_scorer

# Initialize the XGBClassifier
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
```

```
# Define the parameter grid
param_grid = {
    'max_depth': [3, 5, 7, 10, None],
    'learning_rate': [0.05, 0.1, 0.15, 0.2],
    'n_estimators': [300, 500, 900],
    'colsample_bytree': [0.3, 0.7, 0.9]
}

# Set up GridSearchCV to find the best hyperparameters
grid_search = GridSearchCV(estimator=xgb, param_grid=param_grid,␣
 ↪scoring='f1_micro', cv=5, n_jobs=-1, verbose=3)
grid_search.fit(X_train, y_train)
clf = grid_search.best_estimator_
# Best parameters and best score
print("Best parameters found: ", grid_search.best_params_)
print("Best average F1 score found: ", f1_score(y_val, clf.predict(X_val)))
```

[3]:
```
from sklearn.model_selection import GridSearchCV
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report, f1_score,␣
 ↪make_scorer
# Initialize the XGBClassifier with the best hyperparameters
xgb_clf = XGBClassifier(colsample_bytree=0.9, learning_rate=0.1, max_depth=5,␣
 ↪n_estimators=900, use_label_encoder=False, eval_metric='logloss')

xgb_clf.fit(X_train, y_train)

y_pred_xgb = xgb_clf.predict(X_val)
print(f1_score(y_val, y_pred_xgb))
print(classification_report(y_val, y_pred_xgb))
```

```
0.9085714285714286
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     96876
         1.0       0.98      0.85      0.91       375

    accuracy                           1.00     97251
   macro avg       0.99      0.92      0.95     97251
weighted avg       1.00      1.00      1.00     97251
```

[4]:
```
from sklearn.model_selection import GridSearchCV
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report, f1_score,␣
 ↪make_scorer
```

```
# Initialize the XGBClassifier with another set of hyperparameters
xgb_clf2 = XGBClassifier(colsample_bytree=0.9, learning_rate=0.15,␣
 ↪max_depth=None, n_estimators=900, use_label_encoder=False,␣
 ↪eval_metric='logloss')

xgb_clf2.fit(X_train, y_train)

y_pred_xgb2 = xgb_clf2.predict(X_val)
print(f1_score(y_val, y_pred_xgb2))
print(classification_report(y_val, y_pred_xgb2))
```

```
0.8981348637015782
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     96876
         1.0       0.97      0.83      0.90       375

    accuracy                           1.00     97251
   macro avg       0.99      0.92      0.95     97251
weighted avg       1.00      1.00      1.00     97251
```

[5]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, f1_score
from sklearn.model_selection import GridSearchCV
import pickle
from xgboost import XGBClassifier
import numpy as np
from sklearn.ensemble import BaggingClassifier

# Initialize the Bagging Classifier with the XGB Classifier as the base␣
 ↪estimator
bagging_clf = BaggingClassifier(estimator=xgb_clf, n_estimators=100,␣
 ↪random_state=42, n_jobs=-1, verbose=3)

bagging_clf.fit(X_train, y_train)
y_pred_bag = bagging_clf.predict(X_val)
print(f1_score(y_val, y_pred_bag))
```

```
[Parallel(n_jobs=16)]: Using backend LokyBackend with 16 concurrent workers.

Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 7 for this parallel run (total 100)…
```

```
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
```

```
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 7 of 7 for this parallel run (total 100)…

[Parallel(n_jobs=16)]: Done   3 out of  16 | elapsed:  2.4min remaining: 10.3min

Building estimator 7 of 7 for this parallel run (total 100)…
Building estimator 7 of 7 for this parallel run (total 100)…

[Parallel(n_jobs=16)]: Done   9 out of  16 | elapsed:  2.4min remaining:  1.9min
```

```
Building estimator 7 of 7 for this parallel run (total 100)…

[Parallel(n_jobs=16)]: Done  16 out of  16 | elapsed:  2.7min finished
[Parallel(n_jobs=16)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=16)]: Done   3 out of  16 | elapsed:    8.7s remaining:   37.6s
[Parallel(n_jobs=16)]: Done   9 out of  16 | elapsed:    9.0s remaining:    7.0s

0.8694362017804155

[Parallel(n_jobs=16)]: Done  16 out of  16 | elapsed:    9.7s finished
```

[ ]:
```python
from sklearn.ensemble import RandomForestClassifier

# Initialize a basic Random Forest model
rf = RandomForestClassifier(class_weight='balanced_subsample', random_state=40)

param_grid = {
    'n_estimators': [300, 500, 900],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'criterion': ['gini', 'entropy']
}

# Setup GridSearchCV to find the best hyperparameters
grid_search_rf = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5,␣
 ↪scoring='f1', n_jobs=-1, verbose=3)
grid_search_rf.fit(X_train, y_train)

# Best parameters and model
best_params = grid_search_rf.best_params_
print(f"Best parameters: {best_params}")

best_rf_model = grid_search_rf.best_estimator_
# Prediction and Evaluation
y_pred_rf = best_rf_model.predict(X_val)
print(f1_score(y_val, y_pred_rf))
print(classification_report(y_val, y_pred_rf))
```

[6]:
```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, f1_score
from sklearn.model_selection import GridSearchCV
import pickle
from xgboost import XGBClassifier
import numpy as np
from sklearn.ensemble import BaggingClassifier

# Initialize the Random Forest Classifier with the best hyperparameters
```

```
rf1 = RandomForestClassifier(class_weight='balanced_subsample',␣
 ↪n_estimators=300, max_depth=None, min_samples_split=10, min_samples_leaf=4,␣
 ↪criterion='entropy', n_jobs=-1, random_state=42)

rf1.fit(X_train, y_train)

y_pred_rf1 = rf1.predict(X_val)
print(f1_score(y_val, y_pred_rf1))
print(classification_report(y_val, y_pred_rf1))
```

```
0.8115107913669065
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     96876
         1.0       0.88      0.75      0.81       375

    accuracy                           1.00     97251
   macro avg       0.94      0.88      0.91     97251
weighted avg       1.00      1.00      1.00     97251
```

[7]:
```
# Test out another set of hyperparameters
rf2 = RandomForestClassifier(class_weight='balanced_subsample',␣
 ↪n_estimators=900, max_depth=None, min_samples_split=10, min_samples_leaf=1,␣
 ↪criterion='entropy', n_jobs=-1)

rf2.fit(X_train, y_train)

y_pred_rf2 = rf2.predict(X_val)
print(f1_score(y_val, y_pred_rf2))
print(classification_report(y_val, y_pred_rf2))
```

```
0.8036529680365296
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     96876
         1.0       0.94      0.70      0.80       375

    accuracy                           1.00     97251
   macro avg       0.97      0.85      0.90     97251
weighted avg       1.00      1.00      1.00     97251
```

[8]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, f1_score
from sklearn.model_selection import GridSearchCV
import pickle
from xgboost import XGBClassifier
```

```python
import numpy as np
from sklearn.ensemble import BaggingClassifier

# Initialize the Bagging Classifier for the Random Forest model
bagging_rf_clf = BaggingClassifier(estimator=rf1, n_estimators=100, n_jobs=-1,␣
 ↪verbose=3)

bagging_rf_clf.fit(X_train, y_train)
y_pred_bag_rf = bagging_rf_clf.predict(X_val)
print(f1_score(y_val, y_pred_bag_rf))
```

```
Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 7 for this parallel run (total 100)…
[Parallel(n_jobs=16)]: Using backend LokyBackend with 16 concurrent workers.
Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
```

```
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
```

```
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 7 of 7 for this parallel run (total 100)…
Building estimator 7 of 7 for this parallel run (total 100)…
Building estimator 7 of 7 for this parallel run (total 100)…
Building estimator 7 of 7 for this parallel run (total 100)…
```

```
[Parallel(n_jobs=16)]: Done   3 out of  16 | elapsed: 15.0min remaining: 64.9min
[Parallel(n_jobs=16)]: Done   9 out of  16 | elapsed: 15.1min remaining: 11.8min
[Parallel(n_jobs=16)]: Done  16 out of  16 | elapsed: 15.5min finished
[Parallel(n_jobs=16)]: Using backend LokyBackend with 16 concurrent workers.
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/joblib/externals/loky/process_executor.py:752: UserWarning: A worker
stopped while some jobs were given to the executor. This can be caused by a too
short worker timeout or by a memory leak.
  warnings.warn(
[Parallel(n_jobs=16)]: Done   3 out of  16 | elapsed:    2.9s remaining:   12.5s
[Parallel(n_jobs=16)]: Done   9 out of  16 | elapsed:    6.5s remaining:    5.1s
```

```
0.7867435158501441
```

```
[Parallel(n_jobs=16)]: Done  16 out of  16 | elapsed:    9.9s finished
```

[10]:
```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, f1_score
from sklearn.model_selection import GridSearchCV
import pickle
from xgboost import XGBClassifier
import numpy as np
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import VotingClassifier

# Initialize the Voting Classifier with the best models
voting_clf = VotingClassifier(
    estimators=[('xgb', xgb_clf), ('xgb2', xgb_clf2), ('rf', rf1), ('bag_xgb',␣
 ↪bagging_clf)],
    voting='soft',
    n_jobs=-1
```

```
)

vot_clf = voting_clf.fit(X_train, y_train)

y_pred_vote = vot_clf.predict(X_val)
f1_score_vote = f1_score(y_val, y_pred_vote)
print(f1_score_vote)
```

```
[Parallel(n_jobs=16)]: Using backend ThreadingBackend with 16 concurrent
workers.

Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
```

```
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
```

```
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 7 of 7 for this parallel run (total 100)…

[Parallel(n_jobs=16)]: Done   3 out of  16 | elapsed:  2.5min remaining: 11.0min
[Parallel(n_jobs=16)]: Done   9 out of  16 | elapsed:  2.5min remaining:  2.0min

Building estimator 7 of 7 for this parallel run (total 100)…
Building estimator 7 of 7 for this parallel run (total 100)…
Building estimator 7 of 7 for this parallel run (total 100)…

[Parallel(n_jobs=16)]: Done  16 out of  16 | elapsed:  2.8min finished
[Parallel(n_jobs=16)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=16)]: Done   3 out of  16 | elapsed:    8.5s remaining:   36.9s
[Parallel(n_jobs=16)]: Done   9 out of  16 | elapsed:    9.0s remaining:    7.0s

0.888243831640058

[Parallel(n_jobs=16)]: Done  16 out of  16 | elapsed:   10.0s finished
```

[18]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, f1_score
from sklearn.model_selection import GridSearchCV
import pickle
from xgboost import XGBClassifier
import numpy as np
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import VotingClassifier

voting_clf = VotingClassifier(
    estimators=[('xgb', xgb_clf), ('xgb2', xgb_clf2), ('rf', rf1), ('bag_xgb',␣
 ↪bagging_clf)],
    voting='soft',
    n_jobs=-1
)

# Train the ensemble model with the full training dataset
vot_clf = voting_clf.fit(X, y)
```

```
[Parallel(n_jobs=16)]: Using backend ThreadingBackend with 16 concurrent
workers.
```

```
Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
```

```
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
```

```
Building estimator 7 of 7 for this parallel run (total 100)…
Building estimator 7 of 7 for this parallel run (total 100)…
Building estimator 7 of 7 for this parallel run (total 100)…
Building estimator 7 of 7 for this parallel run (total 100)…

[Parallel(n_jobs=16)]: Done   3 out of  16 | elapsed:  3.2min remaining: 13.8min
[Parallel(n_jobs=16)]: Done   9 out of  16 | elapsed:  3.2min remaining:  2.5min
[Parallel(n_jobs=16)]: Done  16 out of  16 | elapsed:  3.5min finished
```

[11]:
```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, f1_score
from sklearn.model_selection import GridSearchCV
import pickle
from xgboost import XGBClassifier
import numpy as np
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import VotingClassifier

# Initialize the Voting Classifier with fewer models
voting_clf2 = VotingClassifier(
    estimators=[('xgb', xgb_clf), ('bag', bagging_clf), ('xgb2', xgb_clf2)],
    voting='soft',
    n_jobs=-1
)

vot_clf2 = voting_clf2.fit(X_train, y_train)

y_pred_vote2 = vot_clf2.predict(X_val)
f1_score_vote2 = f1_score(y_val, y_pred_vote2)
print(f1_score_vote2)
```

```
[Parallel(n_jobs=16)]: Using backend ThreadingBackend with 16 concurrent
workers.

Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 7 for this parallel run (total 100)…Building estimator
1 of 7 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…

Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
```

```
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
```

```
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…

[Parallel(n_jobs=16)]: Done   3 out of  16 | elapsed:  2.2min remaining:  9.6min

Building estimator 7 of 7 for this parallel run (total 100)…
Building estimator 7 of 7 for this parallel run (total 100)…

[Parallel(n_jobs=16)]: Done   9 out of  16 | elapsed:  2.3min remaining:  1.8min

Building estimator 7 of 7 for this parallel run (total 100)…
Building estimator 7 of 7 for this parallel run (total 100)…

[Parallel(n_jobs=16)]: Done  16 out of  16 | elapsed:  2.5min finished
[Parallel(n_jobs=16)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=16)]: Done   3 out of  16 | elapsed:    9.6s remaining:   41.8s
[Parallel(n_jobs=16)]: Done   9 out of  16 | elapsed:   10.0s remaining:    7.8s
[Parallel(n_jobs=16)]: Done  16 out of  16 | elapsed:   11.0s finished
```

0.8988439306358381

[16]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, f1_score
from sklearn.model_selection import GridSearchCV
import pickle
from xgboost import XGBClassifier
import numpy as np
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import VotingClassifier

voting_clf2 = VotingClassifier(
    estimators=[('xgb', xgb_clf), ('bag', bagging_clf), ('xgb2', xgb_clf2)],
    voting='soft',
    n_jobs=-1
)

# Train the second ensemble model with the full training dataset
vot_clf2 = voting_clf2.fit(X, y)
```

```
[Parallel(n_jobs=16)]: Using backend ThreadingBackend with 16 concurrent
workers.

Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 7 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 1 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
```

```
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 2 of 7 for this parallel run (total 100)…
Building estimator 2 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 3 of 7 for this parallel run (total 100)…
Building estimator 3 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 4 of 7 for this parallel run (total 100)…
Building estimator 4 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
```

```
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 5 of 7 for this parallel run (total 100)…
Building estimator 5 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 6 of 7 for this parallel run (total 100)…
Building estimator 6 of 6 for this parallel run (total 100)…
Building estimator 7 of 7 for this parallel run (total 100)…
Building estimator 7 of 7 for this parallel run (total 100)…

[Parallel(n_jobs=16)]: Done   3 out of  16 | elapsed:  2.9min remaining: 12.6min

Building estimator 7 of 7 for this parallel run (total 100)…

[Parallel(n_jobs=16)]: Done   9 out of  16 | elapsed:  2.9min remaining:  2.3min

Building estimator 7 of 7 for this parallel run (total 100)…

[Parallel(n_jobs=16)]: Done  16 out of  16 | elapsed:  3.3min finished
```

[19]:
```python
# Create Kaggle Submission 1
pred = test_df.drop(['is_fraud', 'city_pop_cluster', 'Id'], axis=1)
pred2 = test_df.drop(['is_fraud'], axis=1)

pred2['is_fraud'] = vot_clf.predict(pred)
pred2.is_fraud = pred2.is_fraud.astype(int)
submission = pred2[['Id', 'is_fraud']]
submission.to_csv("./data/competition_submission_final.csv", index=False)
```

```
[Parallel(n_jobs=16)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=16)]: Done   3 out of  16 | elapsed:    7.0s remaining:   30.4s
[Parallel(n_jobs=16)]: Done   9 out of  16 | elapsed:    7.1s remaining:    5.6s
[Parallel(n_jobs=16)]: Done  16 out of  16 | elapsed:    7.5s finished
```

[17]:
```
# Create Kaggle Submission 2
pred3 = test_df.drop(['is_fraud'], axis=1)

pred3['is_fraud'] = vot_clf2.predict(pred)
pred3.is_fraud = pred3.is_fraud.astype(int)
submission2 = pred3[['Id', 'is_fraud']]
submission2.to_csv("./data/competition_submission_final2.csv", index=False)
```

```
[Parallel(n_jobs=16)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=16)]: Done   3 out of  16 | elapsed:    8.4s remaining:   36.4s
[Parallel(n_jobs=16)]: Done   9 out of  16 | elapsed:    8.7s remaining:    6.8s
[Parallel(n_jobs=16)]: Done  16 out of  16 | elapsed:    9.2s finished
```

[14]:
```
import pickle

# Save the two ensemble models
with open('competitioin_model_final.obj', 'wb') as f:
    pickle.dump(vot_clf, f)

with open('competitioin_model_final2.obj', 'wb') as f:
    pickle.dump(vot_clf2, f)
```

[15]:
```
# Create Kaggle Submission 3 with only the Bagging Classifier
pred4 = test_df.drop(['is_fraud'], axis=1)

pred4['is_fraud'] = bagging_clf.predict(pred)
pred4.is_fraud = pred4.is_fraud.astype(int)
submission2 = pred4[['Id', 'is_fraud']]
submission2.to_csv("./data/competition_submission_final3.csv", index=False)
```

```
[Parallel(n_jobs=16)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=16)]: Done   3 out of  16 | elapsed:    5.7s remaining:   24.7s
[Parallel(n_jobs=16)]: Done   9 out of  16 | elapsed:    6.0s remaining:    4.6s
[Parallel(n_jobs=16)]: Done  16 out of  16 | elapsed:    6.5s finished
```

0.1.3 Employing Stratified K-Fold Cross-Validation for the Final Model

[ ]:
```
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report, f1_score

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

f1scores = []  # To store the F1 scores from each fold

# Perform cross-validation
for train_index, test_index in skf.split(X, y):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
```

```
    # Fit the model on the training data
    vot_clf.fit(X_train, y_train)
    # Make predictions on the test set
    y_pred = vot_clf.predict(X_test)
    # Calculate and store the F1 score
    score = f1_score(y_test, y_pred)
    f1scores.append(score)

# Compute the average F1 score across all folds
average_f1 = np.mean(f1scores)
print(f"Average F1 Score across all folds: {average_f1}")
```