# task2

December 10, 2024

## 1 Credit Card Fraud Detection

## 2 Task 2: Predictive Modelling

### 2.1 Required libraries

[97]:
```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
import cartopy.crs as ccrs
import cartopy.feature as cfeature
from adjustText import adjust_text
from geopy.distance import geodesic
from sklearn.preprocessing import StandardScaler
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline
from collections import Counter
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import roc_auc_score, roc_curve
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix, classification_report
from xgboost import XGBClassifier
import pickle
import os
from sklearn.impute import KNNImputer
from sklearn.preprocessing import LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
```

### 2.2

### 2.3 Load training and test dataset

[ ]:
```
with open("variables/X_train.pkl", "rb") as f:
    X_train = pickle.load(f)

with open("variables/y_train.pkl", "rb") as f:
    y_train = pickle.load(f)

with open("variables/X_test.pkl", "rb") as f:
    X_test = pickle.load(f)

with open("variables/y_test.pkl", "rb") as f:
```

```
    y_test = pickle.load(f)

with open("variables/kaggle_data.pkl", "rb") as f:
    kaggle_data = pickle.load(f)

with open("variables/index_mapping.pkl", "rb") as f:
    index_mapping = pickle.load(f)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[117], line 19
     16 with open("variables/index_mapping.pkl", "rb") as f:
     17     index_mapping = pickle.load(f)
---> 19 y_train.dtype()

TypeError: 'CategoricalDtype' object is not callable
```

### 2.4 Class Imbalance - SMOTE and model pipeline

To address the significant class imbalance in the dataset, several methods were considered, including **basic SMOTE**, **SMOTE-Tomek**, and **SMOTE combined with RandomUnderSampling**. After evaluating the characteristics of the data, the combination of **SMOTE + RandomUnderSampling** was chosen as the most appropriate technique.

The decision to use **SMOTE + RandomUnderSampling** instead of **SMOTE-Tomek** or **basic SMOTE** was based on the analysis of the dataset's characteristics and the goals of the project.

**Class Imbalance** The dataset exhibits a significant class imbalance, with very few samples belonging to the minority class (`is_fraud = 1`). Addressing this imbalance is critical to ensure that the model does not become biased towards the majority class (`is_fraud = 0`). SMOTE is effective in resolving this issue by generating synthetic samples for the minority class, thereby improving class representation and model fairness.

**Efficiency** **SMOTE + RandomUnderSampling** is computationally simpler and faster compared to **SMOTE-Tomek**, as it does not involve the additional step of identifying and removing Tomek links. This makes it a practical choice for the dataset, given the observed lack of significant class overlap.

[ ]:
```
oversampling_rates = [0.5,0.6,0.7]  # Fraction of majority class
undersampling_rates = [0.8,0.9] # Fraction of total dataset for the majority␣
 ↪class



def model_pipeline(model, submission_file, use_random_search=False,␣
 ↪param_grid=None, n_iter=10):
    results = []
```

```
    best_model = None
    best_auc = 0
    best_config = None

    for oversampling_rate in oversampling_rates:
        for undersampling_rate in undersampling_rates:
            print(f"\nTesting Oversampling={oversampling_rate},␣
↪Undersampling={undersampling_rate}")

            # Define SMOTE and undersampler
            smote = SMOTE(sampling_strategy=oversampling_rate, random_state=42)
            undersampler =␣
↪RandomUnderSampler(sampling_strategy=undersampling_rate, random_state=42)

            # Create pipeline
            pipeline = Pipeline(steps=[
                ('smote', smote),
                ('undersampler', undersampler),
                ('model', model)
            ])

            if use_random_search:
                search = RandomizedSearchCV(
                    estimator=pipeline,
                    param_distributions=param_grid,  # Prefix for model params
                    n_iter=n_iter,
                    scoring='roc_auc',
                    cv=5,
                    n_jobs=-1,
                    random_state=42
                )
                # Train with hyperparameter tuning
                search.fit(X_train, y_train)
                best_pipeline = search.best_estimator_
                best_params = search.best_params_
                print(f"Best Parameters for this iteration: {best_params}")
            else:
                # Train pipeline without hyperparameter search
                pipeline.fit(X_train, y_train)
                best_pipeline = pipeline
                best_params = "Default parameters"



            # Predict on the test set
            y_pred = best_pipeline.predict(X_test)
```

```
            y_pred_proba = best_pipeline.predict_proba(X_test)[:, 1]  # Get␣
↪probabilities for AUC


            # Evaluate model
            print("Classification Report:")
            report = classification_report(y_test, y_pred, output_dict=True,␣
↪zero_division=0)


            # Confusion matrix
            print("Confusion Matrix:")
            print(confusion_matrix(y_test, y_pred))

            # Calculate AUC
            auc_score = roc_auc_score(y_test, y_pred_proba)


            # Store results
            results.append({
                'oversampling_rate': oversampling_rate,
                'undersampling_rate': undersampling_rate,
                'precision': report['1']['precision'],
                'recall': report['1']['recall'],
                'f1_score': report['1']['f1-score'],
                'auc': auc_score
            })

            # Check if current model is the best
            if auc_score > best_auc:
                best_auc = auc_score
                best_model = best_pipeline
                best_config = {
                    'oversampling_rate': oversampling_rate,
                    'undersampling_rate': undersampling_rate
                }

    # Print best configuration
    print("\nBest Configuration:")
    print(f"Oversampling Rate: {best_config['oversampling_rate']}")
    print(f"Undersampling Rate: {best_config['undersampling_rate']}")
    print(f"Best AUC: {best_auc:.4f}")

    # Predict probabilities for Kaggle submission
    test_probs = best_model.predict_proba(kaggle_data)[:, 1]  # Probabilities␣
↪for class 1 (fraud)
```

```
    # Create submission DataFrame
    submission = pd.DataFrame({
        'index': index_mapping,
        'is_fraud': test_probs
    })

    # Save to CSV
    submission_file_name = f"{submission_file}.csv"
    submission.to_csv(f"submission/{submission_file_name}", index=False)
    print(f"Submission file created: '{submission_file_name}'")

    return results
```

### 2.5 Random Forest Classifier

[100]:
```
# Train a Random Forest Classifier
rf = RandomForestClassifier(random_state=42)

model_pipeline(rf,"submission_random_forest")
```

```
Testing Oversampling=0.5, Undersampling=0.8
Classification Report:
Confusion Matrix:
[[5882    4]
 [ 114    0]]

Testing Oversampling=0.5, Undersampling=1.0
Classification Report:
Confusion Matrix:
[[5880    6]
 [ 114    0]]

Testing Oversampling=0.7, Undersampling=0.8
Classification Report:
Confusion Matrix:
[[5885    1]
 [ 114    0]]

Testing Oversampling=0.7, Undersampling=1.0
Classification Report:
Confusion Matrix:
[[5882    4]
 [ 114    0]]

Best Configuration:
Oversampling Rate: 0.5
Undersampling Rate: 1.0
```

```
Best AUC: 0.4655
Submission file created: 'submission_random_forest.csv'
```

[100]:
```
[{'oversampling_rate': 0.5,
  'undersampling_rate': 0.8,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.43127462727494914)},
 {'oversampling_rate': 0.5,
  'undersampling_rate': 1.0,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.4655434244803309)},
 {'oversampling_rate': 0.7,
  'undersampling_rate': 0.8,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.43850781813521233)},
 {'oversampling_rate': 0.7,
  'undersampling_rate': 1.0,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.43363005287598877)}]
```

### 2.6 Random Search - Random Forest Classifier

[105]:
```
rf = RandomForestClassifier(random_state=42)

param_distributions = {
        'model__n_estimators': [100, 200, 500,1000],
        'model__max_depth': [None, 10, 20, 30],
        'model__min_samples_split': [2, 5, 10],
        'model__min_samples_leaf': [1, 2, 4, 5],
    }

model_pipeline(rf,"submission_random_search_random_forest",True,param_distributions)
```

```
Testing Oversampling=0.5, Undersampling=0.8
Best Parameters for this iteration: {'model__n_estimators': 100,
'model__min_samples_split': 5, 'model__min_samples_leaf': 4, 'model__max_depth':
10}
Classification Report:
Confusion Matrix:
```

```
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.5, Undersampling=1.0
Best Parameters for this iteration: {'model__n_estimators': 100,
'model__min_samples_split': 5, 'model__min_samples_leaf': 4, 'model__max_depth':
10}
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.7, Undersampling=0.8
Best Parameters for this iteration: {'model__n_estimators': 100,
'model__min_samples_split': 5, 'model__min_samples_leaf': 4, 'model__max_depth':
10}
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.7, Undersampling=1.0
Best Parameters for this iteration: {'model__n_estimators': 100,
'model__min_samples_split': 5, 'model__min_samples_leaf': 4, 'model__max_depth':
10}
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Best Configuration:
Oversampling Rate: 0.5
Undersampling Rate: 1.0
Best AUC: 0.4935
Submission file created: 'submission_random_search_random_forest.csv'
```

[105]:
```
[{'oversampling_rate': 0.5,
  'undersampling_rate': 0.8,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.47421624908346294)},
 {'oversampling_rate': 0.5,
  'undersampling_rate': 1.0,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
```

```
  'auc': np.float64(0.4935238836132124)},
 {'oversampling_rate': 0.7,
  'undersampling_rate': 0.8,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.47737271312838675)},
 {'oversampling_rate': 0.7,
  'undersampling_rate': 1.0,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.4841342227468093)}]
```

**Score on Kaggle:** 0.52296

### 2.7 XGBOOST

[106]:
```
xgb = XGBClassifier(n_estimators=500, max_depth=5, learning_rate=0.1,␣
 ↪random_state=42)


model_pipeline(xgb,"submission_xgboost")
```

```
Testing Oversampling=0.5, Undersampling=0.8
Classification Report:
Confusion Matrix:
[[5841   45]
 [ 114    0]]

Testing Oversampling=0.5, Undersampling=1.0
Classification Report:
Confusion Matrix:
[[5840   46]
 [ 114    0]]

Testing Oversampling=0.7, Undersampling=0.8
Classification Report:
Confusion Matrix:
[[5864   22]
 [ 114    0]]

Testing Oversampling=0.7, Undersampling=1.0
Classification Report:
Confusion Matrix:
[[5851   35]
 [ 114    0]]
```

```
Best Configuration:
Oversampling Rate: 0.5
Undersampling Rate: 0.8
Best AUC: 0.4529
Submission file created: 'submission_xgboost.csv'
```

[106]:
```
[{'oversampling_rate': 0.5,
  'undersampling_rate': 0.8,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.452863172201656)},
 {'oversampling_rate': 0.5,
  'undersampling_rate': 1.0,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.43907264338215557)},
 {'oversampling_rate': 0.7,
  'undersampling_rate': 0.8,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.4474265429118157)},
 {'oversampling_rate': 0.7,
  'undersampling_rate': 1.0,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.4478035898444719)}]
```

**Score on Kaggle:**

### 2.8 Random Search - XGBOOST

[107]:
```
xgb = XGBClassifier(n_estimators=500, max_depth=5, learning_rate=0.1,␣
 ↪random_state=42)

param_distributions = {
        'model__n_estimators': [100, 200, 300, 500, 1000],
        'model__learning_rate': [0.01, 0.05, 0.1, 0.2, 0.3],
        'model__max_depth': [3, 5, 7, 10],
        'model__subsample': [0.6, 0.8, 1.0],
        'model__colsample_bytree': [0.6, 0.8, 1.0],
        'model__reg_alpha': [0, 0.1, 1, 10],
        'model__reg_lambda': [1, 10, 50],
        'model__min_child_weight': [1, 3, 5, 7]
```

```
    }


model_pipeline(xgb,"submission_random_search_xgboost",True, param_distributions)
```

```
Testing Oversampling=0.5, Undersampling=0.8
Best Parameters for this iteration: {'model__subsample': 1.0,
'model__reg_lambda': 1, 'model__reg_alpha': 0, 'model__n_estimators': 100,
'model__min_child_weight': 5, 'model__max_depth': 7, 'model__learning_rate':
0.1, 'model__colsample_bytree': 0.8}
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.5, Undersampling=1.0
Best Parameters for this iteration: {'model__subsample': 1.0,
'model__reg_lambda': 1, 'model__reg_alpha': 0, 'model__n_estimators': 100,
'model__min_child_weight': 5, 'model__max_depth': 7, 'model__learning_rate':
0.1, 'model__colsample_bytree': 0.8}
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.7, Undersampling=0.8
Best Parameters for this iteration: {'model__subsample': 1.0,
'model__reg_lambda': 1, 'model__reg_alpha': 0, 'model__n_estimators': 100,
'model__min_child_weight': 5, 'model__max_depth': 7, 'model__learning_rate':
0.1, 'model__colsample_bytree': 0.8}
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.7, Undersampling=1.0
Best Parameters for this iteration: {'model__subsample': 1.0,
'model__reg_lambda': 1, 'model__reg_alpha': 0, 'model__n_estimators': 100,
'model__min_child_weight': 5, 'model__max_depth': 7, 'model__learning_rate':
0.1, 'model__colsample_bytree': 0.8}
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Best Configuration:
Oversampling Rate: 0.7
```

```
Undersampling Rate: 0.8
Best AUC: 0.4484
Submission file created: 'submission_random_search_xgboost.csv'
```

[107]:
```
[{'oversampling_rate': 0.5,
  'undersampling_rate': 0.8,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.44771715220773656)},
 {'oversampling_rate': 0.5,
  'undersampling_rate': 1.0,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.4407209196964549)},
 {'oversampling_rate': 0.7,
  'undersampling_rate': 0.8,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.44839896632508897)},
 {'oversampling_rate': 0.7,
  'undersampling_rate': 1.0,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.4454444384832281)}]
```

**Score on Kaggle:** 0.52380

### 2.9 Decision Tree

[115]:
```python
dt = DecisionTreeClassifier(random_state = 42)

parameter_grid = {
    'model__max_depth': [5, 10, 20, 30, 40, 50],
    'model__min_samples_split': [2, 5, 10, 20],
    'model__min_samples_leaf': [1, 2, 4, 5],
    'model__max_leaf_nodes': [None, 10, 20, 50, 100],
    'model__max_features': [1, 2, 3, 4, 5, 6, 7, 8,]
}

model_pipeline(dt,"submission_decision_tree",True, parameter_grid)
```

```
Testing Oversampling=0.5, Undersampling=0.8
Best Parameters for this iteration: {'model__min_samples_split': 20,
'model__min_samples_leaf': 1, 'model__max_leaf_nodes': 100,
```

```
'model__max_features': 4, 'model__max_depth': 50}
Classification Report:
Confusion Matrix:
[[5118  768]
 [ 102   12]]

Testing Oversampling=0.5, Undersampling=1.0
Best Parameters for this iteration: {'model__min_samples_split': 20,
'model__min_samples_leaf': 1, 'model__max_leaf_nodes': 100,
'model__max_features': 4, 'model__max_depth': 50}
Classification Report:
Confusion Matrix:
[[4791 1095]
 [ 100   14]]

Testing Oversampling=0.7, Undersampling=0.8
Best Parameters for this iteration: {'model__min_samples_split': 10,
'model__min_samples_leaf': 4, 'model__max_leaf_nodes': None,
'model__max_features': 7, 'model__max_depth': 10}
Classification Report:
Confusion Matrix:
[[5850   36]
 [ 112    2]]

Testing Oversampling=0.7, Undersampling=1.0
Best Parameters for this iteration: {'model__min_samples_split': 10,
'model__min_samples_leaf': 5, 'model__max_leaf_nodes': None,
'model__max_features': 1, 'model__max_depth': 20}
Classification Report:
Confusion Matrix:
[[5213  673]
 [  99   15]]

Best Configuration:
Oversampling Rate: 0.7
Undersampling Rate: 1.0
Best AUC: 0.5068
Submission file created: 'submission_decision_tree.csv'
```

[115]:
```
[{'oversampling_rate': 0.5,
  'undersampling_rate': 0.8,
  'precision': 0.015384615384615385,
  'recall': 0.10526315789473684,
  'f1_score': 0.026845637583892617,
  'auc': np.float64(0.4946654565397524)},
 {'oversampling_rate': 0.5,
  'undersampling_rate': 1.0,
```

```
  'precision': 0.012623985572587917,
  'recall': 0.12280701754385964,
  'f1_score': 0.022894521668029435,
  'auc': np.float64(0.4907832144070677)},
 {'oversampling_rate': 0.7,
  'undersampling_rate': 0.8,
  'precision': 0.05263157894736842,
  'recall': 0.017543859649122806,
  'f1_score': 0.02631578947368421,
  'auc': np.float64(0.4855552276886576)},
 {'oversampling_rate': 0.7,
  'undersampling_rate': 1.0,
  'precision': 0.02180232558139535,
  'recall': 0.13157894736842105,
  'f1_score': 0.03740648379052369,
  'auc': np.float64(0.5068322990623008)}]
```

### 2.10 Neural Networks - Multi-Layer Perceptron (MLP)

[109]:
```
mlp = MLPClassifier(hidden_layer_sizes=(100,), max_iter=300, random_state=42)



model_pipeline(mlp,"submission_mlp")
```

```
Testing Oversampling=0.5, Undersampling=0.8
Classification Report:
Confusion Matrix:
[[   0 5886]
 [   0  114]]

Testing Oversampling=0.5, Undersampling=1.0
Classification Report:
Confusion Matrix:
[[   0 5886]
 [   0  114]]

Testing Oversampling=0.7, Undersampling=0.8
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.7, Undersampling=1.0
Classification Report:
Confusion Matrix:
[[5886    0]
```

```
 [ 114    0]]

Best Configuration:
Oversampling Rate: 0.5
Undersampling Rate: 0.8
Best AUC: 0.5000
Submission file created: 'submission_mlp.csv'
```

[109]:
```
[{'oversampling_rate': 0.5,
  'undersampling_rate': 0.8,
  'precision': 0.019,
  'recall': 1.0,
  'f1_score': 0.03729146221786065,
  'auc': np.float64(0.5)},
 {'oversampling_rate': 0.5,
  'undersampling_rate': 1.0,
  'precision': 0.019,
  'recall': 1.0,
  'f1_score': 0.03729146221786065,
  'auc': np.float64(0.5)},
 {'oversampling_rate': 0.7,
  'undersampling_rate': 0.8,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.5)},
 {'oversampling_rate': 0.7,
  'undersampling_rate': 1.0,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.5)}]
```

### 2.11 Random Search - Neural Networks (MLP)

[110]:
```
parameter_grid = {
    'model__hidden_layer_sizes': [(50,), (100,), (100, 50), (150, 100)],
    'model__activation': ['relu', 'tanh'],
    'model__solver': ['adam', 'sgd'],
    'model__alpha': [0.0001, 0.001, 0.01],
    'model__learning_rate': ['constant', 'adaptive']
}

model_pipeline(mlp,"submission_random_search_mlp",True,parameter_grid)
```

```
Testing Oversampling=0.5, Undersampling=0.8
Best Parameters for this iteration: {'model__solver': 'adam',
```

```
'model__learning_rate': 'adaptive', 'model__hidden_layer_sizes': (100, 50),
'model__alpha': 0.0001, 'model__activation': 'relu'}
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.5, Undersampling=1.0
Best Parameters for this iteration: {'model__solver': 'adam',
'model__learning_rate': 'adaptive', 'model__hidden_layer_sizes': (100, 50),
'model__alpha': 0.0001, 'model__activation': 'relu'}
Classification Report:
Confusion Matrix:
[[   0 5886]
 [   0  114]]

Testing Oversampling=0.7, Undersampling=0.8
Best Parameters for this iteration: {'model__solver': 'adam',
'model__learning_rate': 'constant', 'model__hidden_layer_sizes': (50,),
'model__alpha': 0.0001, 'model__activation': 'relu'}
Classification Report:
Confusion Matrix:
[[ 365 5521]
 [   7  107]]

Testing Oversampling=0.7, Undersampling=1.0
Best Parameters for this iteration: {'model__solver': 'adam',
'model__learning_rate': 'constant', 'model__hidden_layer_sizes': (50,),
'model__alpha': 0.0001, 'model__activation': 'relu'}
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Best Configuration:
Oversampling Rate: 0.7
Undersampling Rate: 0.8
Best AUC: 0.5003
Submission file created: 'submission_random_search_mlp.csv'
```

[110]:
```
[{'oversampling_rate': 0.5,
  'undersampling_rate': 0.8,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.5)},
 {'oversampling_rate': 0.5,
```

```
 'undersampling_rate': 1.0,
 'precision': 0.019,
 'recall': 1.0,
 'f1_score': 0.03729146221786065,
 'auc': np.float64(0.5)},
{'oversampling_rate': 0.7,
 'undersampling_rate': 0.8,
 'precision': 0.019012082444918265,
 'recall': 0.9385964912280702,
 'f1_score': 0.03726924416579589,
 'auc': np.float64(0.5003040220326556)},
{'oversampling_rate': 0.7,
 'undersampling_rate': 1.0,
 'precision': 0.0,
 'recall': 0.0,
 'f1_score': 0.0,
 'auc': np.float64(0.5)}]
```

### 2.12 Support Vector Machine (SVM)

[111]:
```
svm = SVC(kernel='rbf', probability=True, random_state=42)


model_pipeline(svm,"submission_svm")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[111], line 1
----> 1 aa
      2 svm = SVC(kernel='rbf', probability=True, random_state=42)
      3 #svm.fit(X_train, y_train)

NameError: name 'aa' is not defined
```