task2

December 10, 2024

1 Credit Card Fraud Detection

2 Task 2: Predictive Modelling

2.1 Required libraries

[1]:

```
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 Load training and test dataset

[2]:

```
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)
```

2.3 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.

[3]:

```
oversampling_rates = [0.5,0.7]   # Fraction of majority class
undersampling_rates = [0.8,1.0] # 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
```

2.4 Random Forest Classifier

[4]:

```python
# 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:
[[5883    3]
 [ 114    0]]

Testing Oversampling=0.5, Undersampling=1.0
Classification Report:
Confusion Matrix:
[[5884    2]
 [ 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:
[[5886    0]
 [ 114    0]]

Best Configuration:
Oversampling Rate: 0.5
Undersampling Rate: 1.0
Best AUC: 0.4390
Submission file created: 'submission_random_forest.csv'
```

[4]:

```
[{'oversampling_rate': 0.5,
  'undersampling_rate': 0.8,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.42704887005144453)},
 {'oversampling_rate': 0.5,
  'undersampling_rate': 1.0,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.4389839702892978)},
 {'oversampling_rate': 0.7,
  'undersampling_rate': 0.8,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.4185668043707639)},
 {'oversampling_rate': 0.7,
  'undersampling_rate': 1.0,
  'precision': 0.0,
  'recall': 0.0,
  'f1_score': 0.0,
  'auc': np.float64(0.41083212618702725)}]
```

2.5 Random Search - Random Forest Classifier

[10]:

```
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:
[[5885    1]
 [ 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]
 [ 113    1]]

Best Configuration:
Oversampling Rate: 0.5
Undersampling Rate: 0.8
Best AUC: 0.4489
Submission file created: 'submission_random_search_random_forest.csv'
```

```
[10]: [{'oversampling_rate': 0.5,
        'undersampling_rate': 0.8,
        'precision': 0.0,
        'recall': 0.0,
        'f1_score': 0.0,
        'auc': np.float64(0.44885947028631723)},
       {'oversampling_rate': 0.5,
        'undersampling_rate': 1.0,
        'precision': 0.0,
        'recall': 0.0,
        'f1_score': 0.0,
        'auc': np.float64(0.43757488778010 26)},
       {'oversampling_rate': 0.7,
        'undersampling_rate': 0.8,
        'precision': 0.0,
        'recall': 0.0,
        'f1_score': 0.0,
        'auc': np.float64(0.43664419288111544)},
       {'oversampling_rate': 0.7,
        'undersampling_rate': 1.0,
        'precision': 1.0,
        'recall': 0.008771929824561403,
        'f1_score': 0.017391304347826087,
        'auc': np.float64(0.43613674434131544)}]
```

2.6 XGBOOST

[ ]:

```python
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:
[[4226 1660]
 [  93   21]]

Testing Oversampling=0.5, Undersampling=1.0
Classification Report:
Confusion Matrix:
[[4215 1671]
 [  91   23]]

Testing Oversampling=0.7, Undersampling=0.8
Classification Report:
Confusion Matrix:
[[4350 1536]
 [  91   23]]

Testing Oversampling=0.7, Undersampling=1.0
Classification Report:
Confusion Matrix:
[[4307 1579]
 [  92   22]]

Best Configuration:
Oversampling Rate: 0.7
Undersampling Rate: 1.0
Best AUC: 0.4238
Submission file created: 'submission_xgboost.csv'
```

[ ]:

```
[{'oversampling_rate': 0.5,
  'undersampling_rate': 0.8,
  'precision': 0.012492563950029744,
  'recall': 0.18421052631578946,
  'f1_score': 0.0233983286908078,
  'auc': np.float64(0.408907398465583)},
 {'oversampling_rate': 0.5,
  'undersampling_rate': 1.0,
  'precision': 0.013577331759149940,
  'recall': 0.20175438596491227,
  'f1_score': 0.025442477876106196,
  'auc': np.float64(0.41800719518810625)},
 {'oversampling_rate': 0.7,
  'undersampling_rate': 0.8,
  'precision': 0.014753046824887749,
  'recall': 0.20175438596491227,
  'f1_score': 0.02749551703526599,
  'auc': np.float64(0.4136182496676622)},
 {'oversampling_rate': 0.7,
  'undersampling_rate': 1.0,
  'precision': 0.013741411617738912,
  'recall': 0.19298245614035087,
  'f1_score': 0.02565597667638484,
  'auc': np.float64(0.4237552980310102)}]
```

2.7 Random Search - XGBOOST

[ ]:

```
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
```

```
/home/ricardo/Desktop/uni/mestrado/fraude/project/venv/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(
```

```
Best Parameters for this iteration: {'model__subsample': 0.6,
'model__reg_lambda': 10, 'model__reg_alpha': 0, 'model__n_estimators': 100,
'model__min_child_weight': 3, 'model__max_depth': 7, 'model__learning_rate':
0.01, 'model__colsample_bytree': 0.8}
Classification Report:
```

```
Confusion Matrix:
[[5150  736]
 [ 105    9]]

Testing Oversampling=0.5, Undersampling=1.0
Best Parameters for this iteration: {'model__subsample': 0.6,
'model__reg_lambda': 10, 'model__reg_alpha': 0, 'model__n_estimators': 100,
'model__min_child_weight': 3, 'model__max_depth': 7, 'model__learning_rate':
0.01, 'model__colsample_bytree': 0.8}
Classification Report:
Confusion Matrix:
[[5133  753]
 [ 105    9]]

Testing Oversampling=0.7, Undersampling=0.8
Best Parameters for this iteration: {'model__subsample': 0.6,
'model__reg_lambda': 10, 'model__reg_alpha': 0, 'model__n_estimators': 100,
'model__min_child_weight': 3, 'model__max_depth': 7, 'model__learning_rate':
0.01, 'model__colsample_bytree': 0.8}
Classification Report:
Confusion Matrix:
[[5201  685]
 [ 105    9]]

Testing Oversampling=0.7, Undersampling=1.0
Best Parameters for this iteration: {'model__subsample': 0.6,
'model__reg_lambda': 10, 'model__reg_alpha': 0, 'model__n_estimators': 100,
'model__min_child_weight': 3, 'model__max_depth': 7, 'model__learning_rate':
0.01, 'model__colsample_bytree': 0.8}
Classification Report:
Confusion Matrix:
[[4990  896]
 [ 105    9]]

Best Configuration:
Oversampling Rate: 0.5
Undersampling Rate: 1.0
Best AUC: 0.4266
Submission file created: 'submission_random_search_xgboost.csv'
```

```
[ ]: [{'oversampling_rate': 0.5,
       'undersampling_rate': 0.8,
       'precision': 0.012080536912751677,
       'recall': 0.07894736842105263,
       'f1_score': 0.020954598370197905,
       'auc': np.float64(0.4217083653748711)},
      {'oversampling_rate': 0.5,
       'undersampling_rate': 1.0,
       'precision': 0.011811023622047244,
       'recall': 0.07894736842105263,
       'f1_score': 0.02054794520547945,
       'auc': np.float64(0.42658613063409456)},
      {'oversampling_rate': 0.7,
       'undersampling_rate': 0.8,
       'precision': 0.012968299711815562,
       'recall': 0.07894736842105263,
       'f1_score': 0.022277227722772276,
       'auc': np.float64(0.4223812376677338)},
      {'oversampling_rate': 0.7,
       'undersampling_rate': 1.0,
       'precision': 0.009944751381215469,
       'recall': 0.07894736842105263,
       'f1_score': 0.017664376840039256,
       'auc': np.float64(0.4227784037054921)}]
```

2.8 Decision Tree

[ ]:

```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': 10,
'model__min_samples_leaf': 4, 'model__max_leaf_nodes': None,
'model__max_features': 7, 'model__max_depth': 10}
Classification Report:
Confusion Matrix:
[[5282  604]
 [  97   17]]

Testing Oversampling=0.5, Undersampling=1.0
Best Parameters for this iteration: {'model__min_samples_split': 20,
'model__min_samples_leaf': 2, 'model__max_leaf_nodes': 50,
'model__max_features': 6, 'model__max_depth': 10}
Classification Report:
Confusion Matrix:
```

```
[[3691 2195]
 [  60   54]]

Testing Oversampling=0.7, Undersampling=0.8
Best Parameters for this iteration: {'model__min_samples_split': 20,
'model__min_samples_leaf': 2, 'model__max_leaf_nodes': 50,
'model__max_features': 6, 'model__max_depth': 10}
Classification Report:
Confusion Matrix:
[[4537 1349]
 [  85   29]]

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:
[[ 803 5083]
 [  19   95]]

Best Configuration:
Oversampling Rate: 0.5
Undersampling Rate: 1.0
Best AUC: 0.5333
Submission file created: 'submission_decision_tree.csv'
```

```
[ ]: [{'oversampling_rate': 0.5,
       'undersampling_rate': 0.8,
       'precision': 0.027375201288244767,
       'recall': 0.14912280701754385,
       'f1_score': 0.04625850340136054,
       'auc': np.float64(0.5215647000614005)},
      {'oversampling_rate': 0.5,
       'undersampling_rate': 1.0,
       'precision': 0.02401067140951534,
       'recall': 0.47368421052631576,
       'f1_score': 0.045704612780363946,
       'auc': np.float64(0.5332613516461898)},
      {'oversampling_rate': 0.7,
       'undersampling_rate': 0.8,
       'precision': 0.02104499274310595,
       'recall': 0.2543859649122807,
       'f1_score': 0.0388739946380697,
       'auc': np.float64(0.4858979976274359)},
      {'oversampling_rate': 0.7,
       'undersampling_rate': 1.0,
       'precision': 0.018346852066434917,
       'recall': 0.8333333333333334,
       'f1_score': 0.035903250188964474,
       'auc': np.float64(0.4907176410274753)}]
```

2.9 Neural Networks - Multi-Layer Perceptron (MLP)

[9]:
```
mlp = MLPClassifier(max_iter=300, random_state=42)



model_pipeline(mlp,"submission_mlp")
```

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

Testing Oversampling=0.5, Undersampling=1.0

/home/ricardo/Desktop/uni/mestrado/fraude/project/venv/lib/python3.12/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:690:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and
the optimization hasn't converged yet.
  warnings.warn(

Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.7, Undersampling=0.8

/home/ricardo/Desktop/uni/mestrado/fraude/project/venv/lib/python3.12/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:690:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and
the optimization hasn't converged yet.
  warnings.warn(

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'
```

```
/home/ricardo/Desktop/uni/mestrado/fraude/project/venv/lib/python3.12/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:690:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

```
[9]: [{'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': 0.0,
       'recall': 0.0,
       'f1_score': 0.0,
       '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)}]
```

2.10 Random Search - Neural Networks (MLP)

```
[8]: mlp = MLPClassifier( max_iter=300, random_state=42,early_stopping=True)



     parameter_grid = {
         'model__hidden_layer_sizes': [(50,), (100,), (100, 50)],
         'model__activation': ['relu', 'tanh'],
         'model__solver': ['adam', 'sgd'],
         'model__alpha': [0.0001, 0.001],
         'model__learning_rate': ['constant', 'adaptive'],
         'model__max_iter': [300, 500, 1000]
     }

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

```
Testing Oversampling=0.5, Undersampling=0.8
Best Parameters for this iteration: {'model__solver': 'sgd', 'model__max_iter':
300, 'model__learning_rate': 'constant', 'model__hidden_layer_sizes': (100, 50),
'model__alpha': 0.0001, 'model__activation': 'tanh'}
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.5, Undersampling=1.0
Best Parameters for this iteration: {'model__solver': 'adam', 'model__max_iter':
1000, 'model__learning_rate': 'adaptive', 'model__hidden_layer_sizes': (50,),
'model__alpha': 0.0001, 'model__activation': 'tanh'}
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.7, Undersampling=0.8
Best Parameters for this iteration: {'model__solver': 'sgd', 'model__max_iter':
300, 'model__learning_rate': 'constant', 'model__hidden_layer_sizes': (100, 50),
'model__alpha': 0.0001, 'model__activation': 'tanh'}
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.7, Undersampling=1.0
Best Parameters for this iteration: {'model__solver': 'sgd', 'model__max_iter':
300, 'model__learning_rate': 'constant', 'model__hidden_layer_sizes': (100, 50),
'model__alpha': 0.0001, 'model__activation': 'tanh'}
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_random_search_mlp.csv'
```

```
[8]: [{'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': 0.0,
       'recall': 0.0,
       'f1_score': 0.0,
       '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 Support Vector Machine (SVM)

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


     model_pipeline(svm,"submission_svm")
```

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

Testing Oversampling=0.7, Undersampling=1.0
Classification Report:
Confusion Matrix:
[[   0 5886]
 [   0  114]]

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

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

3 Conclusion

After analyzing and experimenting with various models, the `Decision Tree` appeared to perform the best in the current setup, even though the overall score was not particularly high.

I made an effort to follow the appropriate steps required for success if the dataset were real, and the process provided valuable insights into model development. While the `Decision Tree` outperformed the other tested algorithms, its performance was still relatively modest. `Decision Trees` are known for capturing complex relationships within data, which made them a suitable choice for this experiment. I aimed to adhere to best practices, including data preprocessing, feature selection, and hyperparameter tuning, and applied methods that are scalable to real-world datasets.

Although the final results were not exceptional, this exercise was a meaningful attempt to systematically develop and evaluate models. With a real-world dataset, this approach could potentially lead to more promising outcomes.