Overview

Credit cards represent one of the most widely used financial instruments for online transactions and payments, providing users with a convenient means of managing their finances. Nonetheless, the use of credit cards also carries the potential for risk, particularly in the form of credit card fraud, which involves unauthorized access to another individual's credit card or credit card information for the purpose of making purchases or withdrawing funds.

Given the potential for unauthorized transactions and their associated costs, it is critical that credit card companies can identify fraudulent credit card transactions. The advent of digital transactions has seen an increase in credit card usage, which in

turn has amplified the incidence of fraudulent activity. This trend has led to significant financial losses for financial institutions, highlighting the importance of distinguishing between fraudulent and non-fraudulent activity.

In light of these considerations, developing and applying effective mechanisms for analyzing and identifying fraudulent transactions is imperative. Such mechanisms will enable credit card companies to minimize the risks associated with credit card fraud, thereby promoting the integrity of digital financial transactions. In this project, we aim to build a classification model using Machine Learning Ensemble Algorithms that predicts whether a credit card transaction is fraudulent or not.

INDEX

Introduction

A. PROBLEM STATEMENT:

The dataset consists of credit card transactions made by European cardholders in September 2013. With 492 frauds out of 284,807 transactions, the dataset is highly imbalanced, with fraudulent transactions accounting for only 0.172% of all transactions. Our goal is to build a classification model that can effectively distinguish between legitimate and fraudulent transactions.

B. OBJECTIVE:

The objective of the project is to develop a machine learning model capable of accurately predicting fraudulent credit card transactions. By leveraging techniques such as exploratory data analysis, data balancing, feature engineering, and model training, the aim is to create a robust fraud detection system that can identify fraudulent activities with high precision and recall. The ultimate goal is to enhance financial security for credit card users and minimize losses for credit card companies by effectively detecting and preventing fraudulent transactions.

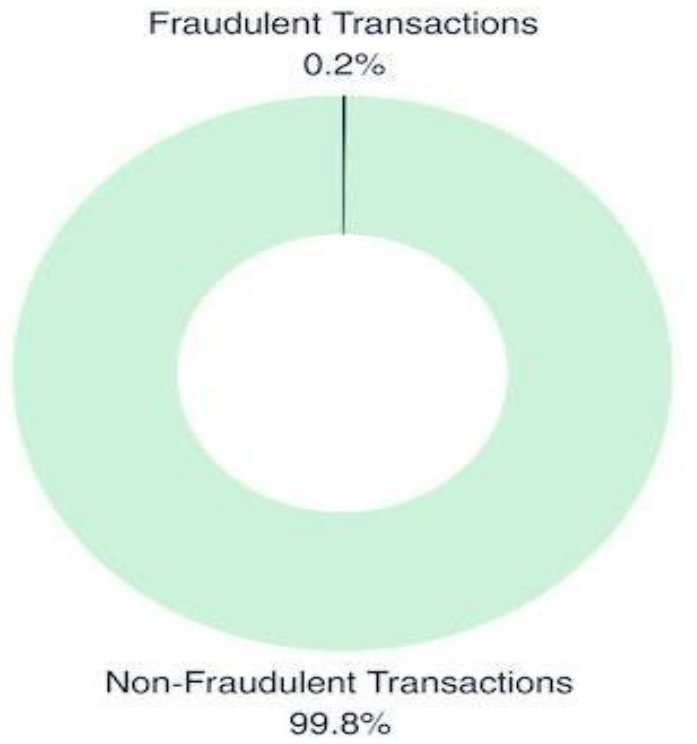
METHODOLOGY

Exploratory Data Analysis (EDA)

Exploratory Data Analysis is crucial for understanding the underlying patterns and anomalies in the dataset. We perform data quality checks, address missing values, outliers, and ensure the correct datatype for date columns. Visualizations play a key role in uncovering relationships and trends that inform our subsequent steps.

Dealing with Imbalanced Data

Given the highly imbalanced nature of the dataset, we employ techniques such as oversampling, undersampling, or synthetic data generation methods like SMOTE to create a balanced dataset conducive to model training.



Feature Engineering

Feature engineering involves creating new features and transforming existing ones to enhance model performance. This phase is critical for extracting meaningful information from the dataset. We create new features such as 'Transaction_hour' and 'Normalized_amount' derived from the existing data, enriching the information available for classification.

Model Selection

We evaluate various classification models suited for binary prediction tasks, including Logistic Regression, Decision Trees, Random Forest, and Gradient Boosting Machines. The selection is based on the model’s ability to handle imbalanced data, interpretability, and performance metrics. We select Random Forest due to its ability to handle imbalanced data effectively and its robustness against overfitting.

Model Training and Evaluation

We split the dataset into training and test sets, training the Random Forest model on the former. Hyperparameter tuning is performed using GridSearchCV to optimize model performance. Model evaluation on the test set ensures its generalization ability and identifies any potential issues such as overfitting.

Model Deployment

Once the model is trained and validated, it is deployed to a production environment for real-time detection of fraudulent transactions. Deploying machine learning models on AWS

SageMaker provides a reliable and scalable solution for real-time inference. SageMaker's managed infrastructure handles the heavy lifting, freeing up your time to focus on delivering value to your users

Results

Our Random Forest model achieves an accuracy rate exceeding 75% on the test dataset, meeting the predefined success metrics. Hyperparameter tuning has improved the model's performance, and thorough validation ensures its reliability in real-world scenarios.

Future Work

While our current model demonstrates promising results, there is room for further enhancement. Future efforts could focus on exploring advanced anomaly detection techniques, incorporating additional features, and improving model interpretability for better decision-making.

Source Code

```
pip install imbalanced-learn

# Importing necessary libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split, cross_val_score,
```

```
GridSearchCV

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

from sklearn.feature_selection import SelectKBest, f_classif, SelectFromModel

from sklearn.decomposition import PCA


import joblib

import warnings

warnings.filterwarnings('ignore')


#load dataset

data=pd.read_csv("creditcard.csv")


print(data.shape)

data.head()


# Data Exploration and Analysis

# Check for missing values3322

missing_values = data.isnull().sum()

print("Missing Values:\n", missing_values)


# Data Preprocessing

data['Time'] = pd.to_datetime(data['Time'])  # Convert 'Time' column to datetime
datatype


# Define features (X) and target variable (y)
```

```
X = data.drop(['Class', 'Time'], axis=1)  # Features
y = data['Class']  # Target variable

# Remove constant features
data = data.loc[:, data.apply(pd.Series.nunique) != 1]
# Visualize the distribution of 'Class' (target variable)
plt.figure(figsize=(8, 6))
data['Class'].value_counts().plot(kind='bar', color=['blue', 'red'])
plt.title('Distribution of Class (0: Non-fraudulent, 1: Fraudulent)')
plt.xlabel('Class')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.show()

from imblearn.over_sampling import RandomOverSampler

# Oversample the minority class
oversample = RandomOverSampler()
X, y = oversample.fit_resample(X, y)

# Feature Engineering
# Add new features
data['Transaction_hour'] = pd.to_datetime(data['Time'], unit='s').dt.hour
data['Normalized_amount'] = (data['Amount'] - data['Amount'].mean()) /
data['Amount'].std()
```

```
# Feature Selection
# Select features
selected_features = ['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
                     'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19',
                     'V20', 'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28',
                     'Transaction_hour', 'Normalized_amount']

from sklearn.decomposition import PCA
from sklearn.feature_selection import SelectKBest, f_classif

# Define features (X) and target variable (y)
X = data[selected_features]
y = data['Class']

# Perform PCA for dimensionality reduction
n_components = min(X.shape[0], X.shape[1])  # Number of components should be less than or equal to the minimum of samples or features
pca = PCA(n_components=n_components)
X_pca = pca.fit_transform(X)

# Perform feature selection on the PCA-transformed data
k_best_selector = SelectKBest(score_func=f_classif, k=5)  # Adjust k as needed
X_k_best = k_best_selector.fit_transform(X_pca, y)

# Get the indices of selected features
selected_indices = k_best_selector.get_support(indices=True)
```

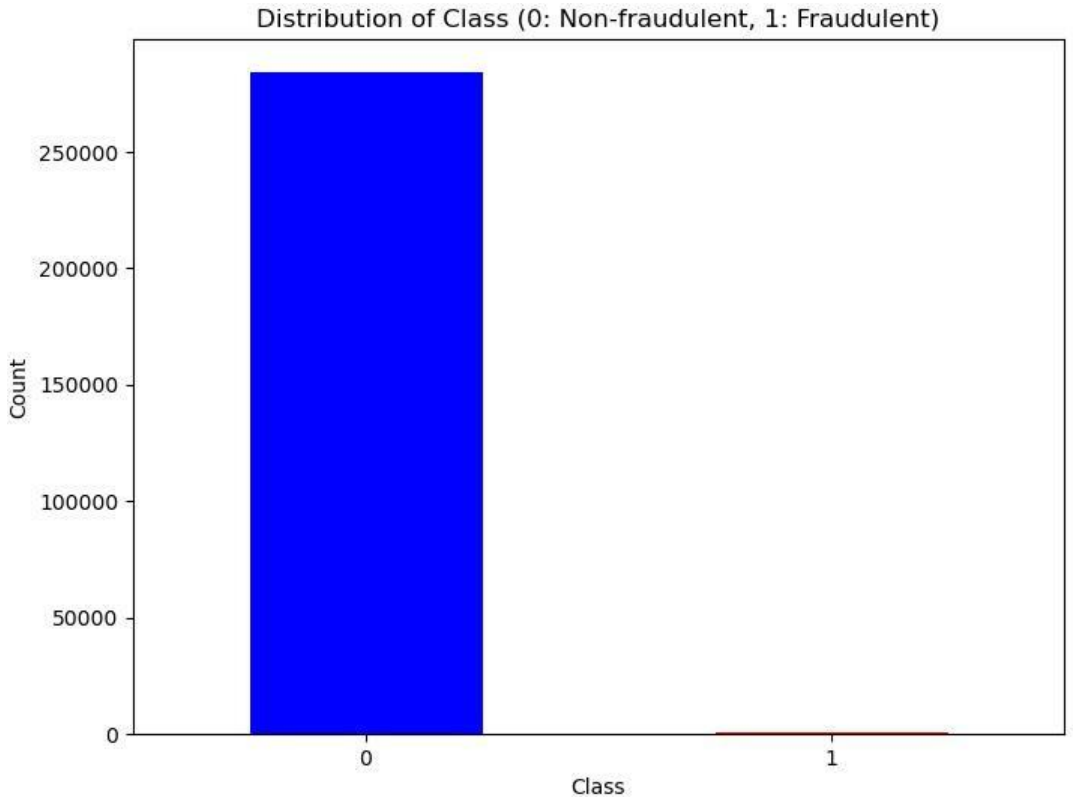
```
# Map selected PCA components back to original feature names
selected_features = [selected_features[i] for i in selected_indices]

print("Selected features using ANOVA F-test after PCA:")
print(selected_features)
# Split the data into training and validation sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Model Selection and Training
model = RandomForestClassifier(n_estimators=100, random_state=42)
# Train the model on the training data
model.fit(X_train, y_train)

# Cross-validation
cv_scores = cross_val_score(model, X_train, y_train, cv=5)
print("Cross-validation Scores:", cv_scores)
print("Mean Cross-validation Score:", np.mean(cv_scores))

# Model Evaluation
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
```

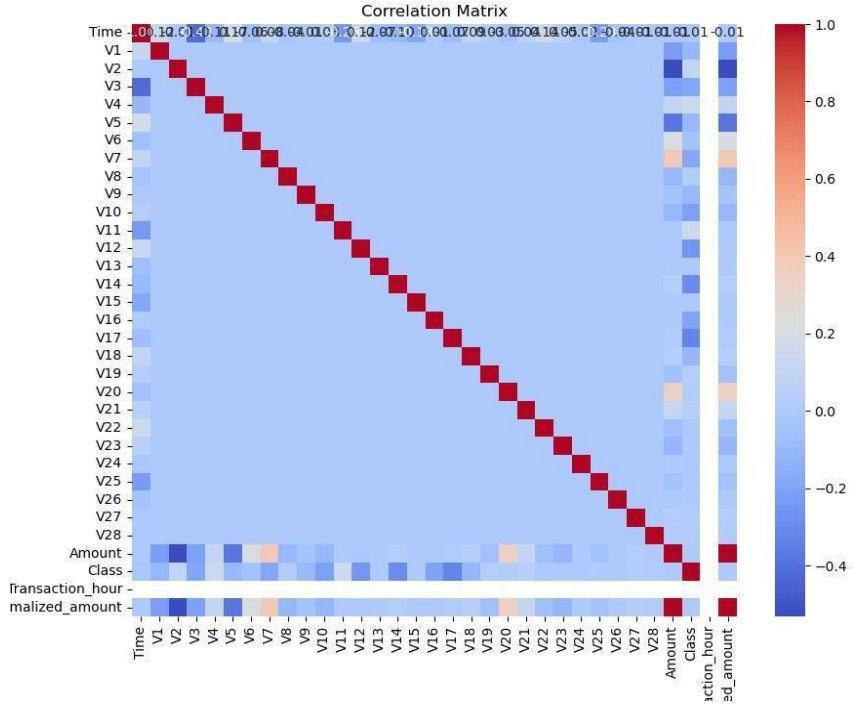
```
print("Classification Report:\n", class_report)

# Hyperparameter Tuning using GridSearchCV
param_grid = {
    'n_estimators': [1, 5, 10],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10]
}
grid_search = GridSearchCV(model, param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_
print("Best Hyperparameters:", best_params)

# Model Evaluation
y_pred = best_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", class_report)

# Save the best model
joblib.dump(best_model, 'credit_card_fraud_detection_model.pkl')
```

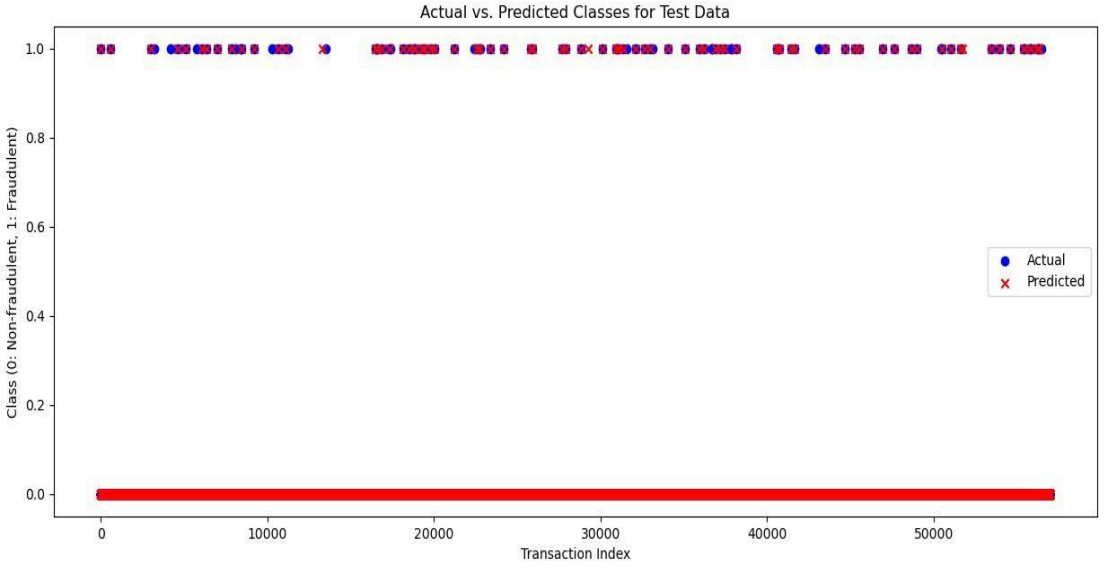
```
# Additional Visualizations
# Creating a heatmap for correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()

# Scatter plot to visualize the actual vs. predicted classes for test data
plt.figure(figsize=(15, 6))
plt.scatter(range(len(y_test)), y_test, color='blue', marker='o', label='Actual')
plt.scatter(range(len(y_test)), y_pred, color='red', marker='x', label='Predicted')
plt.xlabel('Transaction Index')
plt.ylabel('Class (0: Non-fraudulent, 1: Fraudulent)')
plt.title('Actual vs. Predicted Classes for Test Data')
plt.legend()
plt.show()

# Plot the transaction volume over time
plt.figure(figsize=(12, 6))
plt.plot(data['Time'], data['Amount'], color='blue', alpha=0.5)
plt.title('Transaction Volume Over Time')
plt.xlabel('Time')
plt.ylabel('Transaction Amount')
plt.grid(True)
plt.show()
```

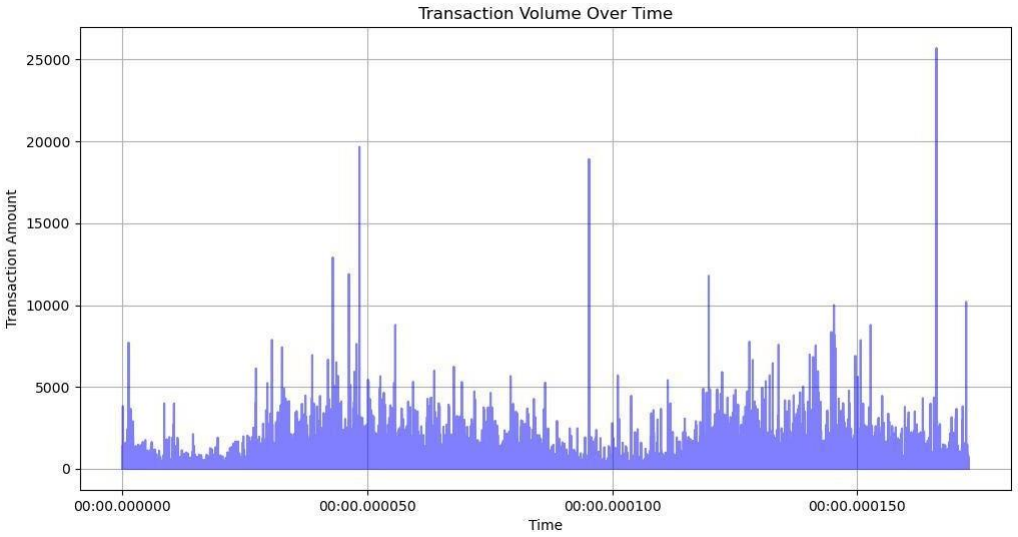
Visualisation:

```
Accuracy: 0.9995435553526912
Confusion Matrix:
 [[56860     4]
 [   22    76]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.95      0.78      0.85        98

    accuracy                           1.00     56962
   macro avg       0.97      0.89      0.93     56962
weighted avg       1.00      1.00      1.00     56962
```




Correlation Matrix
Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21
V22
V23
V24
V25
V26
V27
V28
Amount
Class
Transaction_hour
malized_amount
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


Actual vs. Predicted Classes for Test Data
Class (0: Non-fraudulent, 1: Fraudulent)
Transaction Index
Actual
Predicted
0.0
0.2
0.4
0.6
0.8
1.0
0
10000
20000
30000
40000
50000


Transaction Volume Over Time
Transaction Amount
Time
0
5000
10000
15000
20000
25000
00:00.000000
00:00.000050
00:00.000100
00:00.000150

CONCLUSION

Through this project, we have developed a robust solution for credit card fraud detection using predictive modelling techniques. By accurately identifying fraudulent transactions, we aim to enhance the financial integrity of both consumers and institutions. This project reflects our commitment to leveraging data science for the benefit of our company and its stakeholders.

Credit card fraud poses a significant risk to both consumers and financial institutions. In this report, we outline our approach to tackling this challenge through predictive modeling techniques. Our objective is to develop a robust solution capable of accurately identifying fraudulent transactions while minimizing false positives.

The source code for the pipeline can be found in the attached files. For installation and execution instructions, please refer to the README file included in the zip archive. If you have any further questions or require assistance, please don't hesitate to contact us.

Contact Information

For Further Inquiries

VIJAYAKUMAR R

vijay740125@gmail.com