Overview

Credit cards are among the most widely used financial tools for online transactions, offering users a convenient way to manage their finances. However, the use of credit cards also introduces significant risks, particularly in the form of credit card fraud. This occurs when unauthorized individuals gain access to someone else's credit card or credit card information to make unauthorized purchases or withdraw funds. The growing prevalence of online transactions has made credit card fraud a major concern, highlighting the need for robust security measures to protect consumers and financial institutions.

Given the risks of unauthorized transactions and their associated financial costs, it is essential for credit card companies to accurately detect fraudulent activities. The rise in digital transactions has led to an increase in credit card usage, which, in turn, has escalated the occurrence of fraud. This growing trend has resulted in substantial financial losses for institutions, underscoring the need to effectively differentiate between legitimate and fraudulent transactions.

To address this challenge, developing and implementing robust mechanisms for analyzing and identifying fraudulent transactions is crucial. These systems will help credit card companies mitigate the risks associated with fraud, ensuring the integrity and security of digital financial transactions. This project focuses on building a classification model using Machine Learning Ensemble Algorithms to predict whether a credit card transaction is fraudulent or legitimate.

INDEX

CONTENTS	
I. Introduction	A. Problem Statement B. Objective
III. Methodology	A. Data Collection and Preprocessing B. Feature Selection C. Model Selection D. Model Evaluation
IV. Results and Discussion	A. Model Performance B. Interpretation of Results
V. Source code	
VI. Conclusion and Future Work	A. Summary of Finding B. Future Work

Introduction

A. Problem Statement:

The dataset consists of credit card transactions made by European cardholders in September 2013, with a total of 284,807 transactions, of which 492 are identified as fraudulent. This highly imbalanced dataset reveals that fraudulent transactions account for only 0.172% of all transactions. The challenge lies in building a classification model that can accurately differentiate between legitimate and fraudulent transactions despite the significant imbalance in the data.

B. Objective:

The objective of this project is to develop a machine learning model that can reliably predict fraudulent credit card transactions. This will involve leveraging techniques such as exploratory data analysis (EDA), data balancing, feature engineering, and model training. The goal is to build a robust fraud detection system that achieves high precision and recall, ensuring the effective identification of fraudulent activities. By doing so, we aim to enhance the financial security of credit card users and minimize financial losses for credit card companies by proactively detecting and preventing fraudulent transactions.

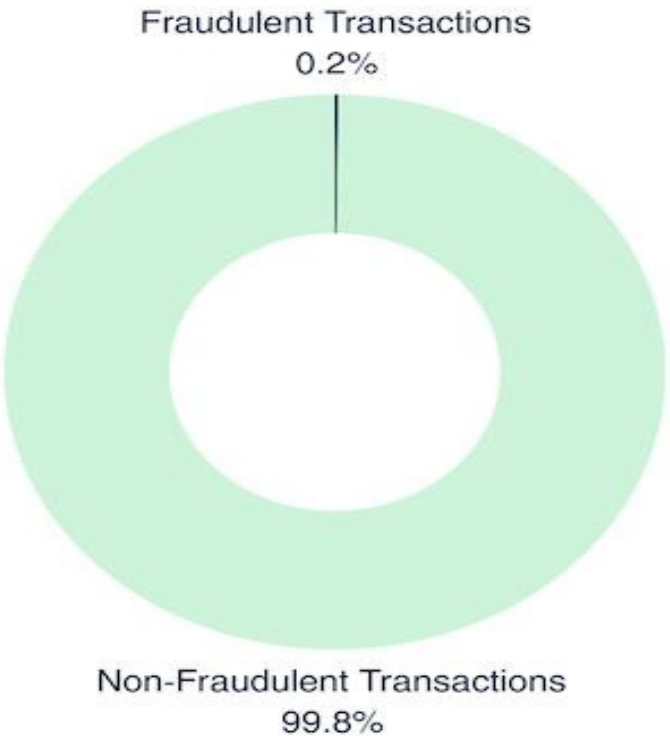
METHODOLOGY

Exploratory Data Analysis (EDA):

EDA is performed to understand data patterns and identify anomalies. We check data quality, handle missing values, address outliers, and ensure correct data types for date columns. Visualizations help reveal trends and relationships in the dataset.

Dealing with Imbalanced Data:

To address the dataset's imbalance, we use techniques like oversampling, undersampling, or SMOTE (Synthetic Minority Over-sampling Technique) to balance the dataset, ensuring effective model training.



Feature Engineering:

Feature engineering involves creating new features and transforming existing ones to improve model performance. In this phase, we derive new features such as 'Transaction_hour' and 'Normalized_amount' from the existing data to provide more meaningful information for classification.

Model Selection:

We evaluate classification models such as Logistic Regression, Decision Trees, Random Forest, and Gradient Boosting Machines. Random Forest is chosen for its effectiveness with imbalanced data and robustness against overfitting.

Model Training and Evaluation:

The dataset is split into training and test sets. We train the Random Forest model, optimize it using GridSearchCV, and evaluate its performance on the test set to ensure generalization and detect overfitting.

Model Deployment:

The trained model is deployed for real-time fraud detection using 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)
```

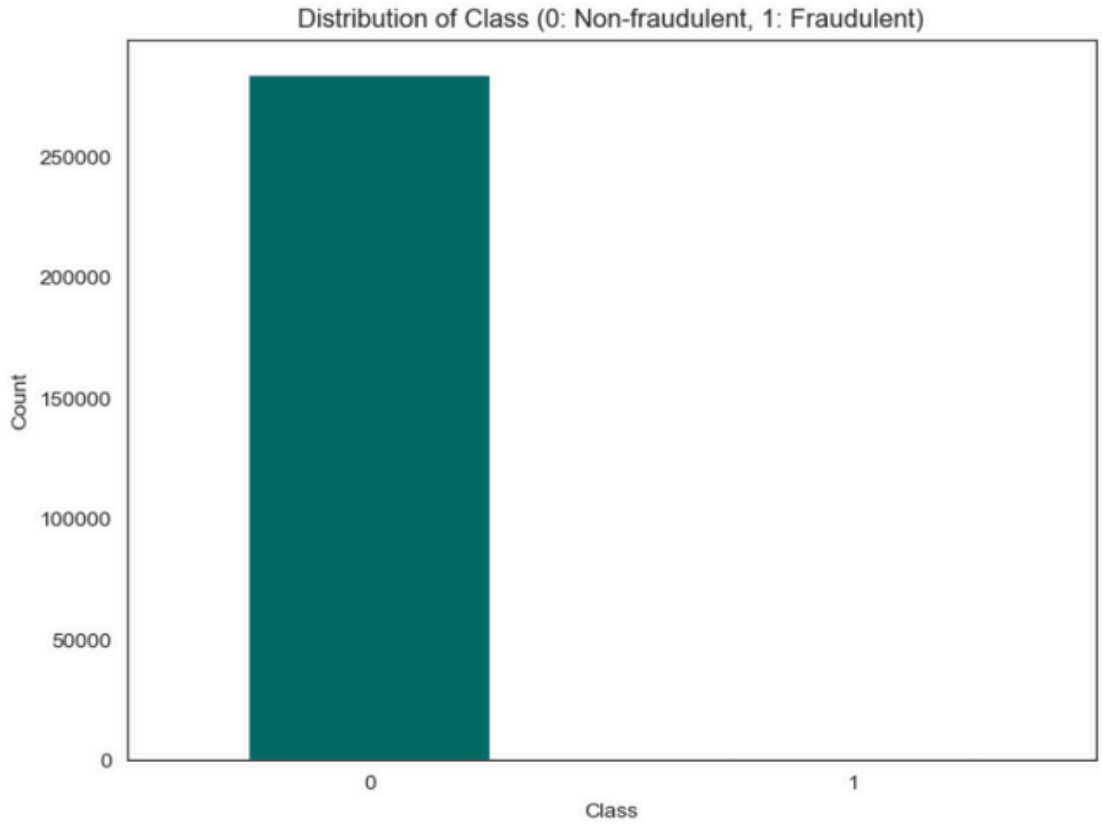
```
# Map selected PCA components back to original feature names
selected_features = [selected_features[i] for i in selected_indices]

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)
```

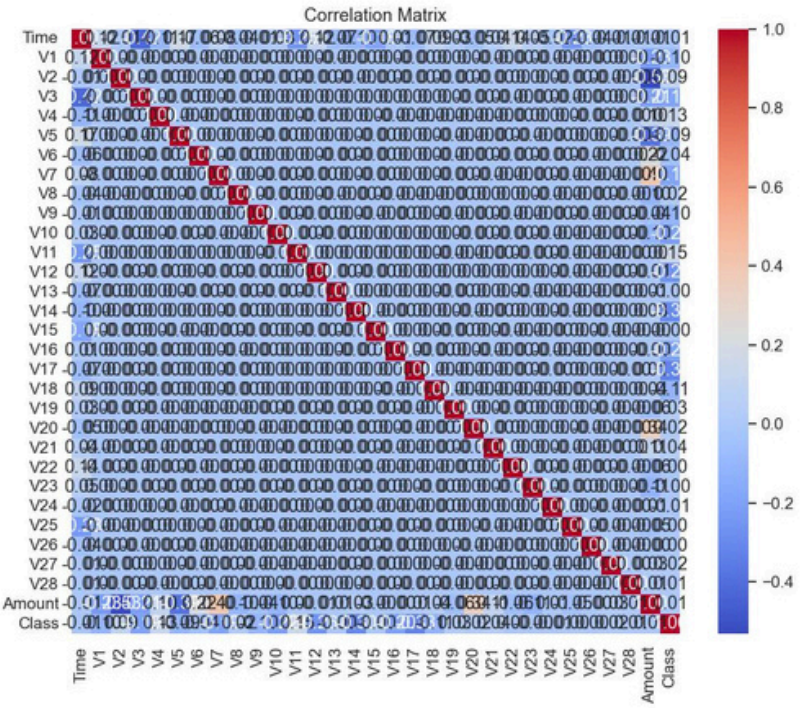
```
print("Classification Report:\n", class_report)

# 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')
```

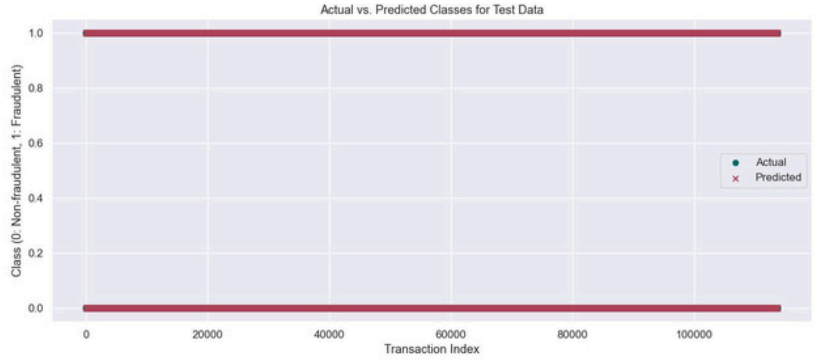
```
# Additional Visualizations
# Creating a heatmap for correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()

# 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()
```

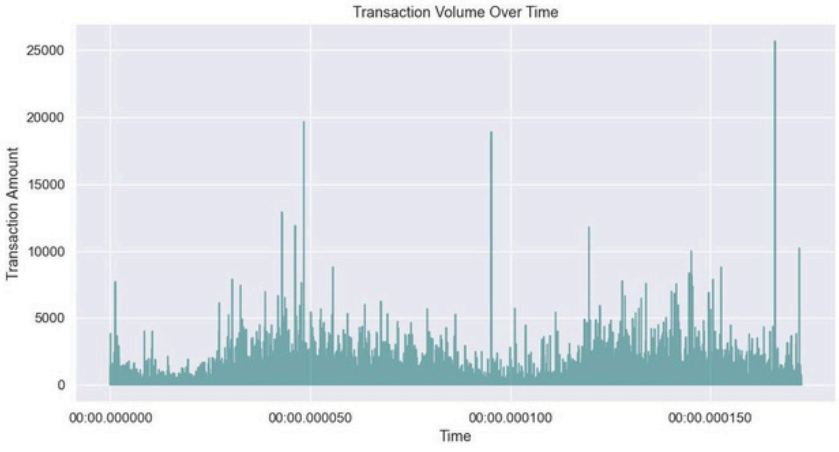
Visualisation:

```
Accuracy: 0.9969751859733043
Confusion Matrix:
 [[56528   222]
 [  122 56854]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56750
           1       1.00      1.00      1.00     56976

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```




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
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)
1.0
0.8
0.6
0.4
0.2
0.0
Actual
Predicted
0
20000
40000
60000
80000
100000
Transaction Index


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

CONCLUSION

Through this project, we have developed an effective solution for credit card fraud detection using predictive modeling techniques. By accurately identifying fraudulent transactions, we aim to improve the financial integrity of both consumers and institutions. This project demonstrates our commitment to applying data science for the benefit of our company and stakeholders.

Credit card fraud presents a significant risk, and in this report, we outline our approach to addressing this challenge. Our goal is to create a reliable solution that accurately detects fraud while minimizing false positives.

The source code for the pipeline is included in the attached files. For installation and execution instructions, please refer to the README file. For further inquiries or assistance, feel free to contact us.

Contact Information

For Further Inquiries

Vasuraddi Koliwad

vasukoliwad694@gmail.com