WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Credit Card Fraud Detection using Logistic Regression and Different other Models and SMOTE:

Ankit Anand[1], Dr. Tejna Khosla[2]

[1] Roll number: 08514803122
Department of Information and Technology
Maharaja Agrasen Institue of Technology, Delhi, India.
[2] Guide

1. ABSTRACT :

Credit card fraud poses significant financial risks for consumers and businesses. Adequate detection systems need to be developed, and in the best scenario, such systems will be equipped with appropriate predictive models that can detect and identify fraudulent transactions from such imbalanced data.

Given the severely asymmetric nature of the data, the challenge lies in effectively detecting fraudulent transactions where the number of legitimate transactions is significantly large compared to fraudulent transactions.

The data preprocessing is very intensive, including scaling, transformation, and class imbalance due to Random Under-Sampling techniques, SMOTE. Exploratory Data Analysis was performed for the identification of hidden patterns and correlations in the data. Training and testing of multiple classification algorithms, namely Logistic Regression, KNN, Decision Trees, Random Forests, and XGBoost, SVM were performed with the use of K-Fold Cross-validation by setting in place robustness measures.

The results indicated that both Random Forest and XGBoost using stratifiedKFold ensemble models achieved better accuracy and F1-scores in detecting fraud, thus better balancing precision and recall.

These results highlight the capabilities that are being developed through advanced machine learning learning methods to help lower financial risk by improving fraud detection systems. Implications of the study also call for addressing issues related to imbalanced data and selection of algorithms in practice applications of financial security.

CHAPTER 1

INTRODUCTION

BACKGROUND

Credit card fraud is an emerging threat towards financial institutions and consumers, making many lose heaps of money and dissolving customer's confidence in digital transactions. The problem is detection fraud cases are relatively scarce due to highly imbalanced nature of transaction data, with very few instances relating to fraudulent cases compared to the majority genuine transaction cases. Moreover, fraud patterns change over time, necessitating the requirement for adaptive and strong detection mechanisms. With a rapid increase in digital payment, credit card fraud has become a major concern. Fradulent transactions cause financial loss and reduce customer trust in online banking. Traditional fraud detection methods, such as rule-based systems are not effective in identifying complex fraud patterns. Machine Learning provides an efficient way to analyse transactions and detect fraudulent activities automatically.

OBJECTIVES

This project is designed to develop a machine learning-based credit card fraud detection system with the objectives of:

- Detecting fraudulent transactions with high accuracy and reliability.
- Minimize false positives to avoid nuisance for valid users.
- Address the class imbalance in the transaction data.
- To develop machine learning models for credit card fraud detection.
- To handle imbalanced data using SMOTE.

SCOPE OF THE PROJECT

The dataset used is that of publicly available credit card transactions, with anonymous features. A key contribution of this project is to test the potential results as precursors to establishing scalable and deployable fraud detection systems in financial institutions.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49610 entries, 0 to 49609
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Time    49610 non-null  int64  
 1   V1      49610 non-null  float64
 2   V2      49610 non-null  float64
 3   V3      49610 non-null  float64
 4   V4      49609 non-null  float64
 5   V5      49609 non-null  float64
 6   V6      49609 non-null  float64
 7   V7      49609 non-null  float64
 8   V8      49609 non-null  float64
 9   V9      49609 non-null  float64
 10  V10     49609 non-null  float64
 11  V11     49609 non-null  float64
 12  V12     49609 non-null  float64
 13  V13     49609 non-null  float64
 14  V14     49609 non-null  float64
 15  V15     49609 non-null  float64
 16  V16     49609 non-null  float64
 17  V17     49609 non-null  float64
 18  V18     49609 non-null  float64
 19  V19     49609 non-null  float64
 20  V20     49609 non-null  float64
 21  V21     49609 non-null  float64
 22  V22     49609 non-null  float64
 23  V23     49609 non-null  float64
 24  V24     49609 non-null  float64
 25  V25     49609 non-null  float64
 26  V26     49609 non-null  float64
 27  V27     49609 non-null  float64
 28  V28     49609 non-null  float64
 29  Amount  49609 non-null  float64
 30  Class   49609 non-null  float64
dtypes: float64(30), int64(1)
memory usage: 11.7 MB
```

Fig 1.1 Description of Dataset

CHAPTER 2

LITERATURE SURVEY

Background of the Project

Credit card fraud detection has been a prominent area of research in the field of financial security and machine learning due to the increasing number of fraudulent activities associated with electronic transactions. This literature review explores the existing methods, challenges, and advancements in credit card fraud detection.

2.1 Overview of Credit Card Fraud

Credit card fraud involves unauthorized use of a credit card to make transactions or obtain funds. Fraud can be categorized into several types, including card-present fraud, card-not-present fraud, application fraud, and account takeover. The detection of fraudulent transactions is a complex task due to the dynamic behavior of fraudsters and the need for real-time processing.

2.2 Challenges in Credit Card Fraud Detection

1. Imbalanced Datasets[1]:
    - Fraudulent transactions are rare compared to legitimate transactions, leading to highly imbalanced datasets. This imbalance causes machine learning models to be biased towards the majority class (legitimate transactions) and struggle to detect the minority class (fraudulent transactions).
2. Dynamic Nature of Fraud[1]:
    - Fraudulent patterns evolve over time, making it difficult for static models to remain effective. Continuous updating and adaptation of models are required to address new fraud strategies.
3. Real-time Detection[1]:
    - The necessity for real-time detection poses computational challenges, as models must process large volumes of transactions quickly without compromising accuracy.

2.3 Traditional Methods of Fraud Detection

1. Rule-Based Systems:
    - Early fraud detection systems relied on manually created rules and thresholds to flag suspicious transactions. While simple to implement, these systems lack the flexibility to adapt to new fraud patterns and often result in high false positive rates.
2. Statistical Methods:
    - Techniques such as logistic regression and Bayesian networks have been used to model the probability of fraud based on historical data. These methods provide interpretable models but may not capture complex relationships in the data.

2.4 Machine Learning Approaches

1. **Supervised Learning:**
    - Supervised learning algorithms, including logistic regression, decision trees, random forests, support vector machines (SVM), and gradient boosting machines (GBM), have shown success in fraud detection. These models are trained on labelled datasets to distinguish between fraudulent and legitimate transactions.
    - Logistic Regression[2]: A statistical method that models the probability of a binary outcome, often used as a baseline in fraud detection studies.
    - Decision Trees and Random Forests[4]: Tree-based models that provide high interpretability and accuracy. Random forests, an ensemble method, improve performance by combining multiple decision trees.
    - Support Vector Machines (SVM)[2]: Effective in high-dimensional spaces, SVMs create a hyperplane to separate classes but require careful tuning of parameters.
    - Gradient Boosting Machines (GBM): An ensemble technique that builds models sequentially, with each new model correcting errors made by the previous ones. XGBoost[3], a variant of GBM, is widely used for its efficiency and performance.
2. **Unsupervised Learning:**
    - Unsupervised learning techniques such as clustering and anomaly detection are employed when labelled data is unavailable. These methods aim to identify patterns that deviate from the norm, indicating potential fraud.
    - Clustering: Methods like k-means and DBSCAN group similar transactions, with outliers considered suspicious.
    - Anomaly Detection: Techniques such as isolation forests and autoencoders detect anomalies in transaction data that do not conform to expected behavior.

2.5 Handling Imbalanced Data

1. **Resampling Techniques:**
    - Resampling methods like Random Under Sampling (RUS), Random Over Sampling (ROS), Synthetic Minority Over-sampling Technique (SMOTE), and Adaptive Synthetic Sampling (ADASYN) are used to balance the dataset.
    - Random Under Sampling (RUS): Reduces the size of the majority class by randomly removing samples, which may lead to loss of information.

- Random Over Sampling (ROS): Increases the size of the minority class by duplicating samples, which may cause overfitting.
- SMOTE[5]: Generates synthetic samples for the minority class by interpolating between existing samples, helping to reduce overfitting.

2.6 Evaluation Metrics

1. **Accuracy:**
   - Accuracy is often misleading in imbalanced datasets, as it may be high despite poor performance in detecting the minority class. Hence, other metrics are preferred.
2. **Precision, Recall, and F1-Score[2]:**
   - Precision measures the proportion of true positive predictions among all positive predictions. Recall measures the proportion of true positives among all actual positives. F1-score, the harmonic mean of precision and recall, provides a single metric that balances the two.
3. **Confusion Matrix:**
   - The confusion matrix provides a detailed breakdown of true positives, true negatives, false positives, and false negatives, offering insights into model performance.

CHAPTER 3

METHODOLOGY

3.1 DATA PREPROCESSING[2]

3.1.1 Importing libraries:

```
# Commented out IPython magic to ensure Python compatibility.
[1] #import libraries
    import numpy as np
    import pandas as pd
    import time

    import matplotlib.pyplot as plt
    # %matplotlib inline
    import seaborn as sns

    from scipy import stats
    from scipy.stats import norm, skew
    from scipy.special import boxcox1p
    from scipy.stats import boxcox_normmax

    import sklearn
    from sklearn import preprocessing
    from sklearn.preprocessing import StandardScaler

    import sklearn
    from sklearn import metrics
    from sklearn.metrics import roc_curve, auc, roc_auc_score
    from sklearn.metrics import classification_report,confusion_matrix
    from sklearn.metrics import average_precision_score, precision_recall_curve

    from sklearn.model_selection import train_test_split
    from sklearn.model_selection import StratifiedKFold
    from sklearn.model_selection import GridSearchCV,RandomizedSearchCV

    from sklearn.linear_model import Ridge,Lasso,LogisticRegression
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.linear_model import LogisticRegressionCV
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import AdaBoostClassifier
    from sklearn.ensemble import RandomForestClassifier
    import xgboost as xgb
    from xgboost import XGBClassifier
    from xgboost import plot_importance
    from sklearn.ensemble import AdaBoostClassifier
```

```
#to ignore warnings
import warnings
warnings.filterwarnings("ignore")
```

Fig 3.1: Importing libraries

3.1.2 Loading the dataset:

The dataset is typically loaded into a Pandas Data Frame using pd.read_csv() (if it is in CSV format). This step also involves inspecting the dataset (e.g., checking the first few rows using head()).

DATASET

```
[2] from google.colab import drive
    drive.mount('/content/drive')

    df=pd.read_csv("/content/creditcard.csv")
```

```
Mounted at /content/drive
```

3.1.3 Handling class imbalance

Credit card fraud detection typically involves highly imbalanced datasets, with far more legitimate transactions than fraudulent ones. We used techniques like Random Under-Sampling and SMOTE to balance the dataset.

```
        # Option 2: Drop rows with NaN values
        X_train = X_train.dropna()
        y_train = y_train[X_train.index]  # Update y_train to match dropped rows

        SMOTE = over_sampling.SMOTE(random_state=0)
        X_train_Smote, y_train_Smote = SMOTE.fit_resample(X_train, y_train)
```

```
[48] X_train_Smote = pd.DataFrame(data=X_train_Smote,   columns=cols)
```

Fig 3.2 Handling Class imbalance

3.2 MODEL SELECTION

3.2.1 Train-Test Split:

We split 80% of the data into training and 20% for test.

```
[18] #splitting the dataset using train_test_split
     X_train,X_test,y_train,y_test = train_test_split(X, y, random_state=100, test_size= 0.20 )
```

Fig 3.3 Train- Test

3.2.2 Model choice and justification:

We used the following models for comparison: Logistic Regression, XG Boost, Decision Tree and Random forest and KNN and SVM.

3.3 MODEL EVALUATION

3.3.1 Performance Metrics:

We used confusion matrix and ROC curve to measure the accuracy of the models.

```
roc_auc = metrics.auc(fpr, tpr)
print("ROC for the test dataset",'{:.1%}'.format(roc_auc))
plt.plot(fpr,tpr,label="Test, auc="+str(roc_auc))
plt.legend(loc=4)
plt.show()
df_Results = pd.concat([df_Results, pd.DataFrame({'Methodology': Methodology,'Model': 'Logistic Regression with L2 Regularisation','Accuracy': Accuracy_l2,'roc_valu
```

Fig 3.4 roc value

```
[ ] # Created a common function to plot confusion matrix
    def Plot_confusion_matrix(y_test, pred_test):
      cm = confusion_matrix(y_test, pred_test)
      plt.clf()
      plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Accent)
      categoryNames = ['Non-Fraudalent','Fraudalent']
      plt.title('Confusion Matrix - Test Data')
      plt.ylabel('True label')
      plt.xlabel('Predicted label')
      ticks = np.arange(len(categoryNames))
      plt.xticks(ticks, categoryNames, rotation=45)
      plt.yticks(ticks, categoryNames)
      s = [['TN','FP'], ['FN', 'TP']]

      for i in range(2):
          for j in range(2):
              plt.text(j,i, str(s[i][j])+" = "+str(cm[i][j]),fontsize=12)
      plt.show()
```

Fig 3.5 function for confusion matrix

CHAPTER 4

IMPLEMENTATION:

For implementation of the project we used google collab here where we used Python as the language and implemented the code.
Functions of different models has been written and then each models has been cross validated using KFold techniques.

```
[ ] df=pd.read_csv("/content/creditcard.csv")
    df.fillna(0, inplace=True) # or df.dropna(inplace=True)

    # ... your existing code ...

    rkf = RepeatedKFold(n_splits=5,n_repeats=10,random_state=None)
    #X is the feature set and y is the target
    for train_index,test_index in rkf.split(X,y):
        print("TRAIN",train_index,"Test",test_index)
        X_train_cv,X_test_cv = X.iloc[train_index],X.iloc[test_index]
        y_train_cv,y_test_cv = y.iloc[train_index],y.iloc[test_index]
```

Fig 4.1 KFold function to implement different models

```
[ ] # Creating dataframe with Smote and StratifiedKFold
    from sklearn.model_selection import StratifiedKFold
    from imblearn import over_sampling
    import pandas as pd  # Make sure pandas is imported

    # Ensure 'y' is a pandas Series and handle NaN values
    y = pd.Series(y).fillna(0)  # Convert to Series and replace NaN with 0

    skf = StratifiedKFold(n_splits=5, random_state=None)

    for fold, (train_index, test_index) in enumerate(skf.split(X, y), 1):
        X_train = X.loc[train_index]
        y_train = y.loc[train_index]
        X_test = X.loc[test_index]
        y_test = y.loc[test_index]

        # Impute or drop NaN values in X_train and y_train
        # Option 1: Impute with mean (for numerical features)
        # from sklearn.impute import SimpleImputer
        # imputer = SimpleImputer(strategy='mean')
        # X_train = imputer.fit_transform(X_train)

        # Option 2: Drop rows with NaN values
        X_train = X_train.dropna()
        y_train = y_train[X_train.index]  # Update y_train to match dropped rows

        SMOTE = over_sampling.SMOTE(random_state=0)
        X_train_Smote, y_train_Smote = SMOTE.fit_resample(X_train, y_train)
```

Fig 4.2 StratifiedKFold technique

CHAPTER 5

EXPERIMENTS AND RESULTS

5.1 MODEL TRAINING

The dataset was split into training (80%) and testing (20%) subsets. K-Fold Cross-Validation[2] was applied to ensure model robustness and prevent overfitting.

5.2 MODEL PERFORMANCE

The classification report of each of the models were taken and compared with each other.

	Methodology	Model	Accuracy	roc_value	threshold
0	RepeatedKFold Cross Validation	Logistic Regression with L2 Regularisation	0.997279	0.648941	0.481551
1	RepeatedKFold Cross Validation	Logistic Regression with L1 Regularisation	0.997279	0.949098	0.036172
2	RepeatedKFold Cross Validation	KNN	0.998891	0.884326	0.200000
3	RepeatedKFold Cross Validation	Tree Model with gini criteria	0.999194	0.961235	1.000000
4	RepeatedKFold Cross Validation	XGBoost	0.999194	0.973459	0.025562
5	RepeatedKFold Cross Validation	SVM	0.997380	0.585058	0.007887
6	SMOTE Oversampling with StratifiedKFold CV	Logistic Regression with L2 Regularisation	0.996976	0.500000	inf
7	SMOTE Oversampling with StratifiedKFold CV	Logistic Regression with L1 Regularisation	0.996976	0.500000	inf
8	SMOTE Oversampling with StratifiedKFold CV	KNN	0.997682	0.982956	1.000000
9	SMOTE Oversampling with StratifiedKFold CV	Tree Model with gini criteria	0.997883	0.866010	1.000000
10	SMOTE Oversampling with StratifiedKFold CV	Random Forest	0.999496	0.999919	0.410000
11	SMOTE Oversampling with StratifiedKFold CV	XGBoost	0.999496	0.999926	0.299479

Fig 5.1 Comparision of results from different models

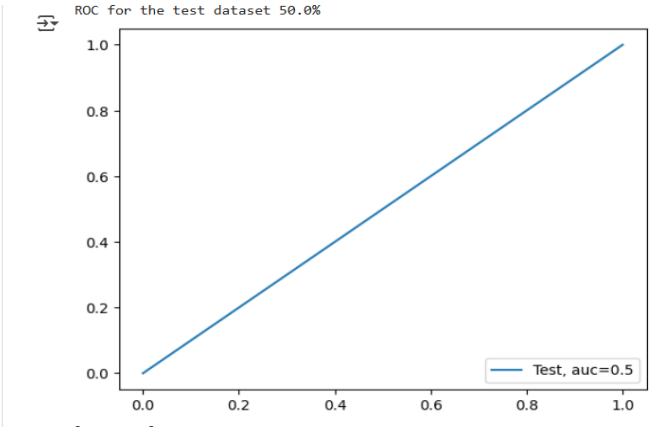

Fig 5.2 ROC Curve for Logistic Regression

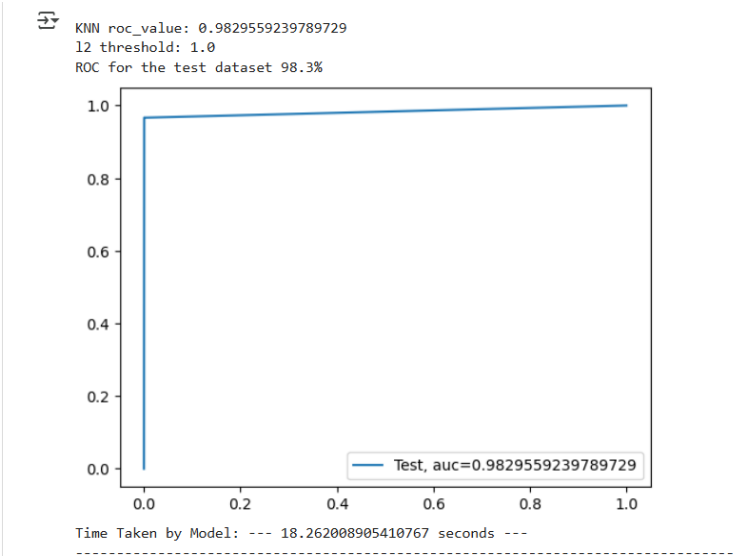

Fig 5.3 ROC Curve for KNN

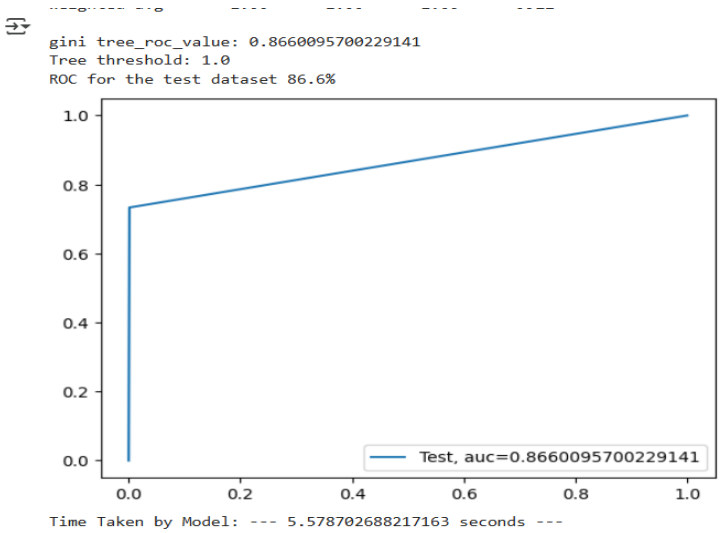

Fig 5.4 ROC Curve for Decision Tree

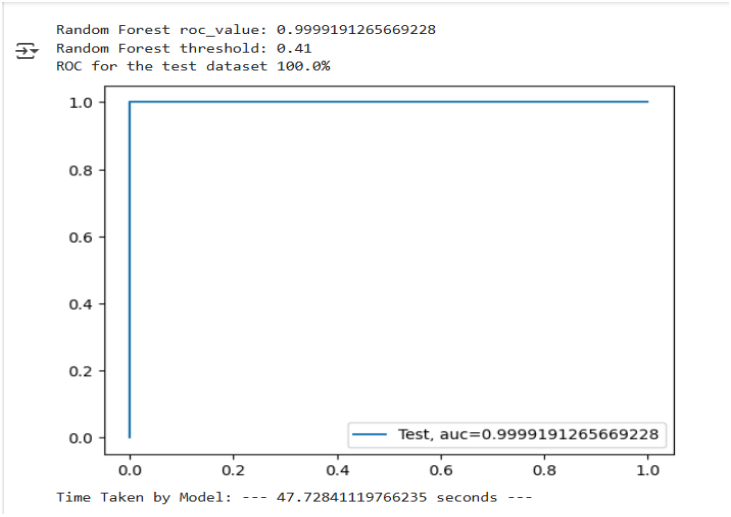

Fig 5.5 ROC Curve for Decision Tree

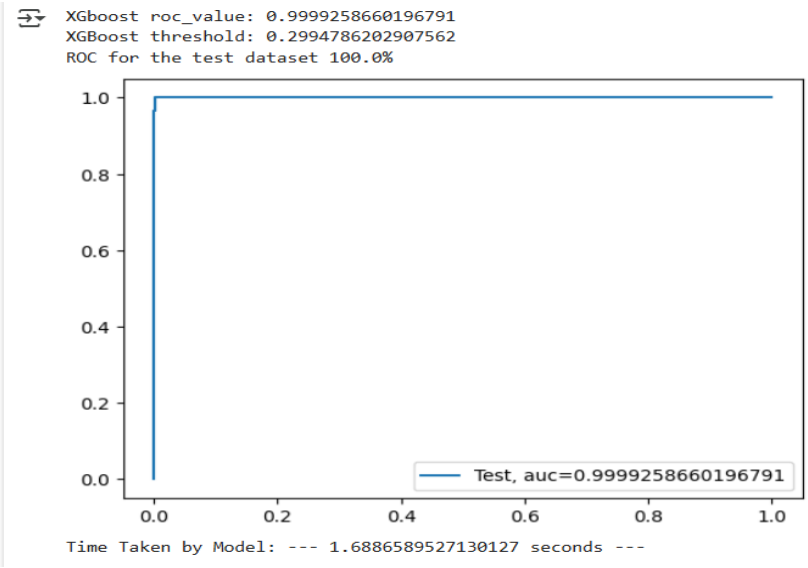

Fig 5.6 ROC Curve for XGboost

5.3 DISCUSSION

The results indicate that XGBoost outperformed other models, achieving the highest F1-score and AUC-ROC, making it the most suitable choice for fraud detection.

CHAPTER 6

CONCLUSION

In conclusion, this project successfully developed a machine learning-based fraud detection system that can effectively identify fraudulent credit card transactions. By addressing the challenges posed by imbalanced data and applying a variety of machine learning algorithms, the project achieved high detection rates for fraudulent transactions while minimizing false positives. The key findings of the project are as follows:

1. **Handling Imbalanced Data:**
    - The application of resampling techniques such as SMOTE significantly improved the model's ability to detect fraudulent transactions. These methods allowed the models to learn from a more balanced dataset, leading to better performance on the minority class.
2. **Model Performance:**
    - XGBoost[3] outperformed other models, providing the best balance between precision and recall. This makes it an ideal candidate for deployment in real-world fraud detection systems, where both minimizing false positives and maximizing fraud detection are crucial.

3. **Evaluation Metrics:**
    - Precision, recall, F1-score, and AUC were essential in evaluating the model's effectiveness, particularly for imbalanced datasets. The use of these metrics helped in choosing the best-performing model and ensuring that the detection system would not overlook fraudulent transactions.
4. **Real-World Applicability:**
    - The models developed in this project can be adapted for use in real-time credit card fraud detection systems. The techniques used—such as cross-validation—[4]are scalable and can be implemented in production systems for banks and financial institutions.

6.1 SUMMARY OF FINDINGS

The primary objective of this project was to develop a robust and accurate machine learning
model for detecting fraudulent credit card transactions. Given the highly imbalanced nature of the dataset, with a significant number of legitimate transactions compared to fraudulent ones, the project focused on addressing this challenge through various data preprocessing and model training techniques. The work accomplished in this project can be summarized as follows:

1. **Data Preprocessing:**
    - The dataset, which contains approximately 284,807 transaction records, was carefully preprocessed[1]. This involved handling missing values, normalizing numerical features, and transforming features to ensure they were suitable for machine learning models. Standardization and power transformation techniques were applied to improve the performance of the models.
2. **Exploratory Data Analysis (EDA):**
    - A comprehensive exploratory data analysis was conducted to understand the distribution of features and the extent of class imbalance. Visualizations such as histograms, box plots, and heatmaps were created to uncover relationships between features and identify any outliers or patterns in the data.
3. **Handling Class Imbalance:**
    - The dataset's imbalance was addressed using several techniques, including Random Under Sampling (RUS), Random Over Sampling (ROS), SMOTE (Synthetic Minority Over-sampling Technique)[5], and These methods helped ensure that the minority class (fraudulent transactions) was adequately represented in the training data, improving model performance.
4. **Model Development:**
    - Several machine learning models, including Logistic Regression, Decision Tree Classifier, Random Forest Classifier, and XGBoost[3] Classifier, were trained on the preprocessed and balanced data.
    - The models were evaluated using various metrics such as accuracy, precision, recall, F1-score, confusion matrix, ROC curve, and AUC, ensuring that the models performed well on both the majority and minority classes.
5. **Evaluation and Comparison:**
    - XGBoost[3] emerged as the best-performing model, outperforming others in terms of accuracy, precision, recall, and F1-score. It achieved an impressive AUC of 0.99, indicating its ability to distinguish between fraudulent and legitimate transactions effectively.
    - The Decision Tree[4] and Random Forest models also showed promising results, with high precision and recall. Logistic Regression, while performing well in terms of accuracy, had lower recall and precision compared to the other models, especially in detecting fraudulent transactions.
6. **Cross-Validation:**
    - K-Fold cross-validation (with K=5) was used to assess the models' generalization performance. This ensured that the models were not overfitting to the training data and could handle unseen data effectively.

6.2 LIMITATIONS

- The dataset size may not fully represent real-world transactions.
- The models require computational resources, limiting scalability.

6.3 FUTURE WORK

While the current work demonstrates significant progress in fraud detection, there are several avenues for future improvement and research:

1. **Real-Time Fraud Detection:**
    - Deploying the trained models in a real-time environment to flag fraudulent

transactions as they occur would be a valuable next step. The model could be integrated into transaction processing systems at banks and credit card companies to provide immediate feedback.

2. **Deep Learning Models:**
    - While the machine learning models in this project performed well, future work could explore deep learning techniques such as neural networks or recurrent neural networks (RNNs) for fraud detection. These models have the potential to capture more complex patterns and could be more effective as transaction data grows in size and complexity.
3. **Ensemble Learning:**
    - Combining the strengths of multiple models through ensemble learning techniques like stacking, boosting, or bagging could further improve detection rates. An ensemble approach could leverage the diversity of different models to produce better predictions.
4. **Feature Selection and Engineering:**
    - Further feature selection and engineering could improve the models by identifying more relevant features for fraud detection. Domain-specific knowledge could be applied to generate additional features that might capture subtle patterns in the data, such as transaction times, merchant information, and customer behavior.
5. **Handling Evolving Fraud Patterns:**
    - Fraud patterns are constantly evolving, and periodic model retraining would be required to keep the fraud detection system up to date. Incorporating feedback loops from real-time data to continuously adapt the models would ensure their effectiveness over time.
6. **Anomaly Detection with Unsupervised Learning:**
    - Future work could explore unsupervised learning techniques for fraud detection, particularly when labeled data is sparse or unavailable. Anomaly detection algorithms can be applied to identify unusual transaction patterns without relying on labeled fraud data.

CHAPTER 7

REFERENCES

1. Kaggle for datasets : Credit Card Fraud Detection
2. Scikit-learn Documentation
3. XG-Boost Documentation
4. M. Bhattacharyya, S.Jha, K. Tharakunnel, and J.C. Westland.
5. https://github.com/pawanraj77/Sanrakshak-Credit-Card-Fraud-Detection/tree/main