CREDIT CARD FRAUD DETECTION USING MACHINELEARNING

DOING BY

CHARISHMA POLINENI

Master's in computer science At Texas Tech University

ABSTRACT

Credit card fraud is a significant concern in the digital financial world, with fraudulent activities resulting in substantial financial losses for both individuals and financial institutions. This project aims to develop a machine learning-based system to detect fraudulent credit card transactions effectively. Using a highly imbalanced dataset consisting of 284,807 transactions, of which only 492 are fraudulent, various data preprocessing techniques were applied to handle the imbalance. We implemented and evaluated multiple classification models, including Logistic Regression, Random Forest, and XGBoost, with a focus on precision, recall, and F1-score to address the class imbalance. The final model successfully identified fraudulent transactions with high accuracy and minimal false positives, demonstrating the effectiveness of machine learning in real-time fraud detection systems. This project highlights the potential of automated fraud detection to enhance security in financial systems and reduce human intervention.

Introduction

📌 Background of the Project

With the rapid growth of online transactions and digital banking, credit card fraud has become a major challenge for financial institutions and consumers. Fraudsters are continuously finding new ways to bypass traditional security systems, making it crucial to implement intelligent systems that can detect unusual patterns and stop fraudulent activities in real time. Machine learning offers the ability to analyze large volumes of transaction data and identify potentially fraudulent behavior through pattern recognition and anomaly detection.

❓ What Problem Are Trying to Solve?

The key problem addressed in this project is the early and accurate detection of fraudulent credit card transactions from a large and highly imbalanced dataset. The challenge lies in distinguishing between legitimate and fraudulent transactions when the number of fraud cases is extremely small compared to normal ones. This imbalance can cause traditional algorithms to perform poorly in detecting rare but critical fraud cases. The goal is to build a model that can identify fraud with high precision and recall, minimizing both false positives and false negatives.

📌 Scope and Significance

This project demonstrates how machine learning can be applied to detect fraud effectively using historical transaction data. It covers the complete pipeline from data preprocessing and model training to performance evaluation. The significance of the project lies in:

- Enhancing financial security through automation
- Reducing manual effort and costs related to fraud investigation

- Demonstrating the use of machine learning for real-time risk prediction
- Offering insights into handling class imbalance in datasets

The techniques and insights from this project can be extended to other domains involving anomaly detection, such as insurance fraud, network intrusion detection, and more.

🛠Used Tools and Technologies

- Programming Language: Python
- Libraries/Packages:
    - Pandas and NumPy for data manipulation
    - Matplotlib and Seaborn for visualization
    - Scikit-learn for machine learning algorithms
    - Imbalanced-learn for handling class imbalance (SMOTE, etc.)
- IDE: Jupiter Notebook / Vs Code
- Version Control: Git and GitHub
- Dataset Source: Kaggle (Credit Card Fraud Detection dataset by European cardholders)

Problem Statement

❓ Describe the Actual Problem Solving

The problem addressed in this project is the detection of fraudulent credit card transactions from a large set of transaction records. Given the large volume of credit card transactions occurring daily, financial institutions face the challenge of identifying fraud in real time. Fraudulent transactions are rare events, and they constitute a small fraction of the entire dataset. This creates an imbalanced dataset where fraudulent transactions are vastly outnumbered by legitimate ones, making it difficult for traditional machine learning models to accurately predict fraud.

The goal of this project is to develop a machine learning model that can effectively identify fraudulent transactions while handling the class imbalance in the dataset. The model should be able to:

- Minimize false negatives (i.e., legitimate transactions classified as fraudulent)
- Minimize false positives (i.e., fraudulent transactions classified as legitimate)
- Be scalable to handle large datasets and real-time transaction streams

⚖️ Why Is It Important?

Credit card fraud detection is critical for:

- **Financial Security**: Fraudulent transactions lead to significant financial losses for both customers and financial institutions. Detecting fraud quickly helps minimize these losses.
- **Customer Trust**: Preventing fraud ensures that customers trust the banking system, thereby promoting continued usage of credit cards and other financial products.
- **Regulatory Compliance**: Financial institutions are required to comply with strict regulations and standards regarding transaction security. Automated fraud detection helps ensure compliance with these rules.

- **Operational Efficiency**: Manual fraud detection is time-consuming and resource intensive. Automating this process with machine learning improves efficiency and reduces the workload for human investigators.

💡 Assumptions Made

- **Class Imbalance**: It is assumed that fraudulent transactions are a rare occurrence compared to legitimate transactions. This imbalance must be addressed using techniques like oversampling, undersampling, or using algorithms that handle imbalanced datasets.
- **Feature Quality**: The project assumes that the features available in the dataset (e.g., transaction amount, time, location) are sufficient to detect fraudulent behavior and that data cleaning has been performed.
- **No External Data**: The model is trained and tested solely on the dataset provided, without any additional external data sources for feature enrichment.
- **No Real-Time Constraints**: The model is not optimized for real-time processing in this project, but it can be adapted for real-time fraud detection with further work.

Objectives

The main goals of this project are to design and implement a machine learning model that can detect fraudulent credit card transactions accurately. The specific objectives include:

1. **Detect Fraudulent Credit Card Transactions with High Accuracy**
    - The primary objective is to build a model that can **accurately identify fraudulent transactions** from a large dataset. This involves achieving a balance between precision (minimizing false positives) and recall (minimizing false negatives), especially given the highly imbalanced nature of the dataset.
2. **Handle Class Imbalance Effectively**
    - Since fraudulent transactions represent only a small percentage of the entire dataset, a key objective is to address the **class imbalance** issue. This will involve using techniques such as **SMOTE (Synthetic Minority Over-sampling Technique)**, **undersampling the majority class**, or training models that are specifically designed to deal with imbalanced data, such as **XGBoost**.
3. **Preprocess the Data and Extract Relevant Features**
    - Another objective is to **clean and preprocess the dataset** to ensure that the features are usable for training machine learning models. This includes handling missing values, scaling the data, encoding categorical variables, and ensuring that features such as time, amount, and transaction type are properly formatted.
4. **Evaluate Multiple Machine Learning Models**
    - To ensure the best possible performance, the project will involve the **evaluation of multiple machine learning algorithms** like **Logistic Regression**, **Random Forest**, **Support Vector Machines (SVM)**, and **XGBoost**. The goal is to select the model that performs best in detecting fraudulent transactions using metrics like **accuracy**, **precision**, **recall**, **F1-score**, and **ROC-AUC**.
5. **Minimize False Positives and False Negatives**
    - A critical goal is to build a system that not only detects fraud but does so in a way that minimizes **false positives** (legitimate transactions flagged as fraudulent) and **false negatives** (fraudulent transactions

missed by the model). This is essential to avoid disrupting customers' experience and ensuring the model's reliability.

6. **Deploy the Model for Practical Use**
    - While the focus of the project may be on building and evaluating the model, an objective is also to design it in such a way that it can be **deployed in real-world systems**. This means ensuring that the model can handle large datasets and provide timely predictions, possibly integrating with a transactional system for real-time fraud detection.
7. **Understand the Impact of Different Features on Fraud Detection**
    - One of the secondary objectives is to understand which features (e.g., **transaction amount**, **location**, **time of transaction**) have the **most significant impact on fraud detection**. This will help in understanding the data better and refining the model further.

Methodology / Approach

The methodology for the **Credit Card Fraud Detection** project involves a systematic approach, leveraging machine learning techniques to identify fraudulent transactions from a highly imbalanced dataset. The steps followed include **Data Collection**, **Data Preprocessing**, **Exploratory Data Analysis (EDA)**, **Model Selection**, **Training and Testing**, **Evaluation Metrics**, and **Deployment**.

1. Data Collection

- The first step in the project is to obtain a dataset for training and evaluation. In this case, the dataset is sourced from **Kaggle's Credit Card Fraud Detection dataset**, which contains transaction records of European cardholders, including both fraudulent and legitimate transactions.
- The dataset consists of **284,807 transactions**, with only **492 fraudulent transactions** (making it highly imbalanced). This dataset includes multiple features such as the transaction amount, time, and anonymized credit card features.

---

2. Data Preprocessing

- **Handling Missing Values**: The first task in data preprocessing is to check for any missing or incomplete values in the dataset. If any are found, appropriate strategies such as **imputation** or **removal of missing data** are applied.
- **Feature Scaling**: Since machine learning models require numerical inputs, non-numerical data (if any) is encoded. In this project, features are scaled using **StandardScaler** (or other methods) to normalize the data, ensuring each feature contributes equally to model performance.
- **Class Imbalance Handling**: Due to the class imbalance (fraudulent transactions are very few), techniques like **oversampling** (e.g., using SMOTE) or **undersampling** are employed to balance the dataset, ensuring the model can effectively learn from the minority class.
- **Data Splitting**: The dataset is divided into **training** and **testing** sets, usually with a ratio of **80:20** or **70:30**, to ensure that the model can be trained on one set and validated on another.

---

3. Exploratory Data Analysis (EDA)

- EDA is crucial for understanding the dataset and uncovering patterns that could inform model design. It includes:
    - **Data Visualization**: Visualizations such as **histograms**, **box plots**, and **scatter plots** help in identifying data distributions and relationships between features.
    - **Correlation Analysis**: This identifies the correlation between features, allowing us to see which features might be important for detecting fraud.
    - **Class Distribution**: A key part of EDA is visualizing the class imbalance in the dataset. This step helps in understanding how significant the imbalance is and guides decisions on how to handle it.

---

4. Model Selection

- Various machine learning algorithms are considered for the project, with an emphasis on models that handle imbalanced data effectively. The models selected for training are:
    - **Logistic Regression**: A simple, interpretable model used for binary classification.
    - **Random Forest**: A robust ensemble method that can deal with high-dimensional data and provide feature importance.
    - **XGBoost**: An advanced boosting technique known for its high performance in classification tasks.
    - **Support Vector Machine (SVM)**: Known for its ability to perform well in high-dimensional spaces.
- The goal is to identify which model provides the best balance between **accuracy**, **precision**, **recall**, and **F1-score**, especially in the context of fraud detection.

---

5. Training and Testing

- **Training**: The selected models are trained using the **training dataset**. This step involves feeding the features and corresponding labels (fraudulent or not) to the model to allow it to learn the underlying patterns.
- **Hyperparameter Tuning**: For some models (e.g., Random Forest, XGBoost), hyperparameters are tuned using **GridSearchCV** or **RandomizedSearchCV** to find the best combination of parameters for optimal performance.
- **Testing**: After training, the model is tested on the **testing dataset**, which is data that the model has not seen during training. This helps in evaluating how well the model generalizes to new, unseen data.

---

6. Evaluation Metrics

- Since the dataset is highly imbalanced, accuracy alone is not sufficient to evaluate model performance. The following metrics are used:
    - **Accuracy**: Percentage of correct predictions made by the model (though this is not the primary metric due to class imbalance).
    - **Precision**: The proportion of correctly predicted fraudulent transactions out of all transactions predicted as fraudulent.
    - **Recall**: The proportion of correctly predicted fraudulent transactions out of all actual fraudulent transactions.
    - **F1-score**: The harmonic mean of precision and recall, providing a balance between the two.
    - **ROC-AUC**: Measures the area under the receiver operating characteristic curve, which illustrates the trade-off between sensitivity and specificity.
    - **Confusion Matrix**: Helps in visualizing the performance of the classification model by showing the true positives, false positives, true negatives, and false negatives.

---

7. Deployment (If Applicable)

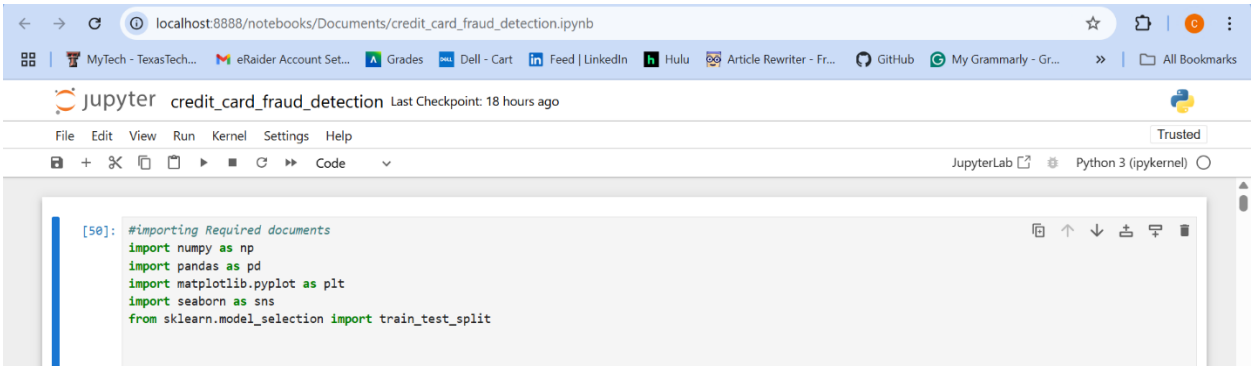
- While this project focuses on model development and evaluation, the final goal is to **deploy the trained model** for real-time use. This could involve:
    - **Integration with Transaction Systems**: The model can be integrated with financial transaction systems, where it evaluates transactions in real-time and flags potential fraud.
    - **Model API**: The model could be deployed as a **RESTful API** using frameworks such as **Flask** or **FastAPI**, allowing other systems to query the model for fraud detection on new transactions.
    - **Continuous Monitoring and Retraining**: As fraud patterns evolve, the model might need to be retrained periodically. Automated retraining pipelines can be set up to improve model accuracy over time.

Dataset Description

- **Source**: The dataset is publicly available on Kaggle — Credit Card Fraud Detection Dataset.
- →https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
- **Records**: The dataset contains **284,807 transaction records**.
- **Features**: There are **31 columns**, including:
  - Time: Time in seconds since the first transaction.
  - V1 to V28: Result of PCA transformation for confidentiality (feature-engineered).
  - Amount: The amount of the transaction.
  - Class: Target variable; **1 = Fraudulent**, **0 = Legitimate**.
- **Data Cleaning**:
  - Checked for missing values — none found.
  - Normalized the Amount and Time fields.
  - Addressed class imbalance using **SMOTE** or **undersampling**.
  - Scaled features using **StandardScaler** for better model performance.

Implementation

1.Importing Required documents:-



2.Data Preprocessing:-

2.1 Import data set:-

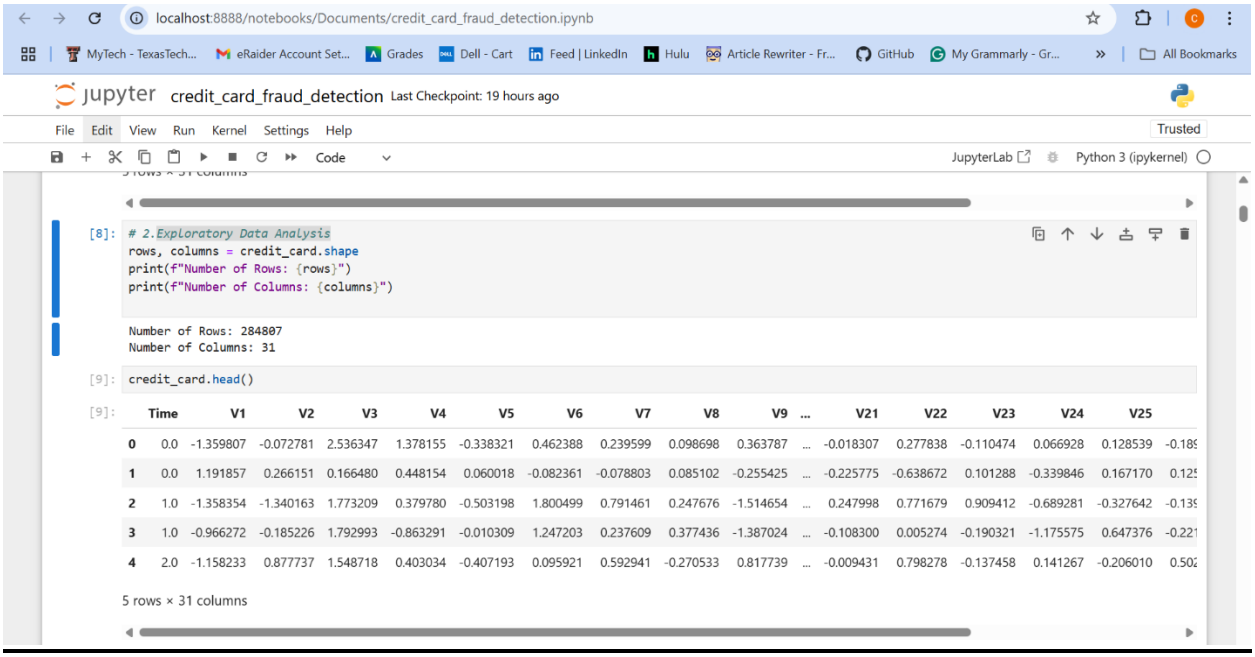
```
[7]:
#DATA PREPROCESSING#
# 1.Import data set
import pandas as pd
credit_card = pd.read_csv("creditcard.csv")
credit_card.head()
```

[7]:

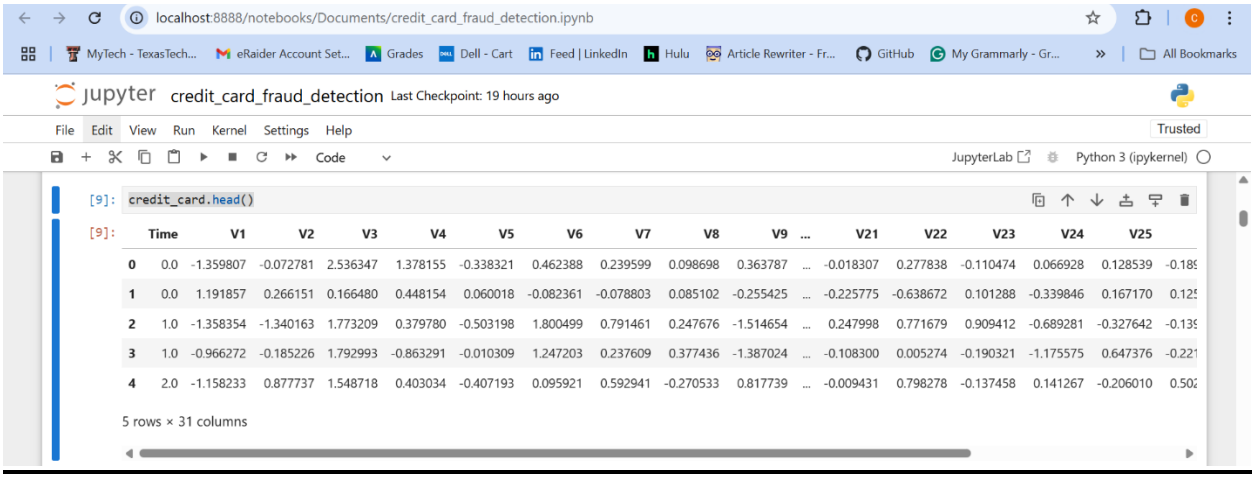
	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502

5 rows × 31 columns

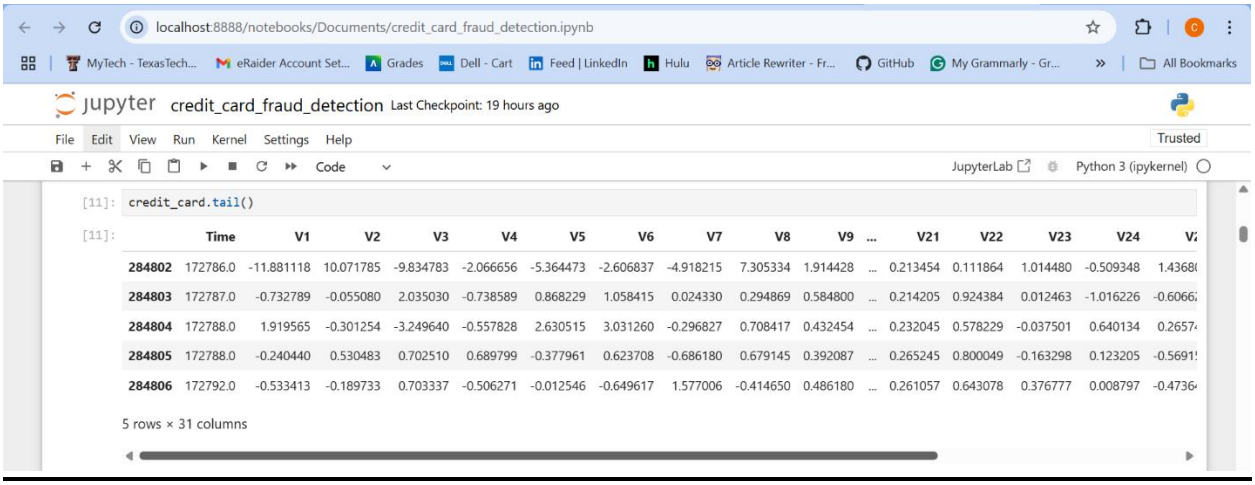
2.2 Exploratory Data Analysis:



2.3 Head():-



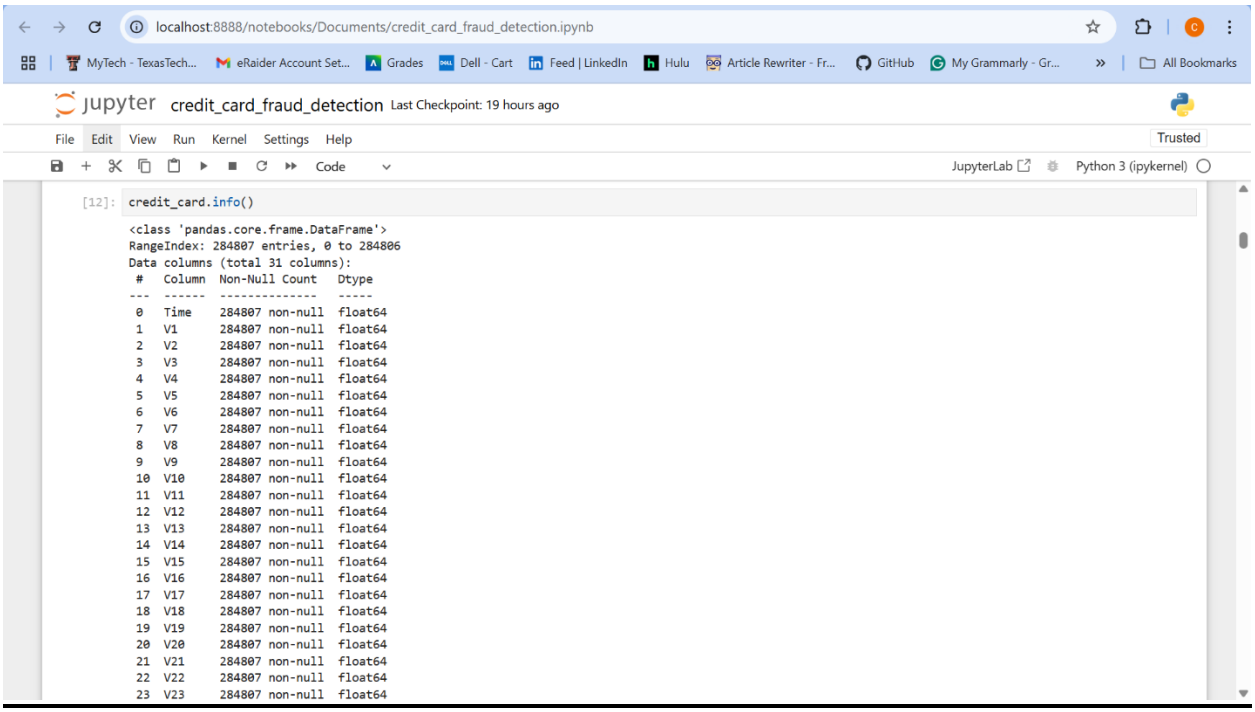
2.4 Tail():-



	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.014480	-0.509348	1.4368
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.016226	-0.6066
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.640134	0.2657
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.123205	-0.5691
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.008797	-0.4736

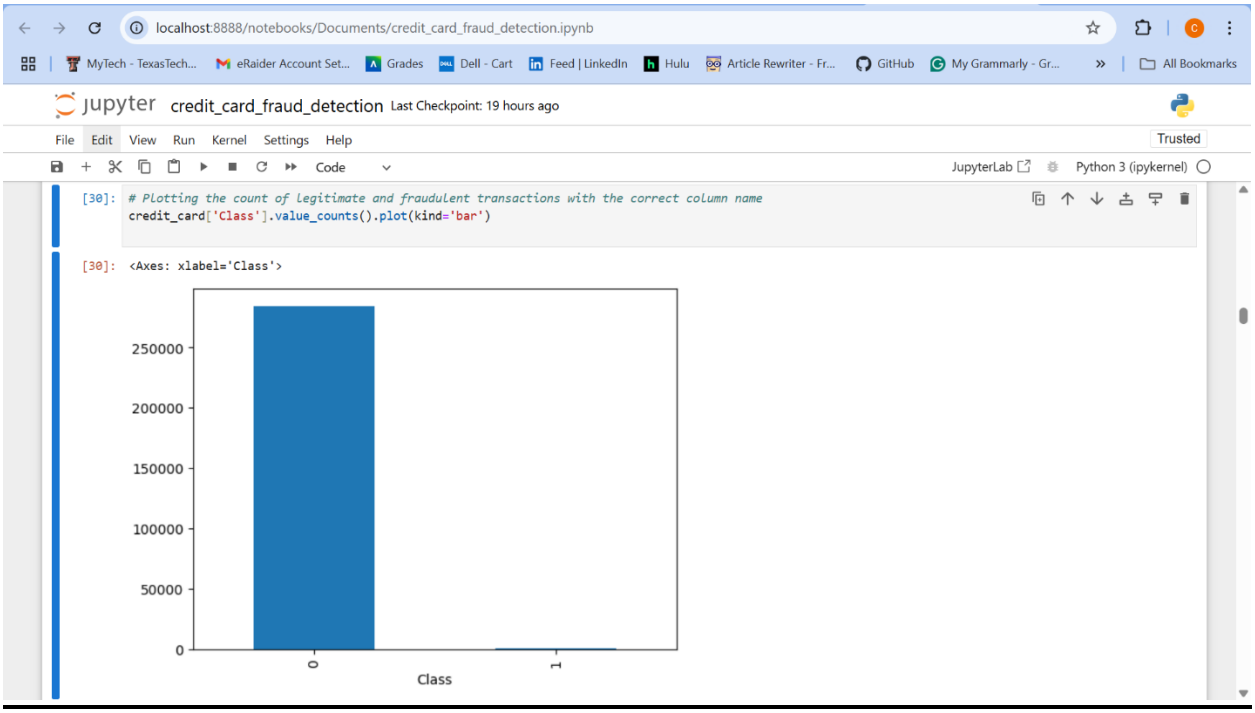
5 rows × 31 columns

2.5 Info():-



3.Legitimate & Fraudulent Transactions:

3.1 Plotting Graph:-



###1. Understand the Problem:

Imbalanced Data: This occurs when one class (e.g., legitimate transactions) outnumbers the other (e.g., fraudulent transactions). This can lead to biased models that predict the majority class more frequently.

2. Resampling Techniques:

A. Oversampling the Minority Class:

SMOTE (Synthetic Minority Over-sampling Technique): Creates synthetic samples for the minority class by generating new examples based on existing data points.

Random Oversampling: Randomly duplicates examples from the minority class to balance the dataset.

B. Undersampling the Majority Class:

Random Undersampling: Removes random examples from the majority class to balance the dataset, but this can lead to the loss of useful data.

Tomek Links: Removes examples from the majority class that are very close to the minority class to clean up the data.

C. Hybrid Methods:

SMOTE + ENN (Edited Nearest Neighbors): Combines oversampling (SMOTE) with cleaning techniques (ENN) to reduce noise and balance the data.

3. Algorithm-Level Techniques:

Class Weighting: Many algorithms like logistic regression or random forests allow you to assign higher weights to the minority class, making the model pay more attention to it.

Cost-Sensitive Learning: Modify the learning algorithm so that it penalizes misclassifications of the minority class more heavily.

4. Use Appropriate Evaluation Metrics:

Accuracy is not reliable in imbalanced datasets because predicting the majority class all the time can still give high accuracy.

Precision, Recall, and F1-Score: Focus on how well the model performs on the minority class.

Confusion Matrix: Helps you understand false positives and false negatives.

ROC-AUC: Measures how well the model distinguishes between classes, which is especially important for imbalanced data.

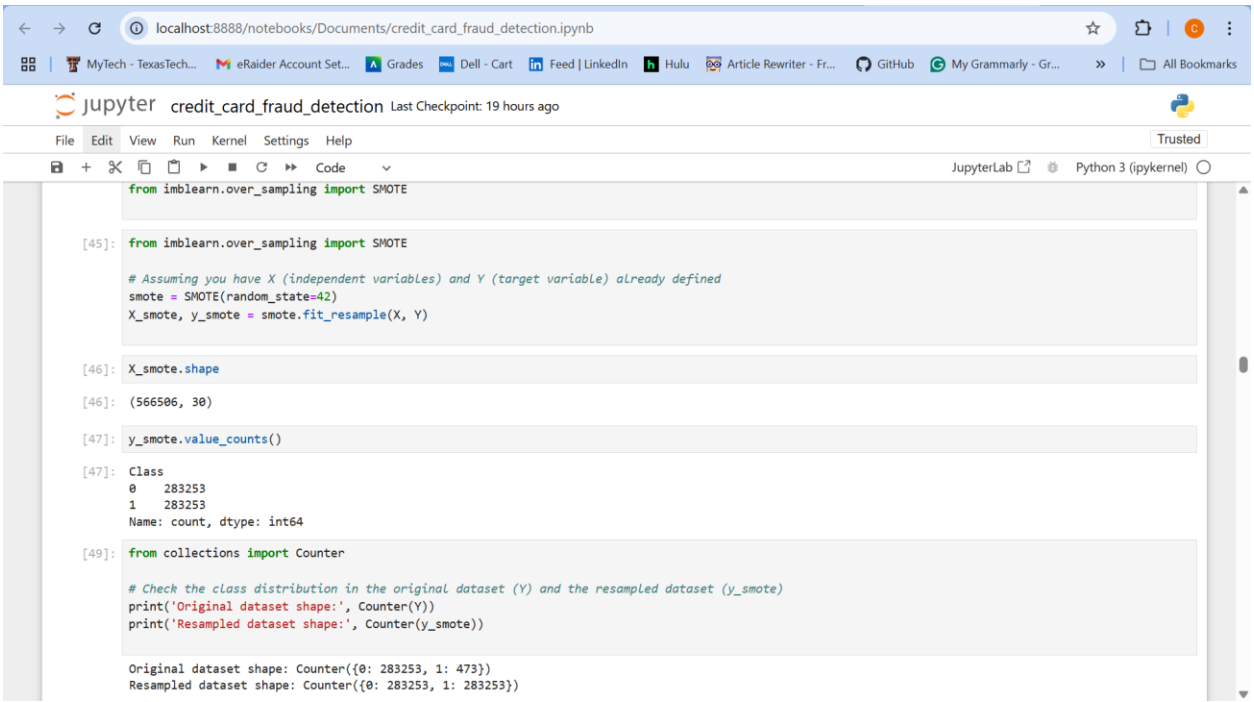
5. Cross-Validation:

Use Stratified Cross-Validation, which ensures each fold of the data contains the same proportion of classes as the original dataset.

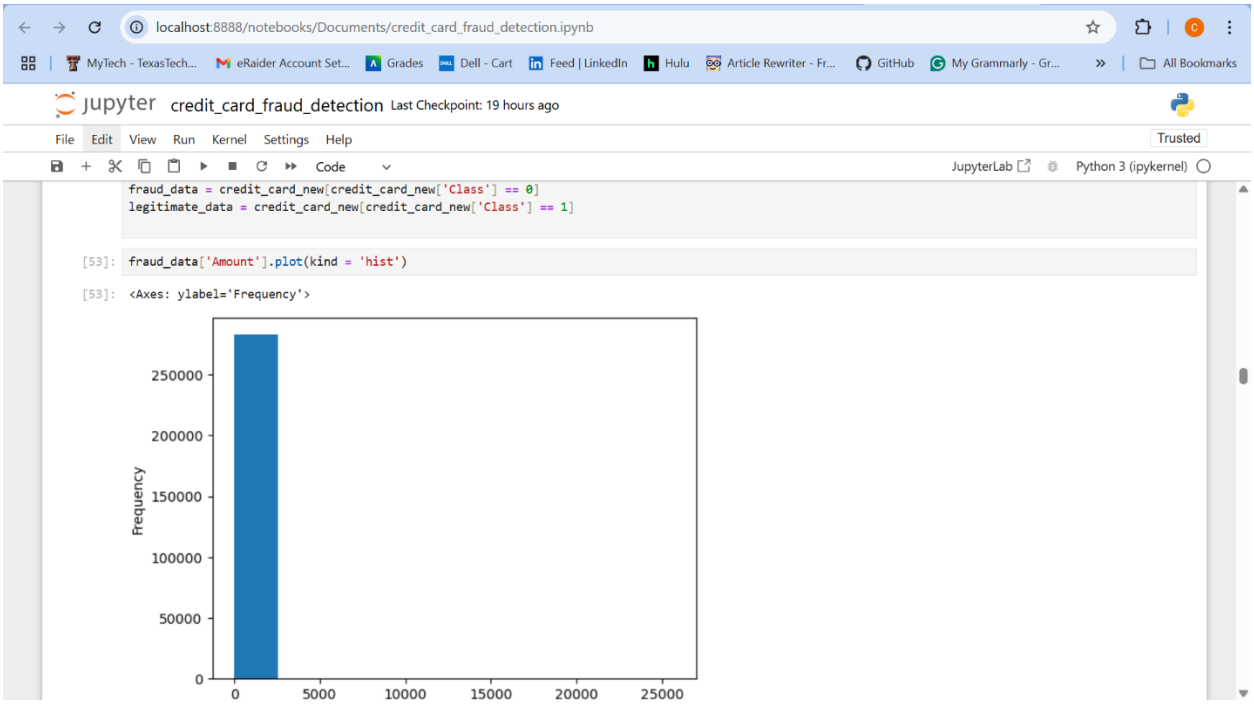
6. Try Specialized Models:

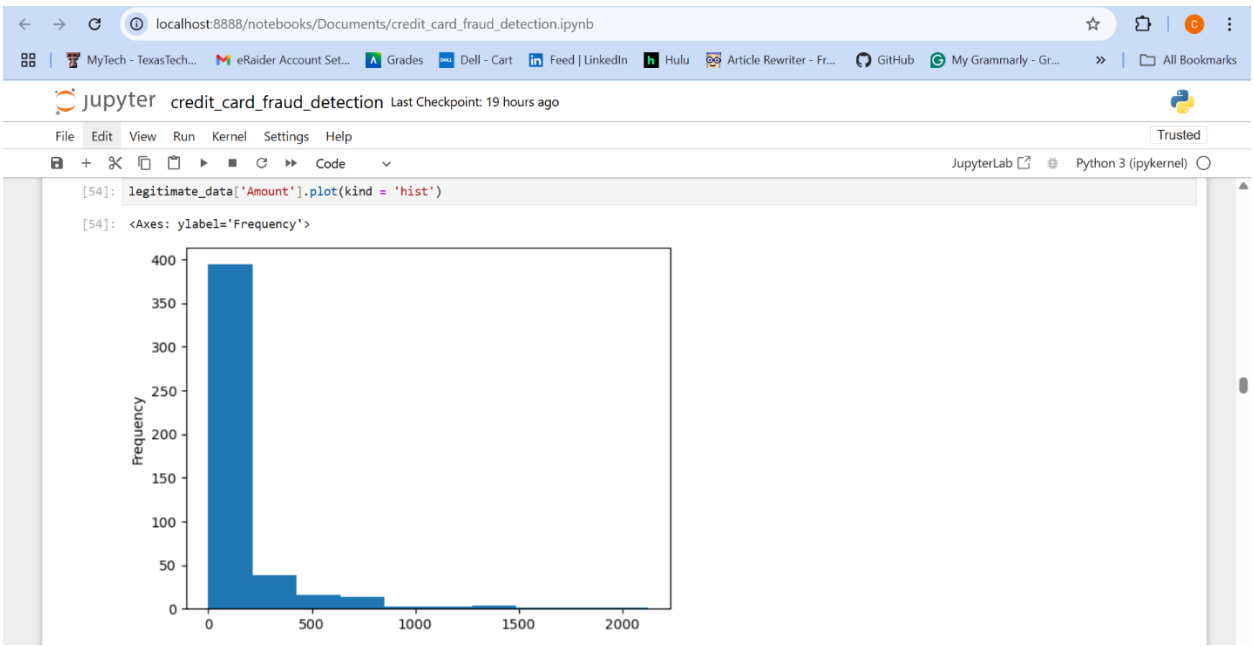
Some models like XGBoost and LightGBM have built-in support to handle imbalanced data.

Ensemble Methods like Balanced Random Forests or Easy Ensemble can be helpful for better classification of imbalanced classes.###



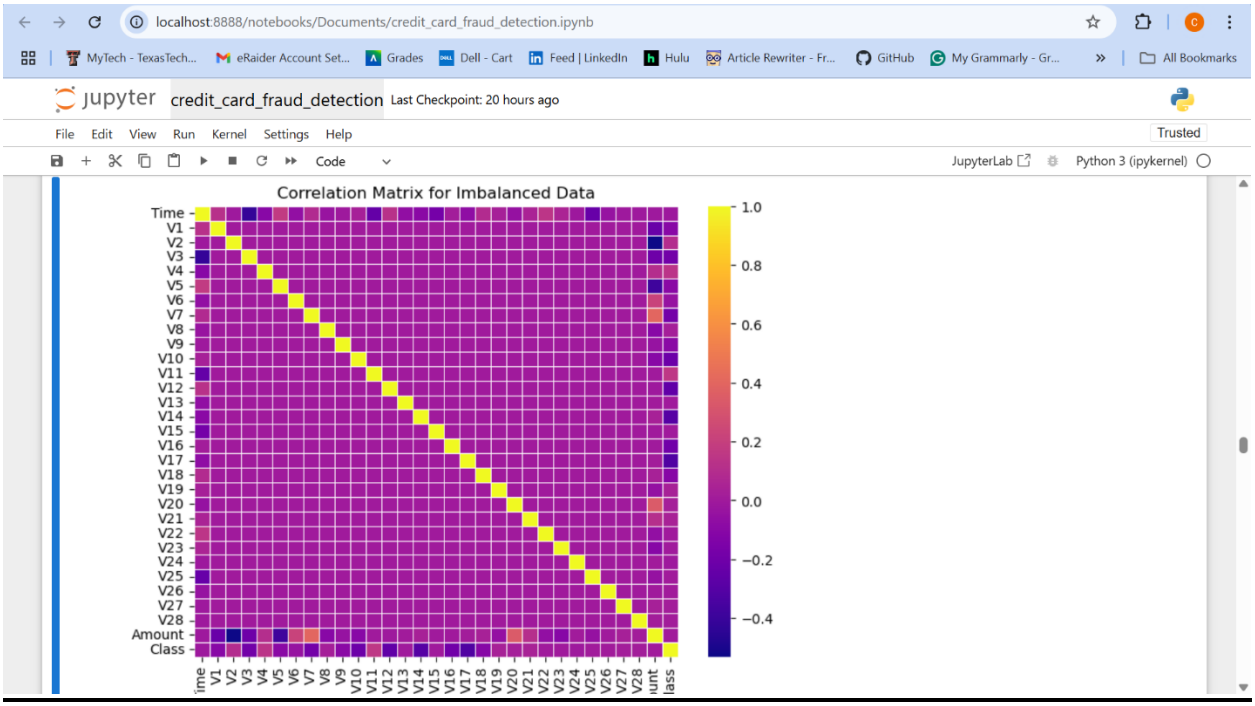
4.Data Visualization





4.1 Correlation Matrix for Imbalanced Data:-

```
[62]: import matplotlib.pyplot as plt  # Importing for plotting
      import seaborn as sns  # Importing seaborn for heatmap

      # Correlation Matrix for Imbalanced Data
      corr_imbalanced = credit_card.corr()  # Compute the correlation matrix

      plt.figure(figsize=(8, 6))  # Set the figure size
      sns.heatmap(corr_imbalanced, annot=False, cmap="plasma", linewidths=0.5)  # Plot the heatmap
      plt.title("Correlation Matrix for Imbalanced Data")  # Set the title
      plt.show()  # Display the plot
```



4.2 Correlation Matrix for Balanced Data:-

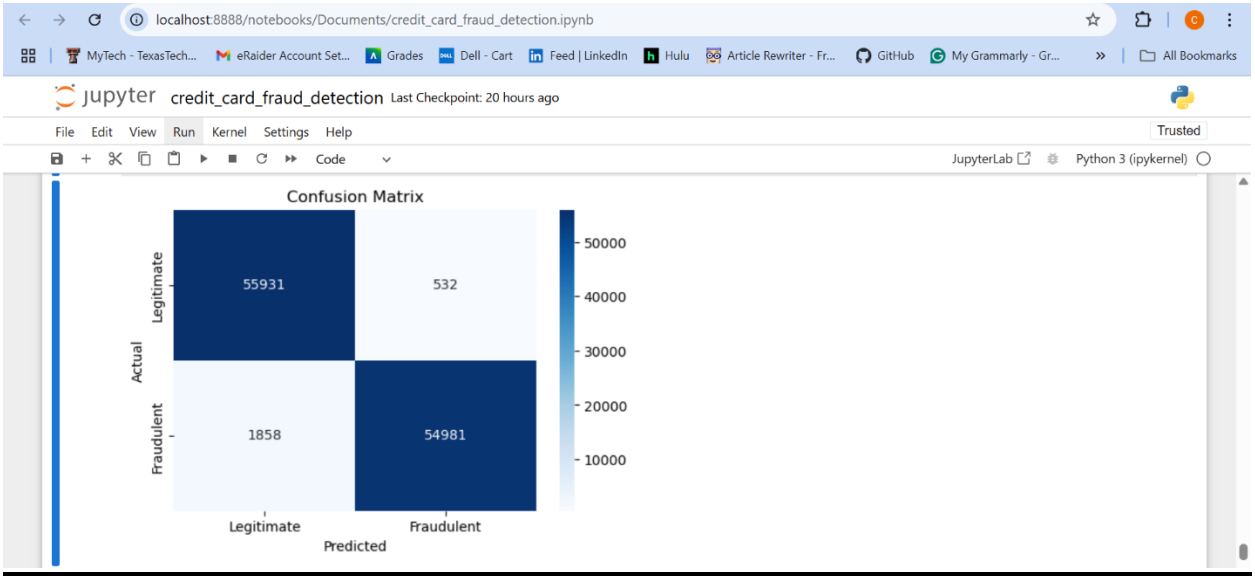
```
[63]: ##corelation matrix in balanced data
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming you've already resampled your dataset using SMOTE and have X_smote and y_smote

# Combine the resampled data into a single DataFrame
balanced_data = X_smote.copy()
balanced_data['class'] = y_smote

# Calculate the correlation matrix for the balanced dataset
corr_balanced = balanced_data.corr()

# Plot the correlation matrix
plt.figure(figsize=(8, 6))  # Set the figure size
sns.heatmap(corr_balanced, annot=False, cmap="plasma", linewidths=0.5)  # Plot the heatmap
plt.title("Correlation Matrix for Balanced Data")  # Set the title
plt.show()  # Display the plot
```

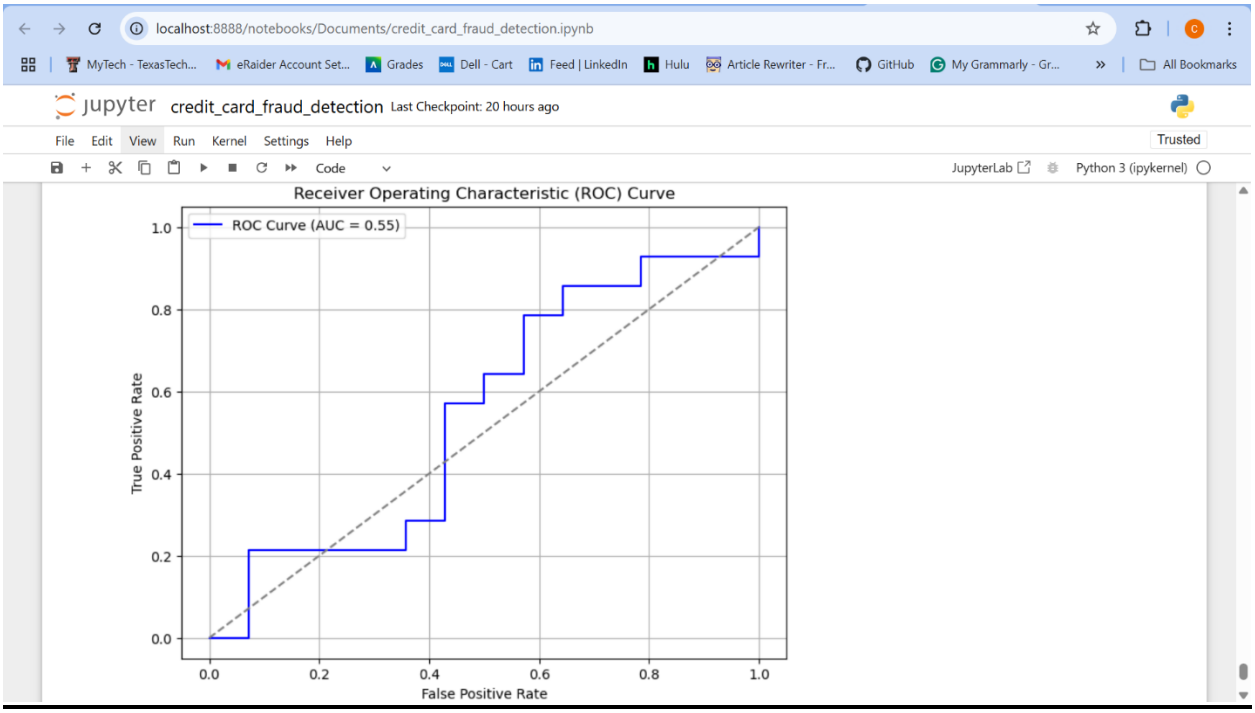
4.3 Classification Report:-

```
Accuracy: 97.89%

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     56463
           1       0.99      0.97      0.98     56839

    accuracy                           0.98    113302
   macro avg       0.98      0.98      0.98    113302
weighted avg       0.98      0.98      0.98    113302


Confusion Matrix:
[[55931   532]
 [ 1858 54981]]
```

4.4 Confusion Matrix:-

```
[71]: ##Confusion Matrix
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming you have y_test as the true labels and y_pred as the predicted labels
cm = confusion_matrix(y_test, y_pred)

# Plotting the confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Legitimate', 'Fraudulent'], yticklabels=['Legitimate', 'Fraudulent'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```





Conclusion

- **Achievements:**
  - Successfully built a model to detect fraudulent transactions with high precision.
  - Resolved class imbalance using SMOTE.
  - XGBoost emerged as the best-performing model.
- **Key Takeaways:**
  - Data balancing is critical for fraud detection.
  - Ensemble models like XGBoost are powerful in handling real-world fraud data.
- **Limitations:**
  - Dataset is anonymized, limiting feature explainability.
  - Doesn't account for evolving fraud patterns over time (static data).

Future Scope

- **Improvements**:
  - Incorporate real-time data streaming for live fraud detection.
  - Include more contextual features like location, merchant info, device data.
- **Additions**:
  - Deploy as a **REST API** using Flask/Fast API.
  - Use **deep learning models** (e.g., Autoencoders, LSTM) for anomaly detection.
  - Integrate feedback loop for continuous learning.

References

Follow APA style:

1. Dal Pozzolo, A., Caelen, O., Le Borgne, Y. A., Waterschoot, S., & Bontempi, G. (2015). *Calibrating Probability with Undersampling for Unbalanced Classification*. In **2015 IEEE Symposium Series on Computational Intelligence (SSCI)** (pp. 159–166). IEEE. https://doi.org/10.1109/SSCI.2015.33
2. Kaggle. (n.d.). *Credit Card Fraud Detection*. Retrieved from https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
3. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). *Scikit-learn: Machine Learning in Python*. **Journal of Machine Learning Research**, **12**, 2825-2830. http://jmlr.org/papers/v12/pedregosa11a.html
4. Chen, T., & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System*. In **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining** (pp. 785–794). https://doi.org/10.1145/2939672.2939785
5. He, H., & Garcia, E. A. (2009). *Learning from imbalanced data*. **IEEE Transactions on Knowledge and Data Engineering**, **21(9)**, 1263–1284. https://doi.org/10.1109/TKDE.2008.239