BAMM 1043

CREDIT CARD DATA

GROUP MEMBERS
Sushmeet Singh Nandra- C0894235
Tolulope Abidoye- C0889074
Muhammad Yaasir
Goolam Dustageer - C0906653
Nakul Deshwal - C0895506
Nirajan Khadka- C0900308
Bivek Yadav-C0906205
Vivian Ekwegh -C0891071

INTRODUCTION TO PYTHON FOR DATA ANALYSIS

Python is a computer language that has gained widespread recognition due to its clear and simple syntax, which prioritizes the readability of code. This particular characteristic greatly facilitates programmers in effectively and efficiently articulating their concepts. The language has notable adaptability, as it is capable of tolerating several programming paradigms, including procedural, object-oriented, and functional. This flexibility empowers developers to select the most suitable technique for their own projects. Moreover, Python's expansive standard library, along with the vast array of third-party libraries and tools available on the Python Package Index (PyPI), enables developers to seamlessly incorporate robust functionality into their applications. Additionally, the cross-platform portability of Python, together with its robust community support, plays a significant role in its ongoing advancement, rendering it a preferred option for a wide range of applications. The software's ease of use has contributed to its widespread adoption as an instructional tool for programming education.

Here are a few Python concepts:

1. ***Data Import and Preparation:***

Python can handle various data sources, including CSV files, Excel spreadsheets, JSON files, SQL databases, and web APIs. Libraries like pandas allow you to read, clean, and transform data efficiently, performing tasks like filtering, merging, and reshaping.

2. ***Data Analysis and Modeling:***

Pandas provides a DataFrame structure, which is a two-dimensional table-like data structure. You can perform descriptive statistics, aggregation, grouping, and more. For advanced analysis, libraries like NumPy offer support for numerical operations and array manipulation. Additionally, scikit-learn provides machine learning tools for building models.

3. ***Data Visualization:***

Matplotlib and Seaborn are powerful libraries for creating a wide range of static and interactive visualizations, such as line charts, bar plots, scatter plots, and heatmaps. These libraries help you present insights visually and effectively.

4. ***Jupyter Notebooks:***

Jupyter Notebooks provide an interactive environment where you can combine code, visualizations, and explanatory text. This enables you to create interactive data reports and share your analyses with others.

5. ***Statistical Analysis and Hypothesis Testing:***

Statsmodels is a library that offers tools for conducting statistical tests, estimating models, and performing hypothesis testing. You can analyze relationships, trends, and correlations within your data.

6. ***Data Visualization Libraries:***

Apart from Matplotlib and Seaborn, Plotly is another library that allows you to create interactive, web-based visualizations. It's suitable for building dashboards and interactive data exploration tools.

7. ***Data Wrangling and Transformation:***

Python's pandas library provides comprehensive support for data cleaning, transformation, and manipulation. You can reshape data, handle missing values, and apply custom functions efficiently.

8. ***Web Scraping and APIs:***

Python has libraries like Beautiful Soup and Requests that enable web scraping to extract data from websites. Additionally, you can interact with APIs to fetch data directly from online services.

9. ***Integration and Deployment:***

Python can be integrated into web applications, data pipelines, and more. Flask and Django are popular web frameworks for integrating Python-powered data analysis into web applications. You can also deploy models and analyses using cloud platforms like AWS and Heroku.

In summary, Python offers a comprehensive toolkit for data analysis, including data import, manipulation, modeling, visualization, and integration. It empowers data analysts and scientists to explore data, derive insights, and make informed decisions.

Project Review

Credit card fraud is a significant concern in today's digital economy, posing substantial financial losses to individuals, businesses, and financial institutions. As electronic payment methods become increasingly prevalent, so too does the need for robust strategies to detect and prevent fraudulent activities. In this project, we delve into the intricate landscape of credit card fraud, exploring how advanced technologies and data-driven approaches play a pivotal role in safeguarding financial transactions. By harnessing the power of machine learning, data mining, and anomaly detection, financial institutions can proactively identify suspicious activities and swiftly respond to potential threats. Throughout this study, we will examine the methodologies, tools, and real-world applications that underpin credit card fraud analysis, highlighting its critical role in ensuring the security and integrity of electronic payment systems. Credit card fraud analysis encompasses a range of techniques and methodologies aimed at identifying patterns, anomalies, and trends in financial transactions to mitigate the risks associated with fraudulent behavior. The main aim of our project is to extract valuable insights from this dataset to elevate credit card fraud.

Introduction

A credit card is the most common method of payment. A credit card is a compact, thin plastic or fiber card that carries information about the person, such as a picture or service to his linked account charges, which are debited on a regular basis. ATMs, swiping machines, retail readers, and bank and online transactions now read card information. Each card has a unique card number, which is very significant; the security of the card is mostly dependent on the physical security of the card as well as the privacy of the credit card number.

Credit card fraud, in general, refers to the unauthorized use of credit or debit cards to make payments. It is considered illegal to attempt to use physical card information without the approval and knowledge of the cardholder. Credit card fraud is classified into two types: online fraud, which may be detected via mobile phones, the internet, the web, and shopping, and offline fraud, which identifies when the card is stolen by utilizing personal information. One of the criminal acts that might occur during an online purchase is credit card fraud. These are basically fraudulent sources of funds that are used in various transactions.

According to the Federal Trade Commission (2020, 2021). Credit card fraud increased 44.7% in the United States from 2019 to 2020. Credit card fraud is the second most common type of identity theft recorded. With the introduction of new technologies that cause massive financial losses, fraud is increasing at an alarming rate.

This is an increasingly prevalent subject that requires the attention of the machine learning and data science fields, and the solution can be automated. This issue is particularly difficult from the standpoint of learning because it is characterized by many aspects such as class imbalance. Banking fraud can't be completely eradicated but at least we can prevent it from happening and its occurrence to a certain level using machine learning techniques.

Dataset

Our dataset is from Kaggle, and the dataset is available from

https://www.kaggle.com/mlg-ulb/creditcardfraud.

The data set is comprised of CSV-formatted transactions from a European bank in September 2013. There are 492 (0.17%) fraudulent transactions and 284,807 (99.83%) real transactions in it. Due to privacy and confidentiality, not much background information is provided, and only PCA-transformed data is given. Except for time and amount, all other supplied values v1, v2, v3,... v28 are PCA transformed numeric values. The dataset contains 31 numerical features, with V1, V2,... V28 derived through Principal Component Analysis (PCA). "Time" is one of the attributes that represents the time between the first transaction and subsequent transactions.

"Amount" represents the total amount of credit card transactions. The "Class" column is the target column, which contains only 1 for fraudulent transactions and 0 for genuine transactions. As a result, binary classification should be used to treat the dataset.

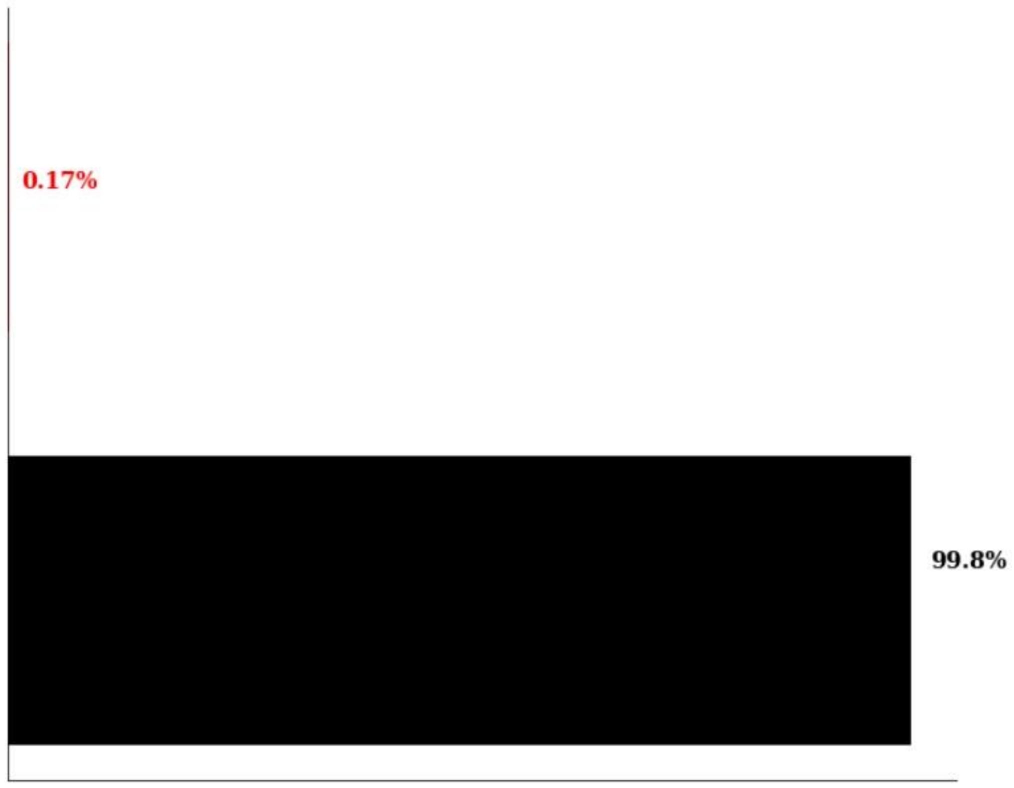


Fig-1: Showing Imbalance of data in percentage

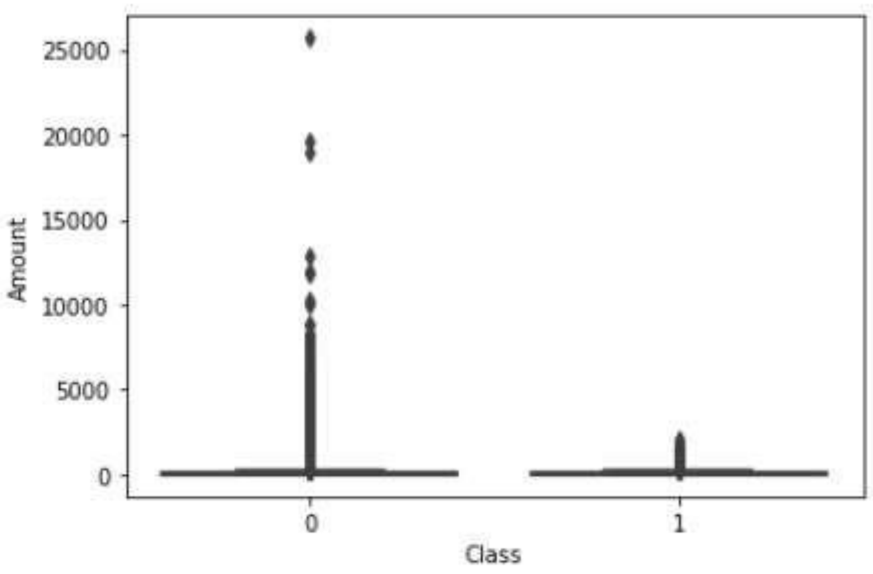
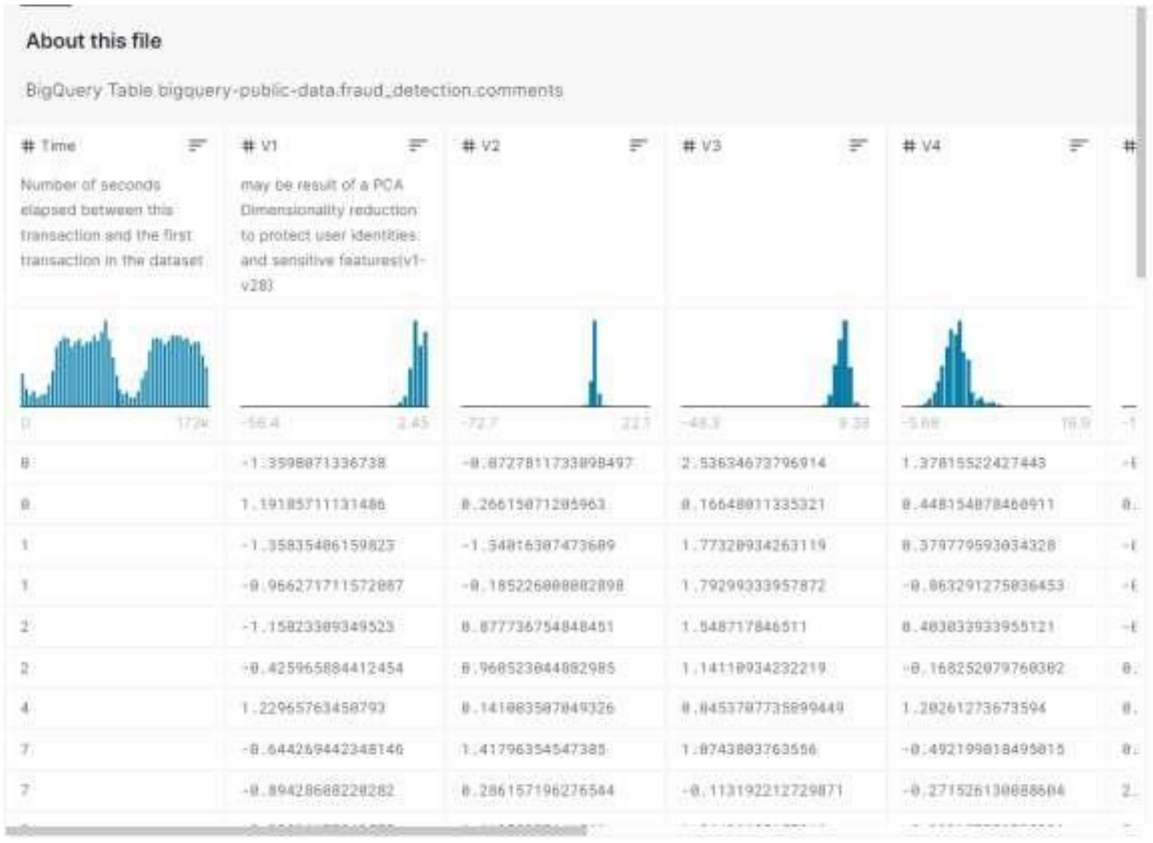


Fig-2: Showing Imbalance of class with respect with Amount



About this file

BigQuery Table bigquery-public-data.fraud_detection.comments

# Time	# V1	# V2	# V3	# V4
Number of seconds elapsed between this transaction and the first transaction in the dataset	may be result of a PCA Dimensionality reduction to protect user identities and sensitive features(v1-v28)			
0	-1.3598071336738	-0.0727811733098497	2.53634673796914	1.37815522427443
0	1.19185711131486	0.26615071205963	0.16648011335321	0.448154078460911
1	-1.35835406159823	-1.34016307473609	1.77320934263119	0.379779593034328
1	-0.966271711572087	-0.185226008082898	1.79299333957872	-0.863291275036453
2	-1.15823309349523	0.877736754848451	1.548717846511	0.403033933955121
2	-0.425965884412454	0.960523044882985	1.14110934232219	-0.168252079760302
4	1.22965763450793	0.141003507049326	0.0453707735899449	1.20261273673594
7	-0.644269442348146	1.41796354547385	1.0743803763556	-0.492199018495015
7	-0.89428608220282	0.286157196276544	-0.113192212729871	-0.271526130088604

Fig-3: Showing dataset page from Kaggle

AGENDA

• Identifying Factors Leading to Transaction Abandonment

• Extracting Customer-Suggested Enhancement Areas

• Enhancing Overall Credit Card Transaction Experience for Customers

Primary Challenges Addressed through Analysis

- Transaction Behavior Analysis: Python's capabilities empower comprehensive analysis of credit card transaction data. Interactive visualizations facilitate the identification of trends, comparison of transaction performance, and prediction of future fraudulent activities based on historical patterns.
- Fraud Detection Optimization: Python helps identify unusual patterns in transaction data, aiding in the detection of anomalies and potential fraud cases. Visualizations highlight irregular activities that might evade traditional detection methods, enhancing Amazon's fraud detection strategies.
- User Segment Insights: Python facilitates the segmentation of credit card users based on demographics and transaction behaviors. This segmentation informs targeted fraud prevention measures and tailored communication strategies.
- Effectiveness of Fraud Alerts: The impact of personalized alerts on transaction behavior can be visualized through Python. The analysis reveals the efficacy of alerts in preventing fraudulent transactions.
- Fraud Detection Impact: Python assesses the effectiveness of fraud detection mechanisms by visualizing the correlation between detected fraud cases and various transaction attributes.
- Fraudulent Pattern Identification: Through Python, financial institutions can access patterns associated with fraud. Visualizing data enables the identification of characteristics common among fraudulent transactions.

Tools and modules used

- Python
- Seaborn
- Matplotlib.pyplot
- Scikit-learn
- Pandas
- Nump

Algorithms Used

- Supervise learning methods

Logistics Regression is less prone to overfitting, although it can overfit in large datasets. To avoid overfitting, we can investigate regularisation approaches. Any significant outliers will be translated into a value between 0 and 1. It primarily aids with classification problems and provides us with information about whether or not an event is occurring.

RandomForest Classifier: A random forest is a meta estimator that fits a number of decision tree classifiers on different sub-samples of the dataset and utilizes averaging to increase predicted accuracy and control over-fitting.

Tree algorithms, which are used in ranking, classification, and a variety of other machine-learning applications

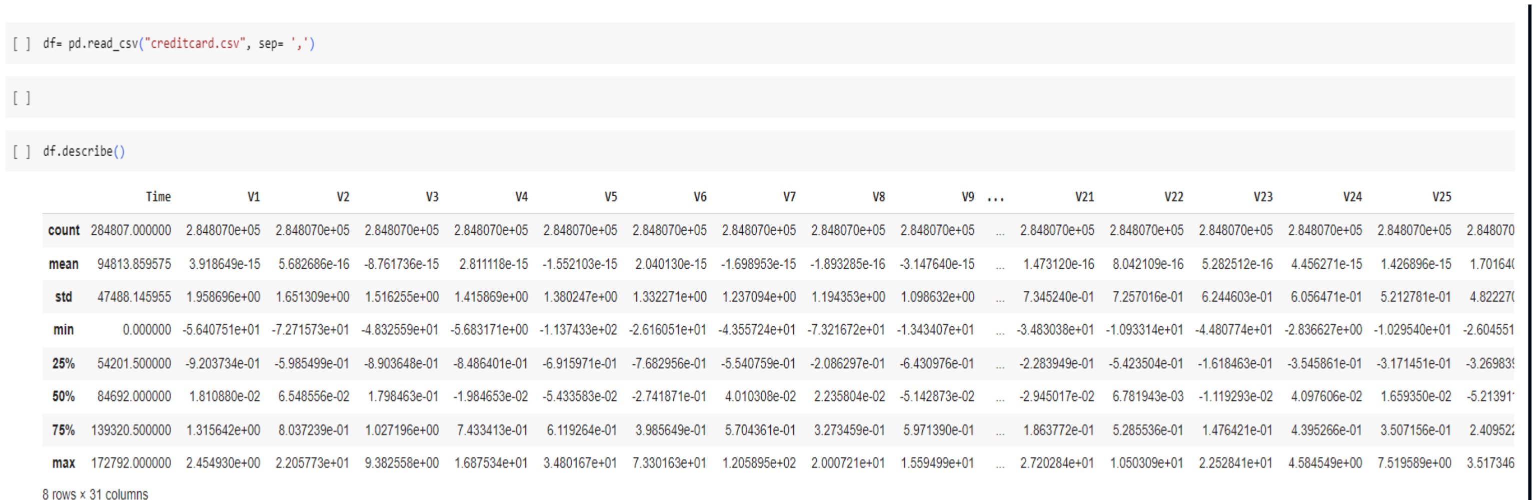
KNN: It is a machine learning algorithm that is supervised. The algorithm can handle classification and regression problem statements. The symbol 'K' represents the number of nearest neighbours to a new unknown variable that must be predicted or categorised.

GaussianNB: This categorization approach learns the probability of an object being linked with a specific category or class based on a specific attribute. When used to large amounts of data, the naive Bayes technique can fit models quickly and accurately while requiring little training.

Isolation Forest: The Isolation Forest algorithm is a powerful anomaly detection technique that operates based on the principles of isolating anomalies rather than profiling normal instances

Data Preprocessing:

- The code imports necessary libraries such as Pandas, NumPy, matplotlib, Seaborn, Warnings, and various machine learning algorithms.
- It reads a CSV file named "creditcard.csv" and loads the data into a panda DataFrame.
- The dataset's summary statistics, class distribution, and missing value information are displayed.



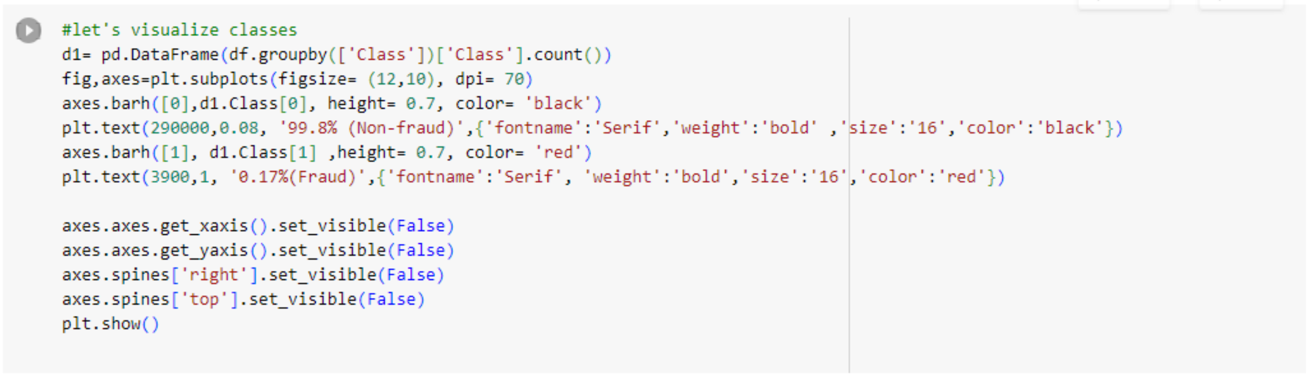
```
[ ] df= pd.read_csv("creditcard.csv", sep= ',')

[ ]

[ ] df.describe()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	...	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070
mean	94813.859575	3.918649e-15	5.682686e-16	-8.761736e-15	2.811118e-15	-1.552103e-15	2.040130e-15	-1.698953e-15	-1.893285e-16	-3.147640e-15	...	1.473120e-16	8.042109e-16	5.282512e-16	4.456271e-15	1.426896e-15	1.70164
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	...	7.345240e-01	7.257016e-01	6.244603e-01	6.056471e-01	5.212781e-01	4.82227
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	...	-3.483038e+01	-1.093314e+01	-4.480774e+01	-2.836627e+00	-1.029540e+01	-2.604551
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	...	-2.283949e-01	-5.423504e-01	-1.618463e-01	-3.545861e-01	-3.171451e-01	-3.26983
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02	...	-2.945017e-02	6.781943e-03	-1.119293e-02	4.097606e-02	1.659350e-02	-5.21391
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01	...	1.863772e-01	5.285536e-01	1.476421e-01	4.395266e-01	3.507156e-01	2.40952
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01	...	2.720284e+01	1.050309e+01	2.252841e+01	4.584549e+00	7.519589e+00	3.517346

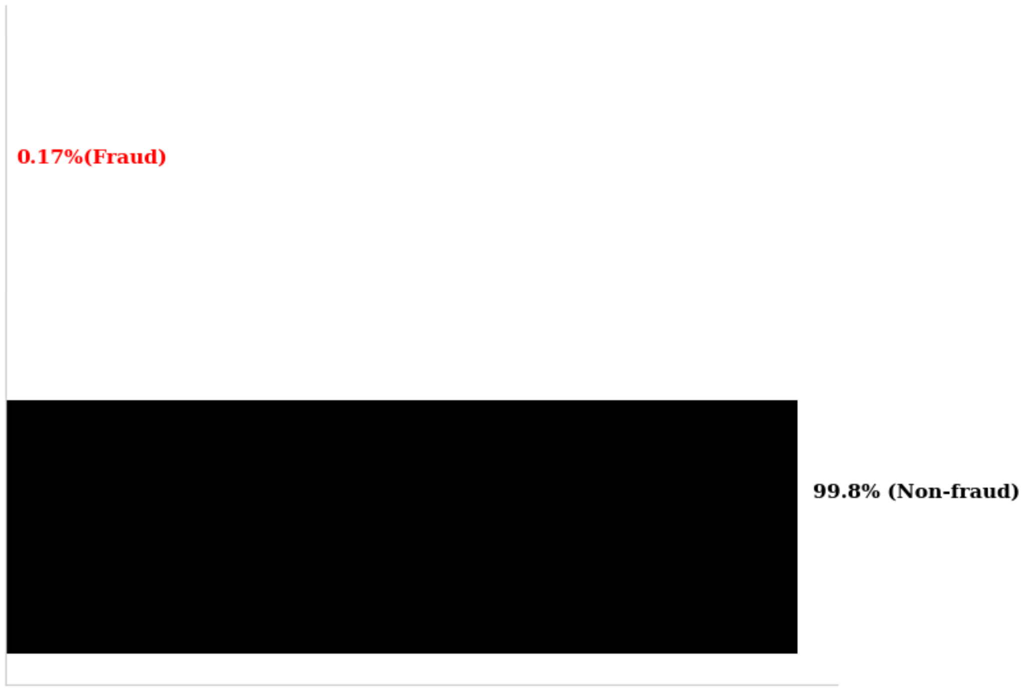
8 rows × 31 columns

Data Visualization:

- Visualizations are created using seaborn and matplotlib to illustrate aspects such as class distribution, time distribution, transaction amount distribution, and scatter plots.
- Box plots and scatter plots are used to examine the relationship between transaction time, amount, and class (fraud or not).

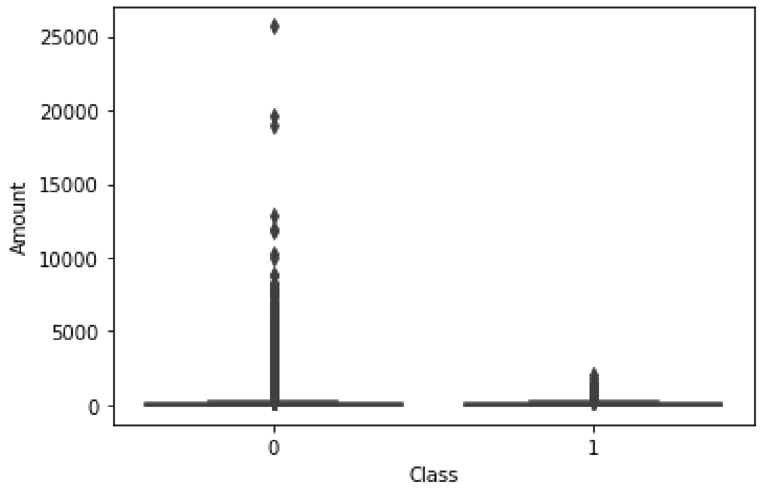


```
#let's visualize classes
d1= pd.DataFrame(df.groupby(['Class'])['Class'].count())
fig,axes=plt.subplots(figsize= (12,10), dpi= 70)
axes.barh([0],d1.Class[0], height= 0.7, color= 'black')
plt.text(290000,0.08, '99.8% (Non-fraud)',{'fontname':'Serif','weight':'bold' ,'size':'16','color':'black'})
axes.barh([1], d1.Class[1] ,height= 0.7, color= 'red')
plt.text(3900,1, '0.17%(Fraud)',{'fontname':'Serif', 'weight':'bold','size':'16','color':'red'})

axes.axes.get_xaxis().set_visible(False)
axes.axes.get_yaxis().set_visible(False)
axes.spines['right'].set_visible(False)
axes.spines['top'].set_visible(False)
plt.show()
```



```
sns.boxplot(x=df['Class'], y=df['Amount'])
```

```
<AxesSubplot:xlabel='Class', ylabel='Amount'>
```



Data Scaling:

- The RobustScaler from sklearn is applied to scale the 'Amount' feature due to the presence of outliers.

```
[ ]  #becuasue of outliers let;s scale it
     from sklearn.preprocessing import RobustScaler

     rbst= RobustScaler()
```

```
[ ]  df['Amount']= rbst.fit_transform(df['Amount'].values.reshape(-1,1))
```

Model Building and Evaluation:

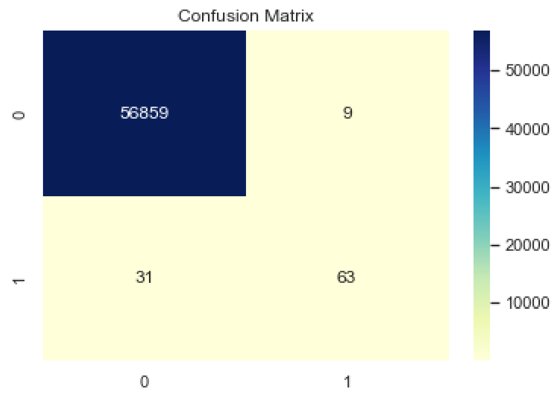
- The dataset is split into training and testing sets.
- Three different machine learning models are trained and evaluated: Logistic Regression, K-Nearest Neighbors (KNN), and Gaussian Naive Bayes.
- The precision, recall, accuracy, classification report, and confusion matrix are calculated for each model.

```
X_train,X_test,y_train,y_test= train_test_split(X,Y,test_size= 0.2, random_state= 10)
```

```
# a fucntion to see things easi;ly
def predict(model,X_train, X_test, y_train, y_test):
    model.fit(X_train,y_train)
    preds=model.predict(X_test)
    print(confusion_matrix(y_test,preds))
    print(classification_report(y_test,preds))
    print("roc_score",roc_auc_score(y_test, preds))
    print("Precision :",metrics.precision_score(y_test, preds))
    print("Recall :",metrics.recall_score(y_test, preds))
    plt.plot(figsize=(27,12))

    plt.title("Confusion Matrix")
    sns.heatmap(confusion_matrix(y_test,preds),annot=True,fmt='0.0f',cmap="YlGnBu")


    plt.show()


    return accuracy_score(y_test,preds)
```

```
[ ] predict(LogisticRegression(), X_train, X_test, y_train, y_test)
```

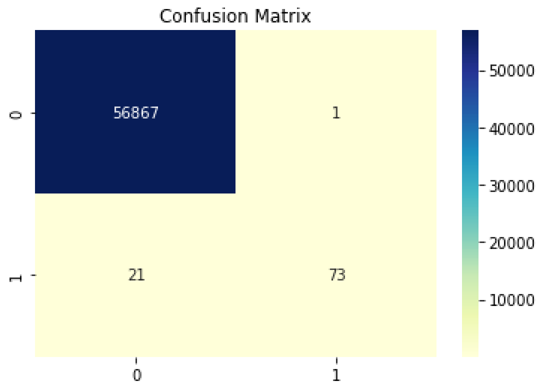
```
[[56859     9]
 [   31    63]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56868
           1       0.88      0.67      0.76        94

    accuracy                           1.00     56962
   macro avg       0.94      0.84      0.88     56962
weighted avg       1.00      1.00      1.00     56962

roc_score 0.8350272523604495
Precision : 0.875
Recall : 0.6702127659574468
```



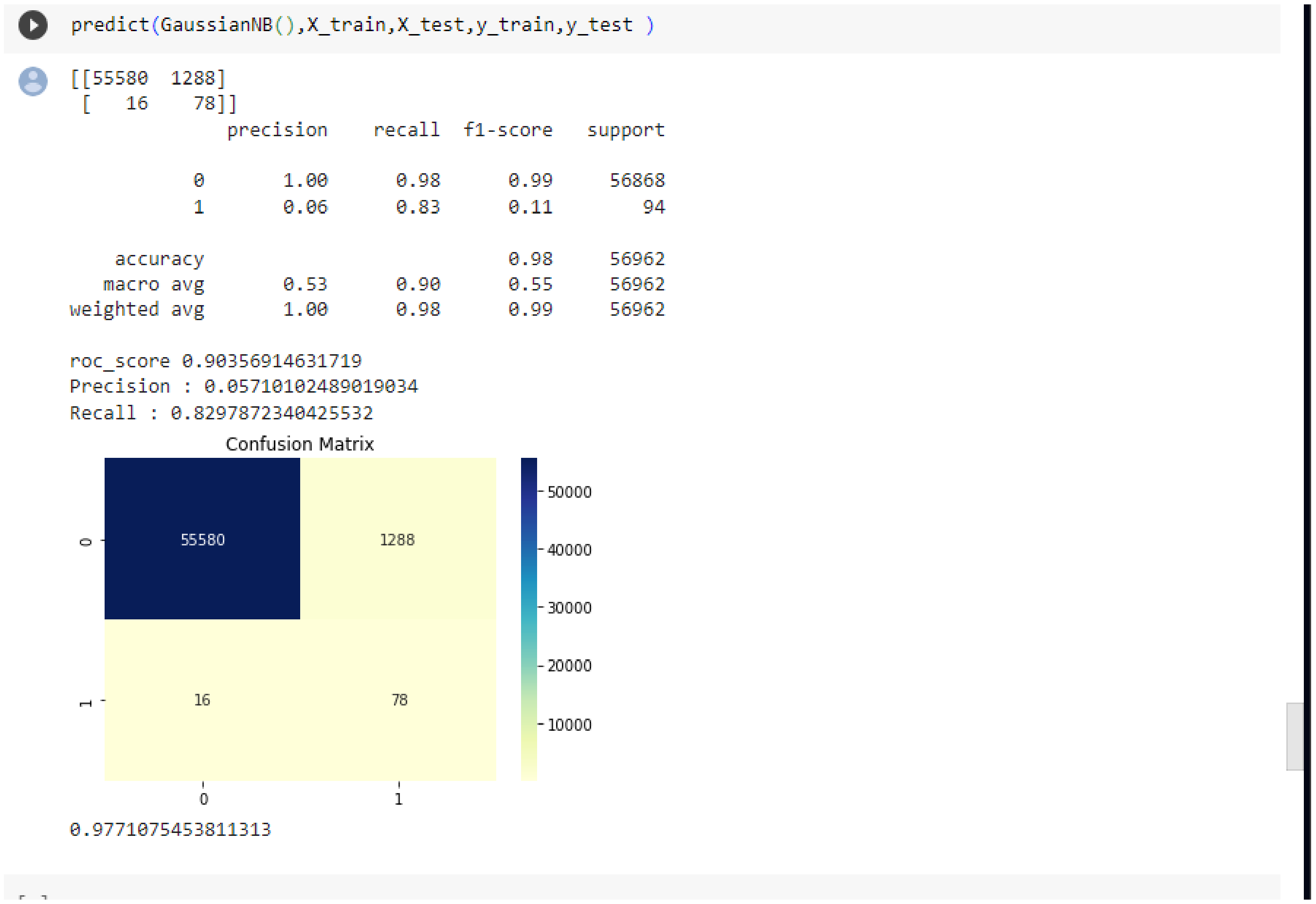
```
0.9992977774656788
```

```
[ ]  predict(Knn(n_neighbors=5),X_train,X_test,y_train,y_test)
```

```
[[56867     1]
 [   21    73]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56868
           1       0.99      0.78      0.87        94

    accuracy                           1.00     56962
   macro avg       0.99      0.89      0.93     56962
weighted avg       1.00      1.00      1.00     56962

roc_score 0.8882890800495062
Precision : 0.9864864864864865
Recall : 0.776595744680851
```

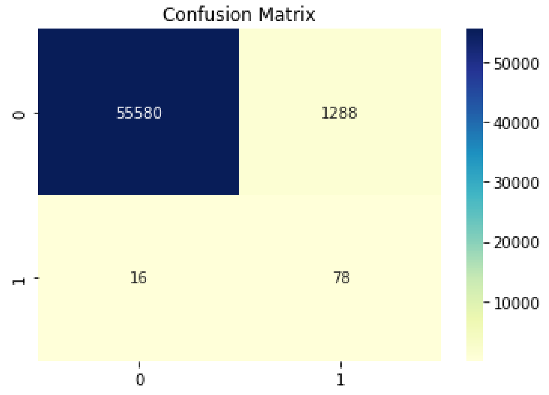


```
0.9996137776061234
```



```
predict(GaussianNB(),X_train,X_test,y_train,y_test )
```

```
[[55580  1288]
 [   16    78]]
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     56868
           1       0.06      0.83      0.11        94

    accuracy                           0.98     56962
   macro avg       0.53      0.90      0.55     56962
weighted avg       1.00      0.98      0.99     56962

roc_score 0.90356914631719
Precision : 0.05710102489019034
Recall : 0.8297872340425532
```

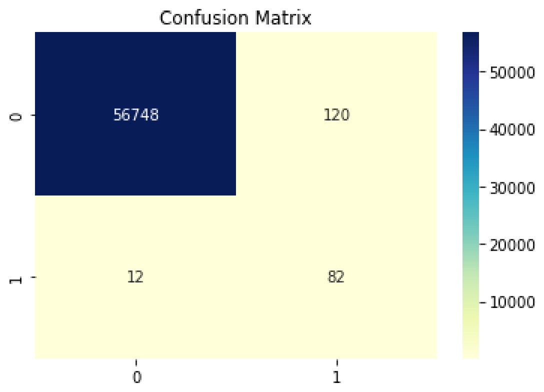


```
0.9771075453811313
```

After Sampling

```
[ ]  predict(Knn(n_neighbors=5),X_smote,X_test,Y_smote,y_test)
```

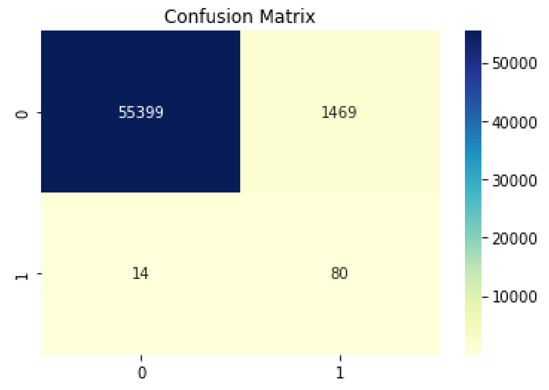
```
[[56748   120]
 [   12    82]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56868
           1       0.41      0.87      0.55        94

    accuracy                           1.00     56962
   macro avg       0.70      0.94      0.78     56962
weighted avg       1.00      1.00      1.00     56962

roc_score 0.9351151378556388
Precision : 0.40594059405940597
Recall : 0.8723404255319149
```



```
0.9976826656367402
```

```
predict(GaussianNB(),X_smote,X_test,Y_smote,y_test )
```

```
[[55399  1469]
 [   14    80]]
              precision    recall  f1-score   support

           0       1.00      0.97      0.99     56868
           1       0.05      0.85      0.10        94

    accuracy                           0.97     56962
   macro avg       0.53      0.91      0.54     56962
weighted avg       1.00      0.97      0.99     56962

roc_score 0.9126160395331331
Precision : 0.051646223369916075
Recall : 0.851063829787234
```



```
0.9739650995400443
```

Balancing Data with SMOTE:

- The Synthetic Minority Over-sampling Technique (SMOTE) is applied to balance the training dataset.
- The same three models are trained and evaluated again using the balanced training set.

now le'ts try same after sampling

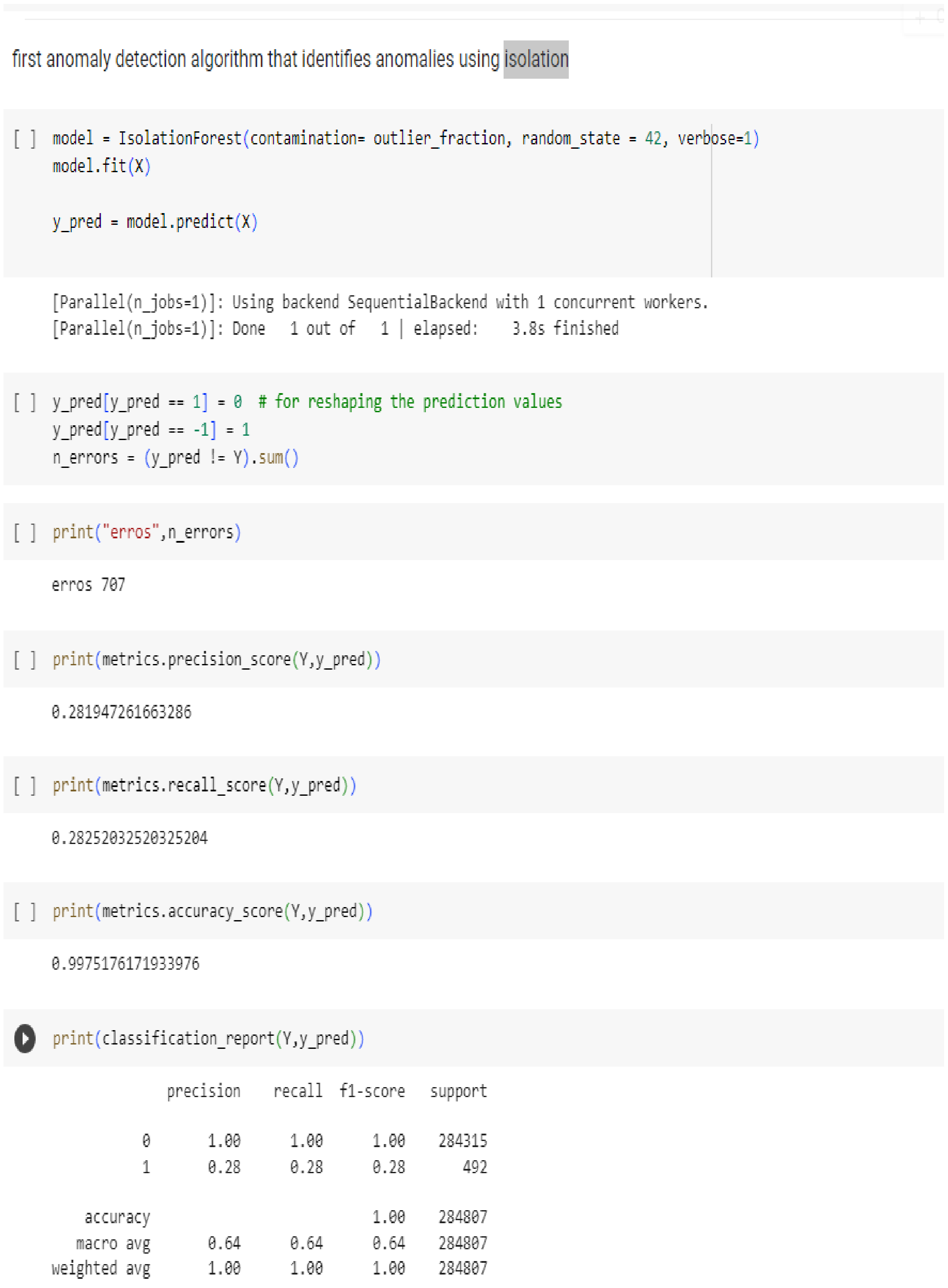
```
[ ]
```

```
[ ] smt = SMOTE(random_state=42)

    X_smote,Y_smote = smt.fit_resample(X_train,y_train)
```

```
[ ]
```

Anomaly Detection with Isolation Forest:

- An Isolation Forest model is used for anomaly detection, aimed at identifying outliers or anomalies in the data.
- The model's predictions are compared to the ground truth class labels to evaluate its performance in detecting fraud cases.



first anomaly detection algorithm that identifies anomalies using isolation

```
[ ] model = IsolationForest(contamination= outlier_fraction, random_state = 42, verbose=1)
    model.fit(X)

    y_pred = model.predict(X)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    3.8s finished
```

```
[ ] y_pred[y_pred == 1] = 0  # for reshaping the prediction values
    y_pred[y_pred == -1] = 1
    n_errors = (y_pred != Y).sum()
```

```
[ ] print("erros",n_errors)
```

```
erros 707
```

```
[ ] print(metrics.precision_score(Y,y_pred))
```

```
0.281947261663286
```

```
[ ] print(metrics.recall_score(Y,y_pred))
```

```
0.28252032520325204
```

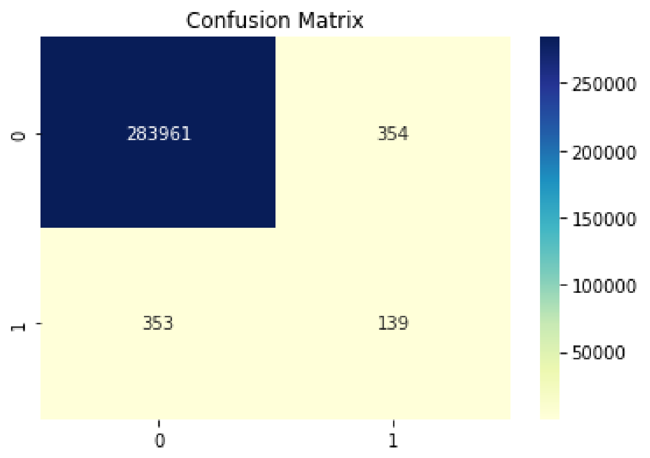
```
[ ] print(metrics.accuracy_score(Y,y_pred))
```

```
0.9975176171933976
```

```
print(classification_report(Y,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    284315
           1       0.28      0.28      0.28       492

    accuracy                           1.00    284807
   macro avg       0.64      0.64      0.64    284807
weighted avg       1.00      1.00      1.00    284807
```

```
[ ]  plt.plot(figsize=(27,12))

     plt.title("Confusion Matrix")
     sns.heatmap(confusion_matrix(Y,y_pred),annot=True,fmt='0.0f',cmap="YlGnBu")


     plt.show()
```



```
[ ]  print("roc_score",roc_auc_score(Y,y_pred))
```

```
roc_score 0.6406376136682246
```

Model Evaluation and Comparison:

- The precision, recall, accuracy, confusion matrix, and ROC AUC score are displayed for the Isolation Forest anomaly detection model.

Concluding Remarks:

The code concludes by mentioning that machine learning models seem effective for credit card fraud detection, but the accuracy may decrease when using balanced data.

Result and Experiment analysis

To compare different machine learning and deep learning models, a comparison table was created. We calculated precision, recall, ROC curve, and accuracy using Sklearn's built-in models such as accuracy and AUC_Score.
Accuracy: This is the fraction of total transactions that have been accurately identified (fraudulent and nonfraudulent).

Precision: Precision represents the model's precision/accuracy. Precision is a good metric to use when the cost of false positives is substantial.

Recall: Recall is the number of Actual Positives that our model catches by labelling them as Positive (True Positive). When there is a substantial cost associated with False Negative, it should be the model metric that we employ.

The term ROC curve refers to the Receiver Operating Characteristic (ROC) curve. The true positive rate (Sensitivity) is represented as a function of the false positive rate (100Specificity) for various cutoff points.The overall accuracy of the test will be high if the ROC curve is closer to the upper left corner. The area under the ROC curve (AUC) is a parameter that can differentiate between two groups (fraudulent/nonfraudulent).

Results comparison

The performance of the algorithm must be closely compared with other algorithms to classify between fraudulent and non-fraudulent data. Comparison with standard classification algorithms like Logistic Regression, Naïve Bayes and KNN has been done.

Table 1 : Comparison of Algorithms

Algorithm	Accuracy	Precision	Recall	Auc_ROC_Score
Logistic Regression(Imbalanced)	0.999	0.875	0.670	0.835
Logistic Regression(balanced)	0.973	0.053	0.904	0.938
Naïve bayes(imbalanced)	0.977	0.057	0.829	0.903
Naïve bayes(balanced)	0.973	0.051	0.851	0.912
KNN(imbalanced)	0.999	0.986	0.776	0.888
KNN(balanced)	0.997	0.405	0.872	0.935
RandomForest	0	0.285	0.282	0.640

Conclusion and Future Scope

Credit card fraud detection methods have gained popularity in the past decade with the evolution of statistical models, machine learning algorithms, and data mining techniques. The fraud transaction prediction has 2 phases which are feature extraction and classification. Within the first phase, the feature extraction technique is applied and within the second phase, classification is applied for fraud transaction detection, Fraud transaction detection is the major issue of prediction because of the frequent and enormous number of transactions. During this comparative research study, we tried to analyze the dataset through various graphs and also tried to detect fraud using some classification algorithms and make a comparative analysis.

As we can see from the data above simple algorithm like Logistic Regression performs best compared to other algorithms. And even after balancing the imbalance between classes performance of various algorithms only improves by a little margin.

The more surprising thing is that Logistic Regression takes less time to train compared to other algorithms.

Also, even when trained on Non-fraud datasets only, anomaly detection techniques like RandomForest can detect anomalies.
In the future Deep learning models like LSTM, BiLSTM, And unsupervised learning methods like clustering or even a combination of both could be used but more importantly, a balanced dataset could be used to properly measure their performance.

References

1. Aditya Saini, Swarna Deep Sarkar, Shadab Ahmed, SP Maniraj " Credit Card Fraud Detection using Machine Learning and Data Science", International Journal of Engineering Research & Technology, ISSN: 2278-0181, Vol. 8 Issue 09, September-2019

2. https://www.kaggle.com/mlg-ulb/creditcardfraud

   report link https://github.com/shaaguunz/college_project_2/tree/main/INT248