IE406 Machine Learning

Project Name: Credit Card Fraud Detection

Group 10

201801140: Sudarshan Kundnani
201801151: Dhruv Chavda
201801225: Chirag Patel
201801229: Sanket Mistry
201801462: Darshil Rana

Introduction

In credit card transactions, **'Fraud'** is considered as an unauthorized and unwanted usage of an account by someone other than the owner of that account. Necessary prevention measures can be taken to stop such fraudulent practices and it becomes important to identify which transactions are fraudulent.

Different **'Fraud detection'** can be used which involves monitoring the activities of populations of users in order to estimate, perceive or avoid objectionable behavior, which consist of fraud, intrusion, and sometimes, even defaulting. We can approach this problem with the help of Data-structures and Machine-learning , with the help of which this problem can be automated.

One of the major issues while handling the data is that the number of valid transactions far outnumber fraudulent ones.

Machine learning algorithms can be used to analyze all the authorized transactions and report the suspicious ones. These reports are investigated by professionals who contact the cardholders to confirm if the transaction was genuine or fraudulent

Motivation and Objective

As we are moving towards the digital world — Cyber security is becoming a crucial part of our life. When we talk about security in digital life then the main challenge is to find the abnormal activity.

When we make any transaction while purchasing any product online — a good amount of people prefer credit cards. The credit limit in credit cards sometimes helps us me making purchases even if we don't have the amount at that time. but, on the other hand, these features are misused by Cyber attackers. One of the most common way these attackers exploit the system is doing a fraudulent transaction.

To tackle this problem we need a system that can detect anomaly in the transaction if it finds fishy.

Here, comes the need for a system that can track the pattern of all the transactions and if any pattern is abnormal then the transaction should be aborted.

Today, we have many machine learning algorithms that can help us classify abnormal (fraudulent) transactions. The only requirement is the past data and the suitable algorithm that can fit our data in a better form. The algorithm can use past model to train itself to detect fraudulent transactions and can be used to detect anomaly in present-day transactions

Methodology

The approach that this paper proposes, uses the different machine learning algorithms to detect anomalous activities, primarily the fraudulent transactions.

The basic rough architecture diagram can be represented with the following figure:

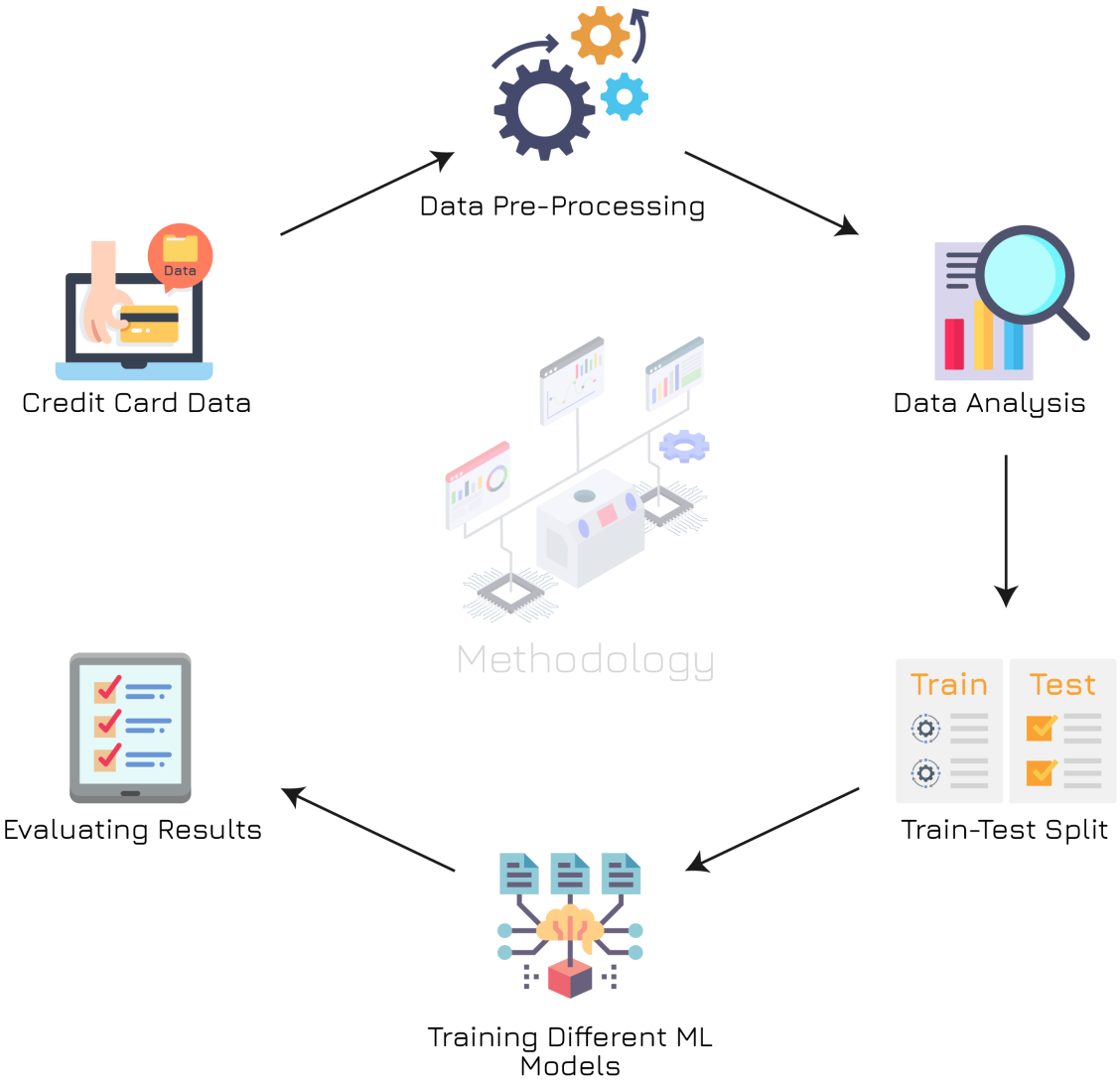

Figure 1: Schematic Diagram of methodology

We have approached this problem by dividing it into six fundamental steps:

- **Credit-Card Data:** We collected data of around 300,000 transactions. These transactions will have some fraudulent transactions as well (around 500)
- **Data Analysis and Pre-Prepossessing:** In this step, we try to analyze the data and find important features that will affect results in our model and remove the unnecessary ones. Here, we will also standardize all the features of the model.
- **Train-Test Split:** The total collected data will be divided into 3:1 train-test data. It means that 75% of the total data will be used to train the ML model and the remaining 25% will be used to test the accuracy of the of the model.
- **Training Different ML Models:** We have applied 4 different Machine Learning Algorithms and trained all of them with the same data. These 4 algorithms are:

  1. - Logistic Regression
  2. - Support Vector Classification (SVM)
  3. - Random Forest Classifier
  4. - Decision Tree Classifier

- **Evaluating Results:** In this step, we will test all the four models with the remaining testing data and try to find model will provide us most accurate results

Data-Set

First of all, we downloaded the data-set from Kaggle[1]. The data-set contains data of about 284,807 transactions and each transaction contains 31 columns. Of those 31 columns, 28 of them were named as V1-V28, while the other three columns were Time, Amount and Class. Class column has only two values 0 and 1. Class 0 represents a valid transaction, while Class 1 represents a fraudulent transaction.

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27
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	-0.189115	0.133558
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	0.125895	-0.008983
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	-0.139097	-0.055353
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	-0.221929	0.062723
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	0.502292	0.219422
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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	1.436807	0.250034	0.943651
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	-0.606624	-0.395255	0.068472
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	0.265745	-0.087371	0.004455
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	-0.569159	0.546668	0.108821
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	-0.473649	-0.818267	-0.002415

284807 rows × 31 columns

Figure 2: Percentage of fraudulent vs non-fraudulent transactions

From the data we can see that each column was of float datatype and there were no null values in our data, so there was no need to perform any changes in the data. We then separated our data depending upon the class value into valid and fraud transactions and we got the following results.

```
Total transactions: 284807
Number of valid transactions: 284315
Number of fraudulent transactions: 492
Percentage of fraud transactions: 0.17
```

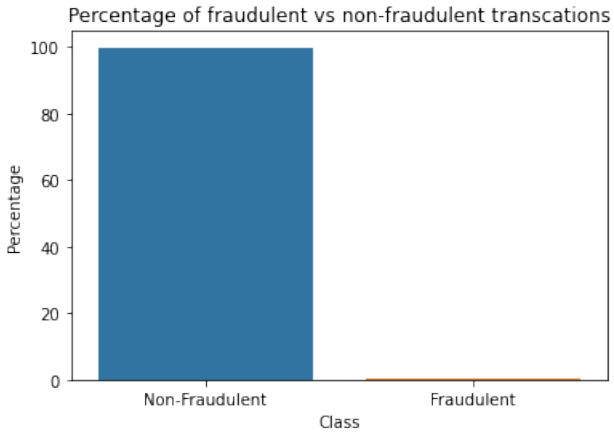

Figure 3: Percentage of fraudulent vs non-fraudulent transactions

From the results, we saw that the number of fraudulent transactions were very less as compared to that of valid transactions and hence our data was quitely imbalanced. Since, the number of fraudulent transactions were only 492, we continued with the same data to train and test our models on.

After properly understanding our data, we tried feature-refinement to reduce some unwanted features and try to make our model work simple. So, first we plotted the correlation matrix for all the features except Class and also calculated the mean-values for valid and fraud transactions to see if we can reduce some of the features.

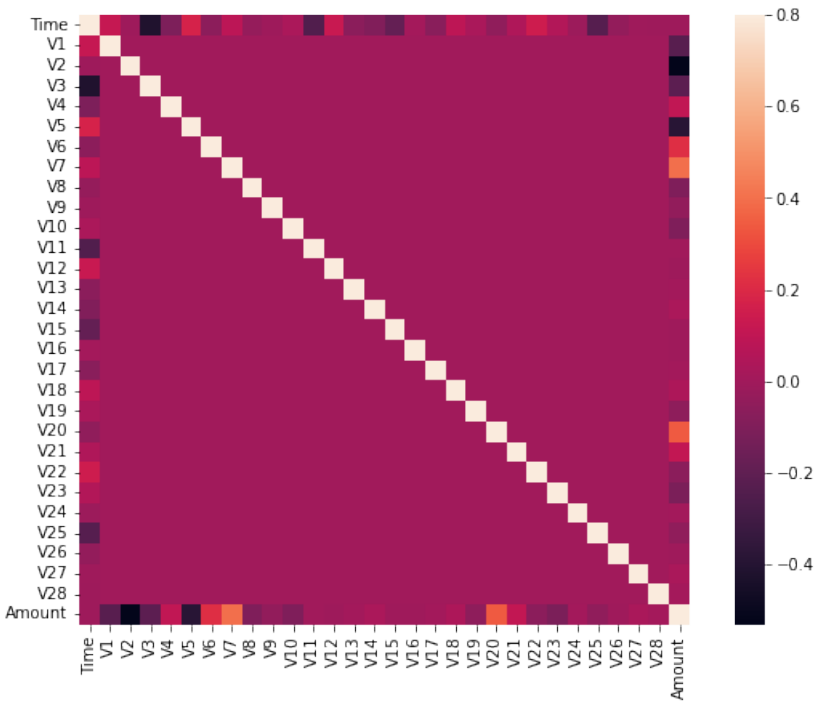

Figure 4: Correlation Matrix between 30 features

Class	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	V23	V24	V25	V26
0	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	0.002419	0.009637	-0.000987	0.004467	...	-0.000644	-0.001235	-0.000024	0.000070	0.000182	-0.000072	-0.000089
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731	0.570636	-2.581123	...	0.372319	0.713588	0.014049	-0.040308	-0.105130	0.041449	0.051648

Figure 5: Mean values for valid and fraud transactions

From the results we can see that the correlation between any two features from V1-V28 is almost 0 and also the correlation between Time and Amount with any other feature does not show any trend. Thus, all features are uncorrelated with each other. Also, we can see that the mean-values for any feature between valid transaction and fraud transaction are quite different and thus we cannot remove any feature and so we need to train our model with all the 30 features.

Since, we need to consider all the features to train our model, the best thing would be standardizing our dataset, so each feature will be equally as important. So, we plotted the histogram and calculated the mean and standard-deviation for each feature and we got the following results.

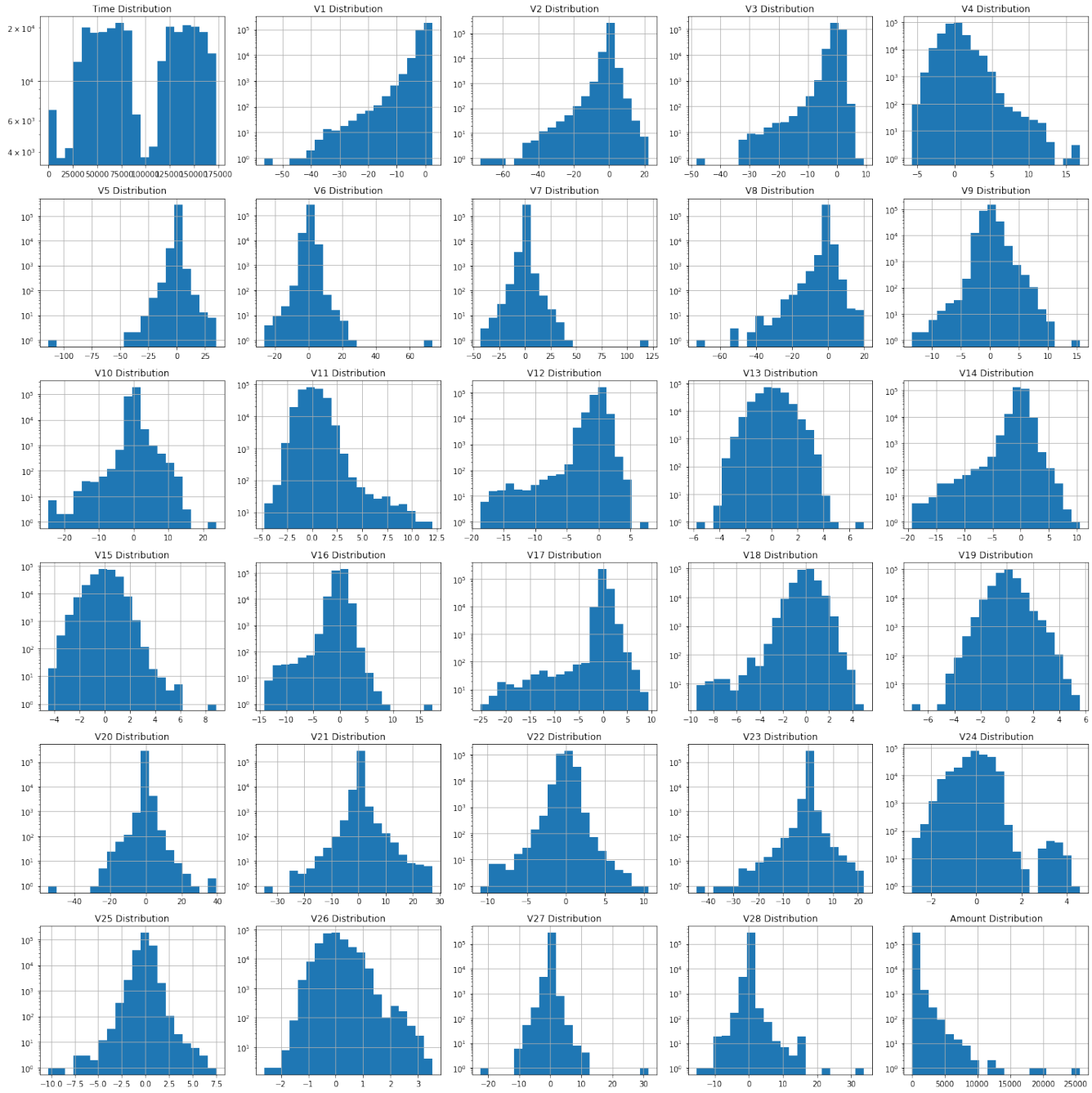

Figure 6: Histogram Plot for each feature

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1.213481e-16	-2.406331e-15	...	6.406204e-16	1.654067e-16	-3.568593e-16	2.57
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	...	7.709250e-01	7.345240e-01	7.257016e-01	6.24

Figure 7: Mean and Standard Deviation for each feature

From the results, we can see that all features except Time and Amount have $mean \approx 0$ and $std-dev \approx 1$ and are normal in nature. Thus, features V1-V28 were already standardized. Hence, we only need to standardize Time and Amount feature.

After standardizing our data, we used PCA to convert the data into 3-dimension and plotted the same to see if we can visually separate the fraud transactions with valid ones or not.

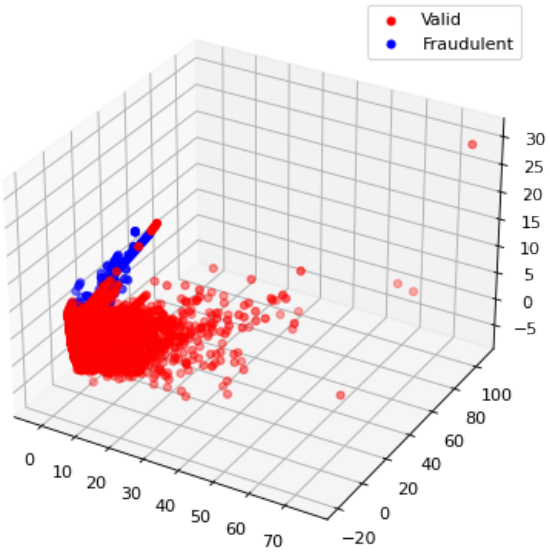

Figure 8: Visualization of standardized data in 3D using PCA

Training and Testing Models

After splitting the final standardized data into train-set and test-set, we applied 4 different machine learning models to predict whether the transaction is fraudulent or not and also calculated different metrics to compare them and analyze them.

The 4 models we used are:

1. Logistic Regression
2. Support Vector Classification (SVM)
3. Random Forest Classifier
4. Decision Tree Classifier

We also preformed cross-validation on our training data for each model to verify that the model does not under-fits or over-fits the training data.

Results

In the results section, we have plotted the confusion matrix and classification report for each model we trained and tested, and to verify whether the model does not under-fits or over-fits the training data we also did cross validation on training data for 10 iterations.

From cross-validation, we obtained the mean same as obtained earlier and also the variance was nearly equal to zero for each model, so we can be sure that all the models does not over-fits or under-fits the training data.

- Logistic Regression

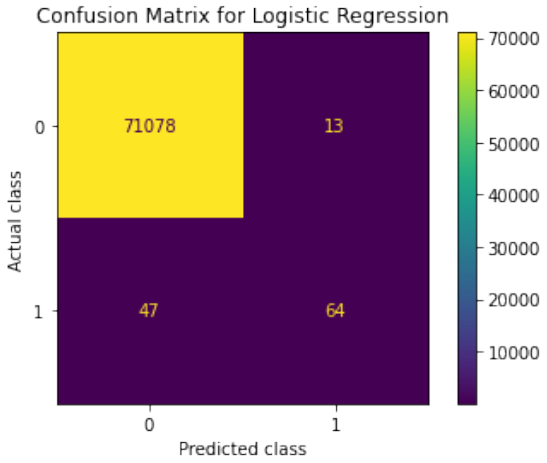

Figure 9: Confusion Matrix for Logistic Regression

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71091
           1       0.83      0.58      0.68       111

    accuracy                           1.00     71202
   macro avg       0.92      0.79      0.84     71202
weighted avg       1.00      1.00      1.00     71202

Results after using Cross-Validation
    Accuracy:
    Mean: 0.9992509540821971
    Standard Deviation: 0.00013424157589603104

    F1 Score:
    Mean: 0.7568735631731156
    Standard Deviation: 0.04708396398694048
```

- Support Vector Machine

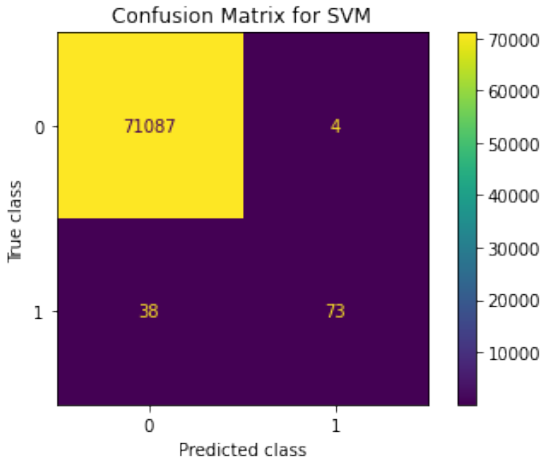

Figure 10: Confusion Matrix for Support Vector Machine

```
Classification Report:
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	71091
1	0.95	0.66	0.78	111
accuracy			1.00	71202
macro avg	0.97	0.83	0.89	71202
weighted avg	1.00	1.00	1.00	71202

- Random Forest Classifier

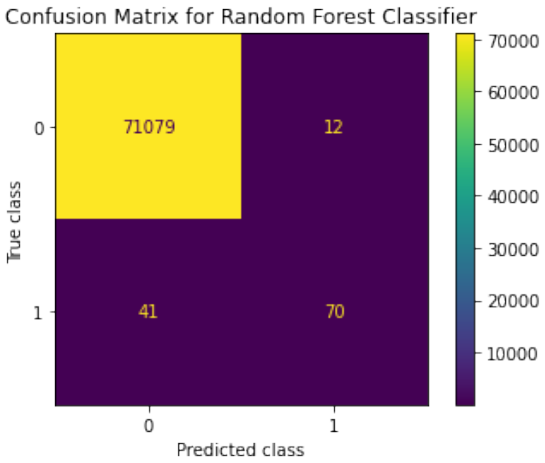

Figure 11: Confusion Matrix for Random Forest Classifier

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71091
           1       0.85      0.63      0.73       111

    accuracy                           1.00     71202
   macro avg       0.93      0.82      0.86     71202
weighted avg       1.00      1.00      1.00     71202

Results after using Cross-Validation
    Accuracy:
    Mean: 0.9993352215538026
    Standard Deviation: 6.193093118289965e-05

    F1 Score:
    Mean: 0.787948053246028
    Standard Deviation: 0.019613328423899814
```

- Decision Tree Classifier

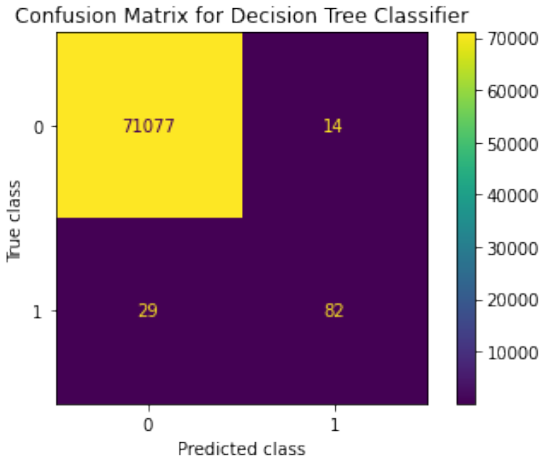

Figure 12: Confusion Matrix for Decision Tree Classifier

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71091
           1       0.85      0.74      0.79       111

    accuracy                           1.00     71202
   macro avg       0.93      0.87      0.90     71202
weighted avg       1.00      1.00      1.00     71202

Results after using Cross-Validation
    Accuracy:
    Mean: 0.9994148071575715
    Standard Deviation: 0.00011520656917410988

    F1 Score:
    Mean: 0.8229043111804899
    Standard Deviation: 0.03713289479077379
```

Table below shows the accuracy and F1-score obtained using the above 4 models for the test data. Since, the data is unbalanced, accuracy will not be the best approach to compare them and so F1-score can be considered to compare the models.

Model	Accuracy (in %)	F1-Score (in %)
Logistic Regression	99.92	68.08
SVM	99.94	77.66
Random Forest Classifier	99.93	72.54
Decision Tree Classifier	99.94	79.23

Table 1: Comparison between different models

Conclusion

In this report we tried different Machine Learning algorithms to detect fraudulent credit-card transactions. From all the results, we finally conclude that **Decision Tree Classifier** gives the best accuracy and F1-score amongst the 4 models used. While, SVM also shows good results to train and test.

We saw all the models are highly accurate, but one of the reason for this is that the number of valid transactions far outnumbered than fraudulent ones ($\approx 300,000 : 500$). But since we have used a real data-set, the models have worked very well. Also, our data being imbalanced we showed that our model does not over-fits or under-fits the data and so we can say that we can use the data for real-life applications.

References

[1] Credit Card Fraud Detection Kaggle Dataset

[2] Credit Card Fraud Detection Using Machine Learning & Python

[3] Credit Card Fraud Detection using Machine Learning and Data Science

[4] ML — Credit Card Fraud Detection