Credit Card fraud detection using Isolation Forest

Spoorthy Reddy Jarugu

Vellore Institute of Technology
India

October 12, 2023

Overview

Problem Definition

Credit card fraud detection

Defn

Credit Cards are the most commonly used mode of payment nowadays. The reason is it has multiple features, which make it easy for users to make payments on the spot. Credit card fraud can be defined as any unauthorized use of a credit card, such as using a stolen credit card or making unauthorized purchases with a valid credit card. The dataset is taken from Kaggle.

- The dataset contains transactions made by credit cards in September 2013 by European cardholders.
- Unfortunately, due to confidentiality issues, original features are not given. The features provided are the result of the PCA transformation.
- There are a variety of techniques that can be used to detect credit card fraud. One common technique is to use machine learning models to identify patterns in fraudulent transactions.

Dataset Description

Name	Count
Rows	284807
Columns	31

Rows

- This dataset contains 284807 rows of vales.
- The entire dataset has 284315 Valid transactions, and 492 are Fraud transactions.

Columns

- The total number of columns present in the dataset is 31
- Time - Time of the transaction happened, V1 to v28 - are the principle component obtained with PCA, Amount - transaction amount, Class - valid or fraud transaction

Data Preprocessing and Visualization

Loading the data

- After the import statements, the initial step is to load the data
- Setting the proper path where the data is available is an important step

```
data = pd.read_csv('./dataset/creditcard.csv')
data.head()
```

Figure 1: Loading the data

Cleaning the data

- Checking for missing values of each column
- Checking for the data type of each column

```
data.isnull().sum()

Time     0
V1       0
V2       0
V3       0
V4       0
V5       0
V6       0
V7       0
V8       0
V9       0
V10      0
V11      0
V12      0
V13      0
V14      0
V15      0
V16      0
V17      0
V18      0
V19      0
V20      0
V21      0
V22      0
V23      0
```

Figure 2: Checking for missing values

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 32 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
```

Figure 3: Data type of each column

Visual representation of the feature class

- Number of classes with respect to frequency that are Valid transaction and Fraud transaction

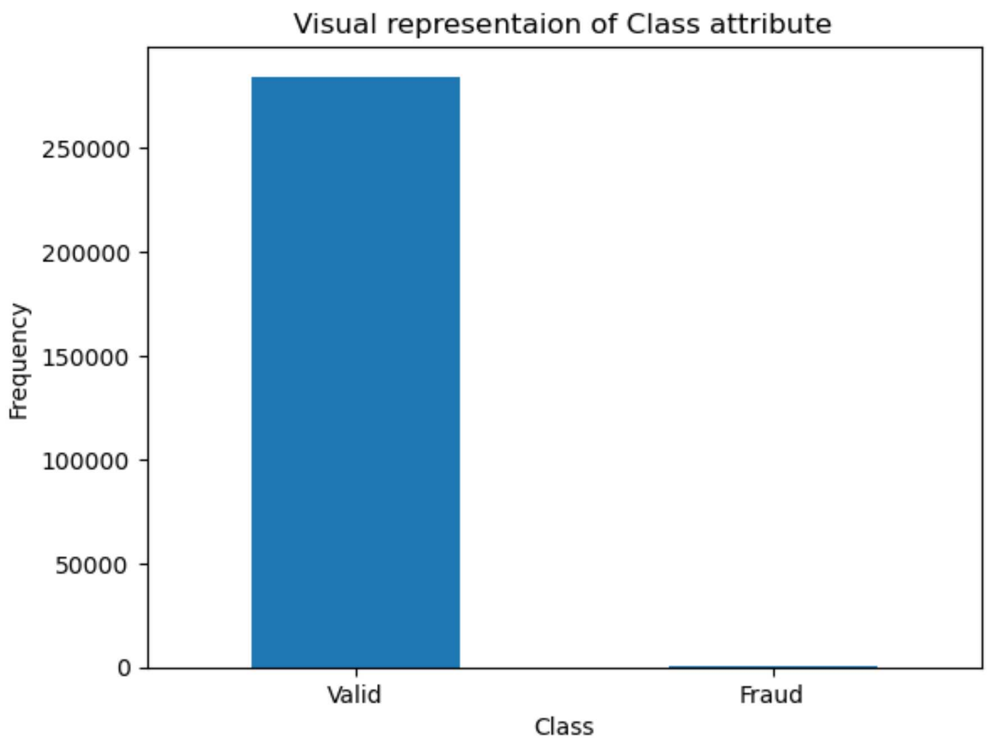

Figure 4: Class Visualization

- From the above diagram, we can see that Valid transactions are way more significant than fraud transactions

Statistical measures of the feature Class

- Diving the feature 'Class==0' as a Valid dataset.

```
print("Details of Valid transaction")
Valid.Amount.describe()
```

```
Details of Valid transaction

count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

Figure 5: Details of Valid transaction

- Diving the feature 'Class==1' as a Fraud dataset.

```
print("Details of the Fraud transaction")
Fraud.Amount.describe()
```

```
Details of the Fraud transaction

count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

Figure 6: Details of Fraud transaction

Valid and Fraud transactions with respect to Amount

- Trying to visually see how many different transactions are for Fraud and Valid in terms of Amount.
- From the visual representation, we can see Fraud transactions are of a small amount compared to Valid transactions.

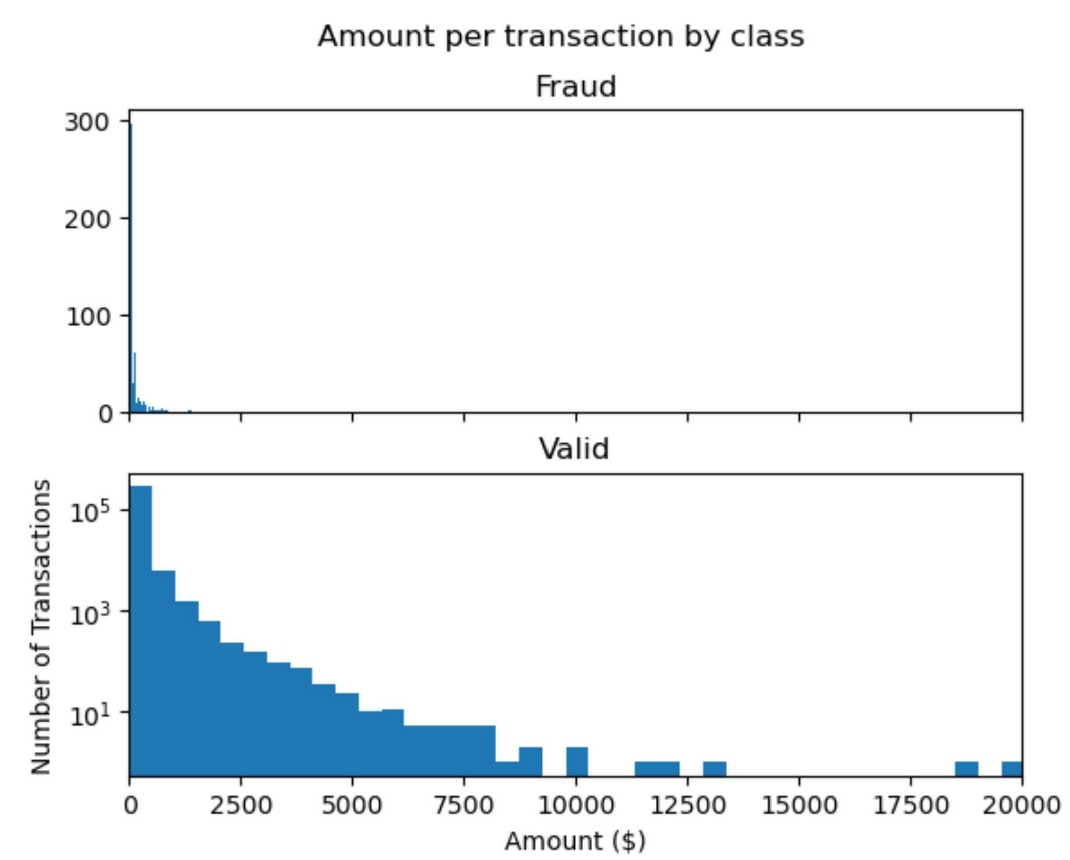

Figure 7: Transactions with respect to Amount

Valid and Fraud transactions with respect to Time

- Trying to visually see how many different transactions are for Fraud and Valid in terms of Time.

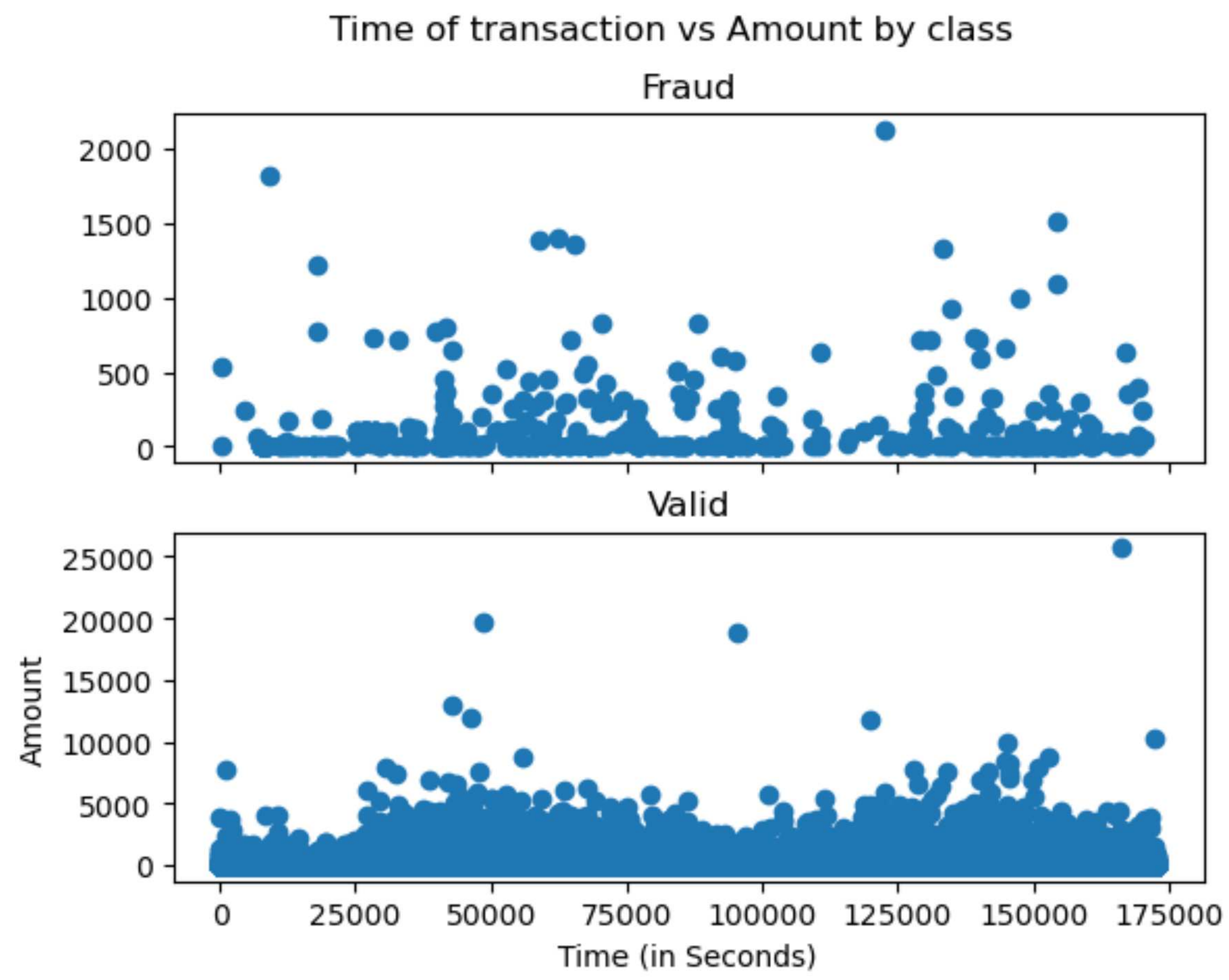

Figure 8: Transactions with respect to Time

Correlation

- Correlation is a statistical measure used to determine if there is a relationship between two variables and how strongly that is related.
- Correlation coefficients range from -1 to +1. A correlation coefficient of -1 indicates a perfect negative correlation, which means that the two variables move in opposite directions. A correlation coefficient of +1 indicates a perfect positive correlation, which means that the two variables move in the same direction.
- A correlation coefficient of 0 indicates no correlation, meaning there is no relationship between the two variables.
- To avoid multicollinearity and improve model performance, we will remove the highly correlated variables to reduce the redundancy of the two variables.

Correlation

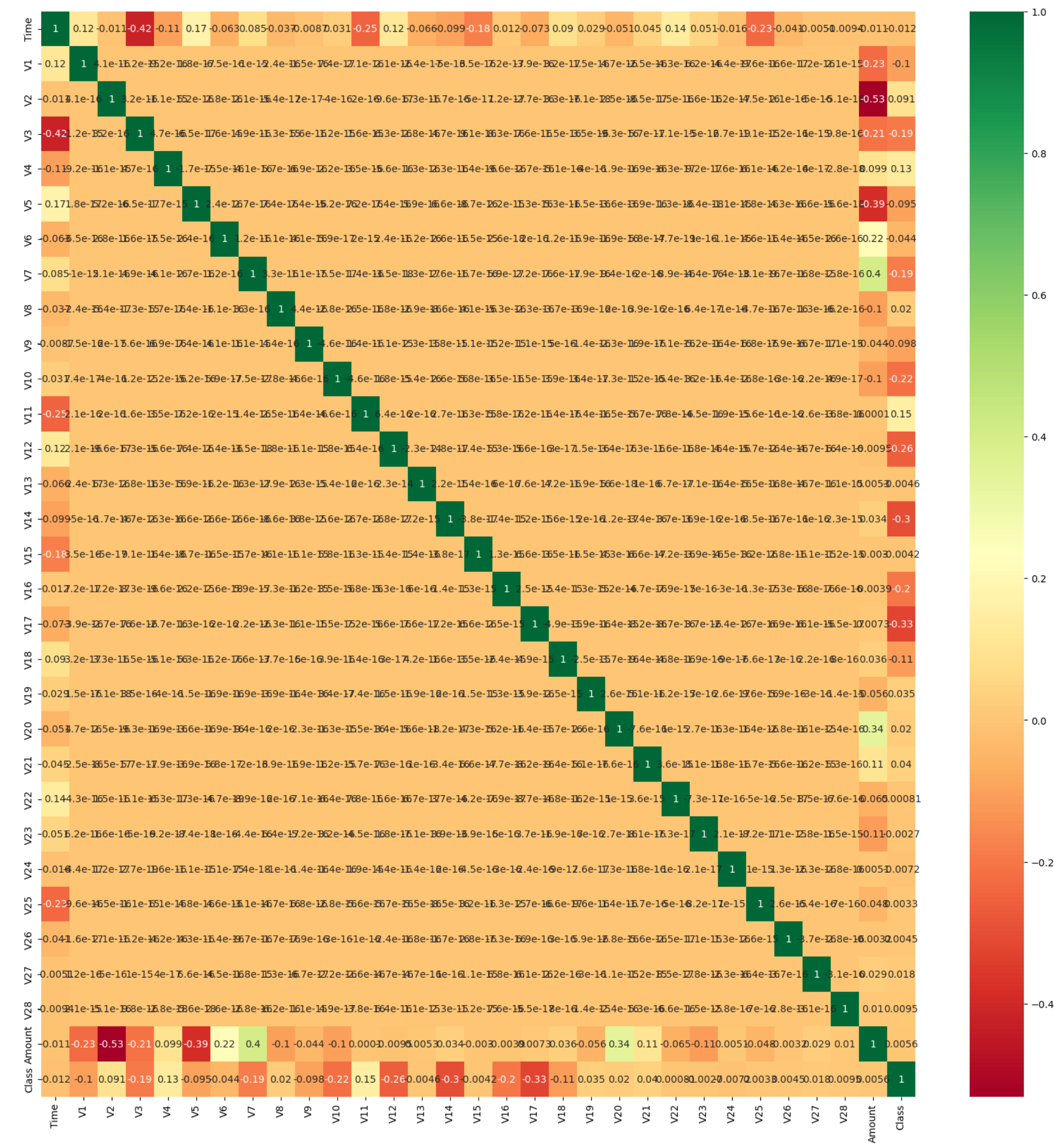

Figure 9: Correlation matrices

Model built and Prediction

Isolation Forest

- The Isolation Forest algorithm is an unsupervised anomaly detection algorithm that works by isolating anomalies by randomly selecting features and split values. The algorithm builds a forest of isolation trees, where each tree is trained on a random sample of the data. The algorithm then calculates the anomaly score for each data point by measuring how deep it is in the forest. Anomalies are typically assigned lower
- The Isolation Forest algorithm is a powerful tool for anomaly detection. It is fast, efficient, and robust to outliers and noise. It can be used to detect anomalies in various applications, including fraud detection, intrusion detection, medical diagnosis, network monitoring, and financial market analysis.
- Here, for this task, we are using the Isolation Forest for Credit card fraud detection.

Isolation Forest

- This is how we create new instances for isolation forest IsolationForest(n_estimators=1000, max_samples="auto", contamination=0.1, random_state=42)
  - n_estimators: This parameter specifies the number of trees to build in the forest. A higher value of n_estimators will result in a more robust model but will also take longer to train.
  - max_samples: This parameter specifies the maximum number of samples to be used for training each tree in the forest. An " auto " value means that all the samples will be used to train each tree.
  - contamination: This parameter specifies the proportion of outliers that the IsolationForest model is expected to find in the data. A value of 0.1 means that the model is expected to find 10% of the data points to be outliers.
  - random_state: This parameter specifies the random seed to use. Setting the random state to a fixed value will ensure the model produces consistent results each time it is trained.

Local Outlier Factor

- Local Outlier Factor (LOF) is an unsupervised anomaly detection algorithm that identifies outliers based on their local density. LOF is calculated by comparing the local density of a data point to the local densities of its neighbours.
- In LOF a data point is considered to be an outlier if it has a significantly lower local density than its neighbors.
- The new instance of Local Outlier Factor is created as follows: LocalOutlierFactor(n_neighbors=50,leaf_size=10, contamination=0.1)
  - n_neighbors: This parameter specifies the number of nearest neighbours to consider when calculating the local density of a data point. A higher value of n_neighbors will result in smoother local density estimates.
  - leaf_size: This parameter specifies the maximum number of data points that can be stored in a single node of the tree used to calculate the nearest neighbours of a data point. A higher value of leaf_size will result in faster computation, but it may also lead to less accurate results.
  - contamination: This parameter specifies the proportion of outliers the LocalOutlierFactor model is expected to find in the data.

Comparing iForest and LOF

- Both the algorithms gave a reasonable accuracy rate. However, Isolation Forest (iForest) effectively fetches outliers for large datasets, whereas Local Outlier Factor (LOF) algorithms are computationally expensive for large datasets.
- iForest is generally more interpretable than LOF.
- LOF has more hyperparameters to tune than iForest.

Algorithm	Accuracy
Isolation Forest	0.997156
Local Outlier Factor	0.996524

Figure 10: Accuracy Report

Classification Report

- Classification report is the performance of an Isolation Forest model on a given dataset.
  - Precision is the fraction of positive predictions that are positive.
  - Recall is the fraction of actual positives that are correctly identified.
  - F1 score is a harmonic mean of precision and recall.
  - Support is the total number of examples in a given class.

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

           0       1.00      1.00      1.00     28432
           1       0.26      0.27      0.26        49

    accuracy                           1.00     28481
   macro avg       0.63      0.63      0.63     28481
weighted avg       1.00      1.00      1.00     28481
```

Figure 11: Classification Report

Classification Report

- The model has perfect accuracy on the Valid Transaction (i.e., class 0), with a precision of 1.0 and a recall of 1.0. However, the model performs poorly on the Fraud transaction (class 1), with a precision of 0.26 and a recall of 0.27.
- Overall, the classification report shows that the model has good performance on the majority class that is a Valid transaction, but poor performance on the minority class that is a Fraud transaction. This is a common problem with classification models, which are often trained on datasets with imbalanced class distributions.
- Here are some ways to improve the performance of the model on the minority class:
  - Use a weighted loss function to give more weight to the minority class.
  - Use oversampling or undersampling techniques to balance the class distribution of the training set.
  - Use a different classification algorithm better suited for imbalanced datasets.

Analysis

- From the above visual representation of the class feature, we can understand that Valid transactions are more in number than Fraud transactions. When understanding the percentage, it seems 0.17% are fraud transactions in the entire data.
- As it is very low, this data is highly imbalanced data.
- In general, to balance the dataset, we use two methods:
  - Undersampling: This approach involves reducing the number of samples in the majority class by randomly removing samples. This can be done until the majority class has the same number of samples as the minority class.
  - Oversampling: This approach involves increasing the number of samples in the minority class by creating synthetic samples. This can be done using various techniques, such as SMOTE (Synthetic Minority Over-sampling Technique).

Evaluation Result

- Undersampling the dataset is done to get accurate results for iForest and LOF algorithms.
- We can see that the iForest algorithm outperformed the LOF algorithm and obtained an accuracy of 97.5%. Whereas LOF obtained 87.8%.

Algorithm	Accuracy
Isolation Forest	0.9756
Local Outlier Factor	0.878647

Figure 12: Accuracy Report

- Outliers obtained are 350

- Overall, the classification report shows that the model performed very well on class 0 data and moderately well on class 1 data. The model has a high accuracy, precision, and recall for class 0 data. The model has a moderate accuracy, precision, and recall for class 1 data.

```
Isolation Forest: 350
Classification Report :
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     15000
           1       0.64      0.65      0.65       492

    accuracy                           0.98     15492
   macro avg       0.82      0.82      0.82     15492
weighted avg       0.98      0.98      0.98     15492
```

Figure 13: Classification Report

Conclusion

Conclusion

- Flip 01 task I have chosen "Credit Card Fraud Detection using Isolation Forest"
- The Isolation Forest algorithm has a number of advantages over other anomaly detection algorithms :
  - It is robust to outliers and noise.
  - It can be used to detect anomalies in both high-dimensional and low-dimensional data.
  - It is easy to implement and interpret.
- Using Isolation Forest for detecting anomalies gives very good accuracy compared to other algorithms.

Questions?

Contact Information

Spoorthy Reddy Jarugu
Vellore Institute of Technology
Vellore, India