

CSE422: Artificial Intelligence

Project Report

Project Title: Credit Card Fraud Detection



Group No: 07 Inspiring Excellence

CSE422 Lab Section: 06, Spring 2024

ID	Name
20301194	Mahdi Hossain
21201236	Noshin Fouzia Tasnim

Table of Contents

Section No	Content	Page No
1	Introduction	3
2	Dataset Description	3-4
3	Dataset Preprocessing	5
4	Feature Scaling	5
5	Dataset Balancing	5-6
6	Dataset Splitting	7
7	Model Training and Testing	7-9
8	Model Selection/Comparison Analysis	10-18
9	Conclusion	19

Introduction

In this report, we present a machine learning project that has been trained to predict fraudulent and non-fraudulent credit cards. Fraud detection in credit cards is an important concern as many scammers try to scam service companies by giving fraudulent credit cards to gain their objective in a dishonest way. Thus, with our project, we have trained a machine-learning model that can detect fraudulent credit cards.

Dataset Description

Source:

Link: <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data>

Reference: Andrea(2018). Credit card fraud detection [Dataset].

Features:

The dataset has a total of 31 features, of which 30 are quantitative features and 1 is categorical feature.

- It contains only numeric values which are the result of a PCA transformation. To protect confidentiality issues, the original features and background information of the transactions have not been provided. Features V1, V2, ... V28 are the principal components obtained with PCA and represent transaction details
Datatype: Float
- 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset.
Datatype: Float
- The amount is the transaction amount
Datatype: Float
- Feature 'Class' is the binary response variable. It takes a value 1 for fraudulent transactions and 0 otherwise.
Datatype: Integer

Problem type:

As our target is to detect whether the transaction is fraudulent or not and the target variable is a binary response variable, it is a classification problem.

Data points:

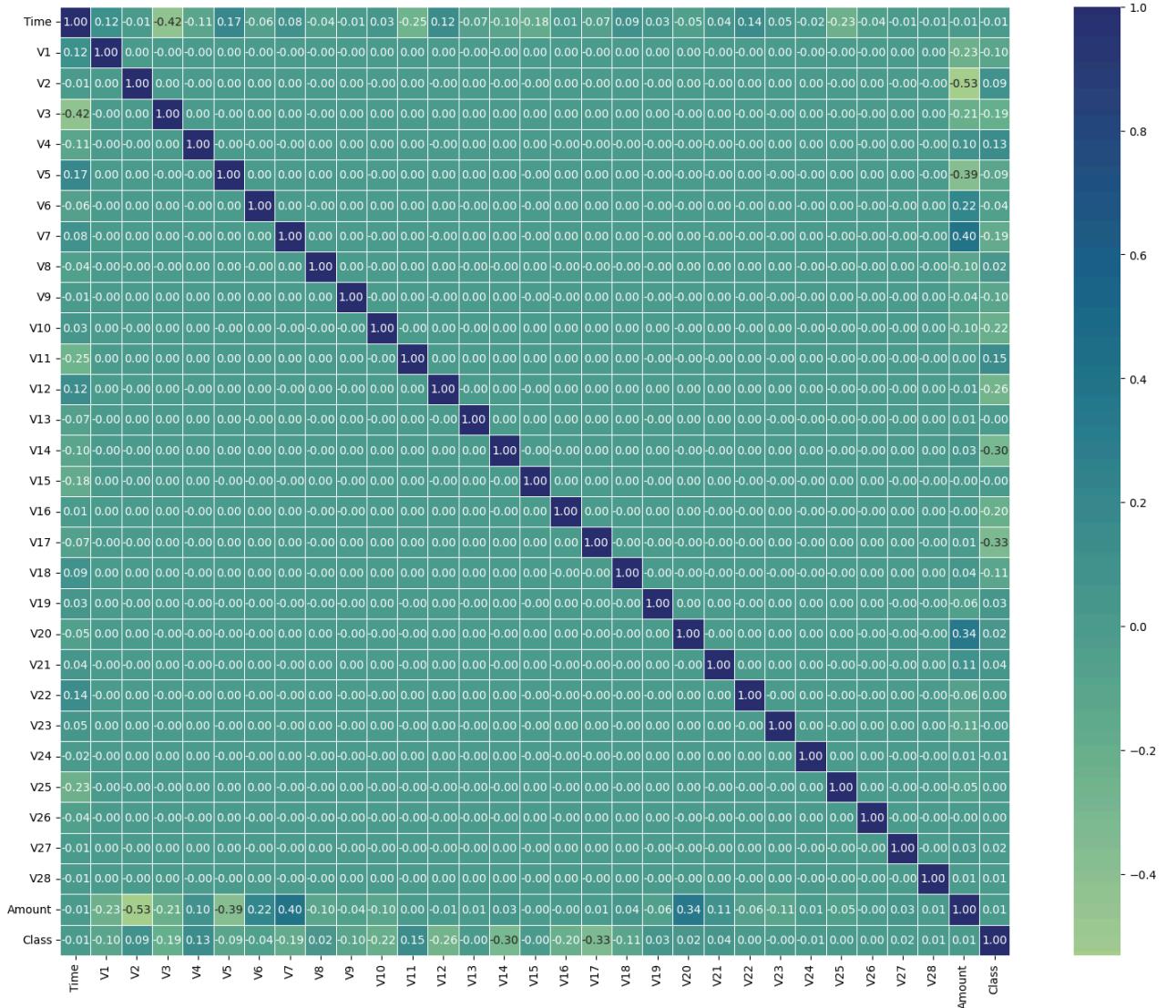
The number of data points in our dataset is 284807.

Feature type:

There are 30 quantitative variables, Time, V1, V2, ..., V28, and Amount, which are in float. The target variable, Class is categorical and the data type is integer.

Correlation of features:

A heatmap generated using Seaborn library of the correlations of the features is as follows:



Dataset Pre-processing

Faults and solutions:

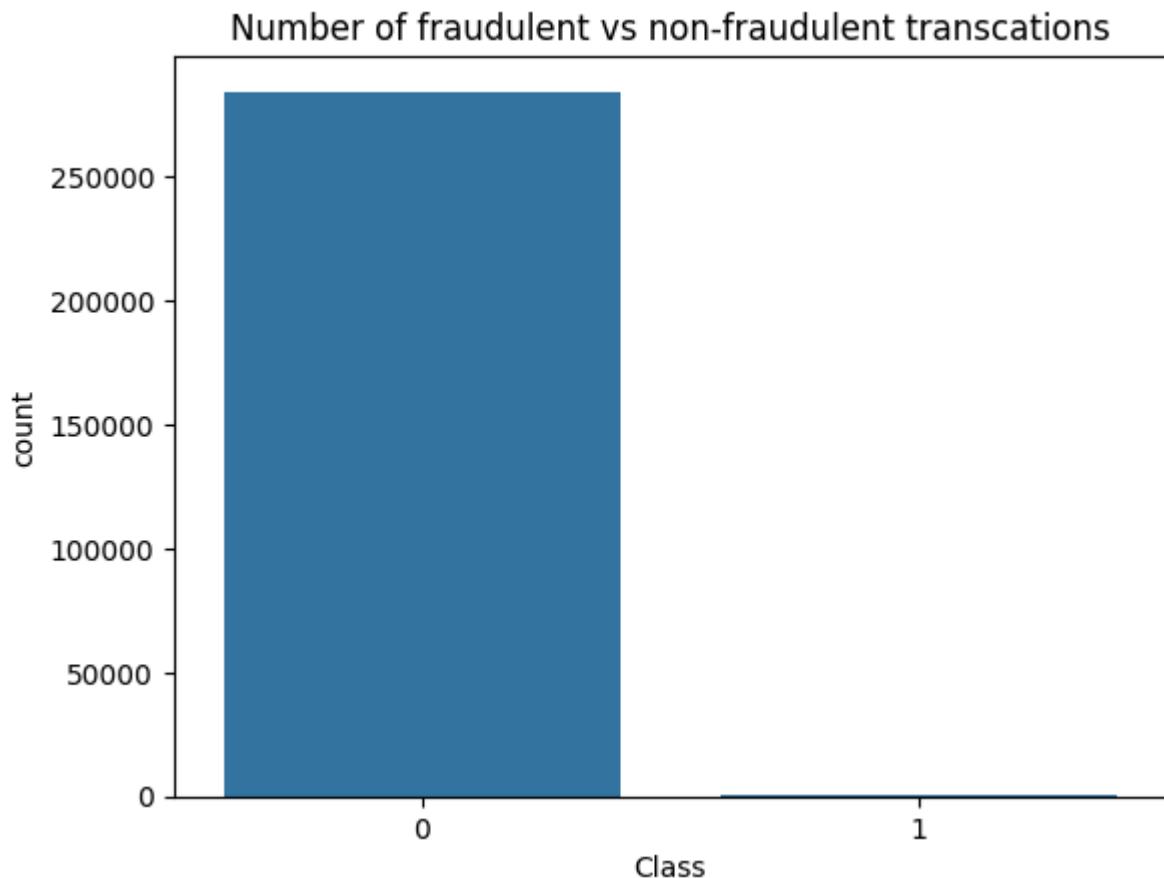
- Null values:
There are no null values in the dataset. So no rows or columns had to be deleted.
- Categorical values:
The dataset has no categorical values, except the target variable. And it is a binary class, represented by 0 or 1. So, categorical features are not encoded.
- Outliers:
There are no outliers in the dataset. So outliers were not handled.
- Duplicates:
The dataset contained a lot of duplicate rows. The duplicated rows were dropped from the dataset.
- Deleting Time column:
The ‘Time’ column is unnecessary as it does not particularly affect the target variable, ‘Class’. So, the Time column was dropped from the dataset.

Feature Scaling

Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step. In order to deal with any data that would affect the training of a model by being in a range that is unnecessary we used the standard scaling library from sklearn to apply feature scaling of our dataset.

Dataset Balancing

For the dataset we have used, we have found it to be an imbalance in the case of the number of fraudulent and non-fraudulent credit cards. Representing the dataset using a bar chart of N classes before applying the balancing techniques:



To properly train the model we have to first balance the dataset to make the model identify fraudulent credit cards properly. We have chosen to do both oversampling and undersampling.

Oversampling: It is used when the quantity of data is insufficient. It tries to balance the dataset by increasing the size of rare samples. Rather than getting rid of abundant samples, new rare samples are generated by using e.g. repetition, bootstrapping, or SMOTE (Synthetic Minority Over-Sampling Technique).

Undersampling: Under-sampling balances the dataset by reducing the size of the abundant class. This method is used when the quantity of data is sufficient. By keeping all samples in the rare class and randomly selecting an equal number of samples in the abundant class, a balanced new dataset can be retrieved for further modeling.

Dataset Splitting

After each case of handling the imbalanced dataset, we have split the dataset into 70% for training and 30% for testing. Then we trained the model using proper machine learning algorithms.

Model Training and Testing

KNN (for classification problem):

K-Nearest Neighbors or KNN in short is a simple and versatile classification algorithm used for both binary and multiclass classification problems. KNN is a non-parametric learning method that doesn't require training before producing predictions and doesn't make any assumptions about the distribution of the underlying data. KNN is comparatively easy to implement, however, the algorithm can be computationally expensive, especially with large datasets, as it requires calculating distances for each prediction.

Logistic Regression (for classification problem):

Logistic Regression is a statistical method commonly used in machine learning for binary classification problems. Logistic Regression is often a good starting point for binary classification tasks due to its simplicity, interpretability, and efficiency. It can handle any number of classes.

Decision Tree (for classification problem):

A decision tree is a decision-support hierarchical model that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

Random Forest (for classification problem):

Random Forest is a versatile ensemble learning method that is used for classification tasks. This algorithm builds a forest of decision trees, each trained on a random subset of the training data. The final classification is determined by a majority vote (for example, "voting" for the most common class among all the trees) or by averaging the predicted probabilities across the trees. It is effective in handling complex tasks, dealing with high-dimensional data, and providing robust predictions.

Evaluation metrics:

For testing the models we have used below evaluation metrics for the test dataset we split earlier:

Accuracy score: It is an evaluation metric that measures the number of correct predictions made by a model in relation to the total number of predictions made.

Precision score: It is an evaluation metric that measures how many selected instances are relevant.

Recall score: It is an evaluation metric that measures how many relevant instances are selected.

F1 score: This is the harmonic mean of precision and recall.

Now, the accuracy of each algorithm we have used after using both oversampling and undersampling is as follows:

KNN Classifier:

KNN Accuracy: 0.9904 (after oversampling)

KNN Accuracy: 0.9437 (after undersampling)

Decision Tree:

After oversampling

- Accuracy: 0.9980
- Precision score: 0.9972185383829967
- f1 score: 0.9980091496896271

After undersampling

- Accuracy: 0.9120
- Precision score: 0.8940397350993378
- f1 score: 0.9152542372881357

Logistic Regression:

After oversampling

- Accuracy: 0.94375
- Precision score: 0.9723991241789177

After undersampling,

- Accuracy: 0.9296
- Precision score: 0.9305555555555556

Random Forest:

After oversampling

- Accuracy: 0.9998
- Precision score: 0.9996827224761748
- Recall score: 1.0
- f1 score: 0.9998413360678376
-

After undersampling,

- Accuracy: 0.9437
- Precision score: 0.9637681159420289
- Recall score: 0.9236111111111112
- f1 score: 0.9432624113475179

Model Selection/Comparison Analysis

As we have done both undersampling and oversampling for the datasets, we have also generated classification reports, confusion matrix and bar chart comparison of prediction accuracy, and bar chart comparison between precision and recall scores of each model for both undersampling and oversampling.

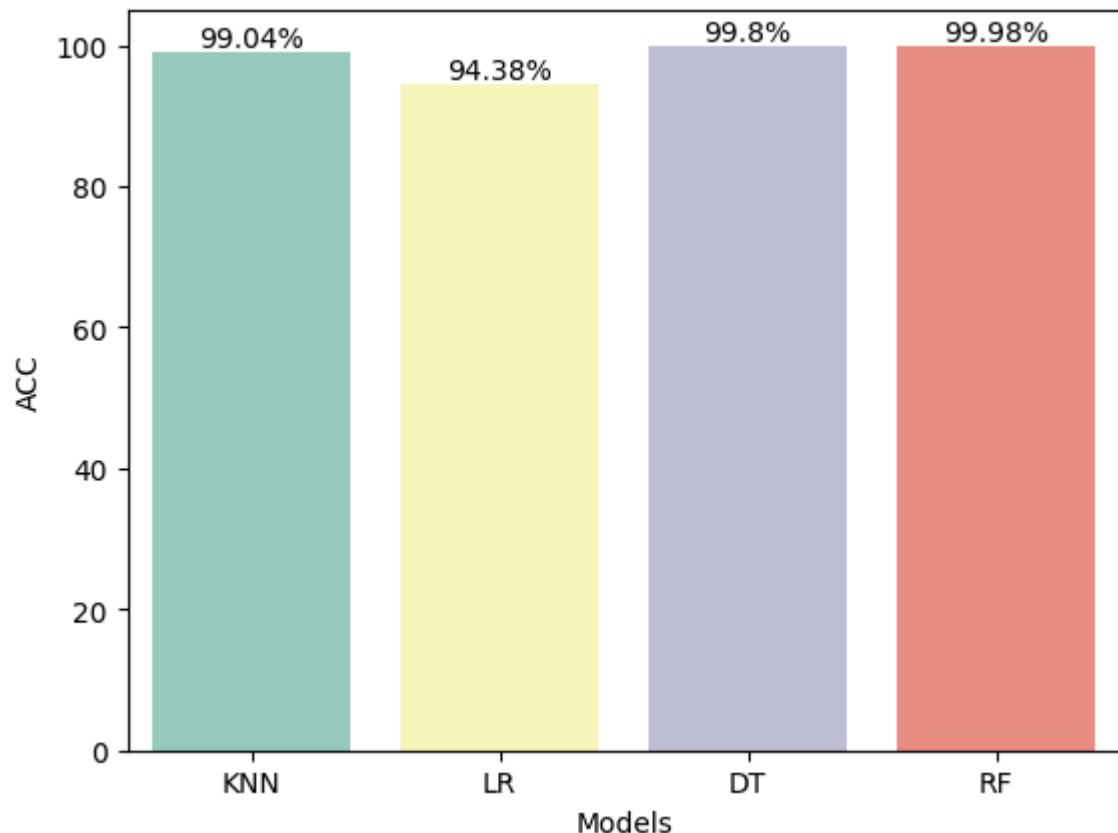
The precision score measures how often the machine learning model accurately predicts the positive class. It is the ratio of true positive predictions to the total predicted positives.

Recall measures how often a machine learning model correctly identifies true positive instances from all the positive samples.. It is the ratio of true positive predictions to the total actual positives.

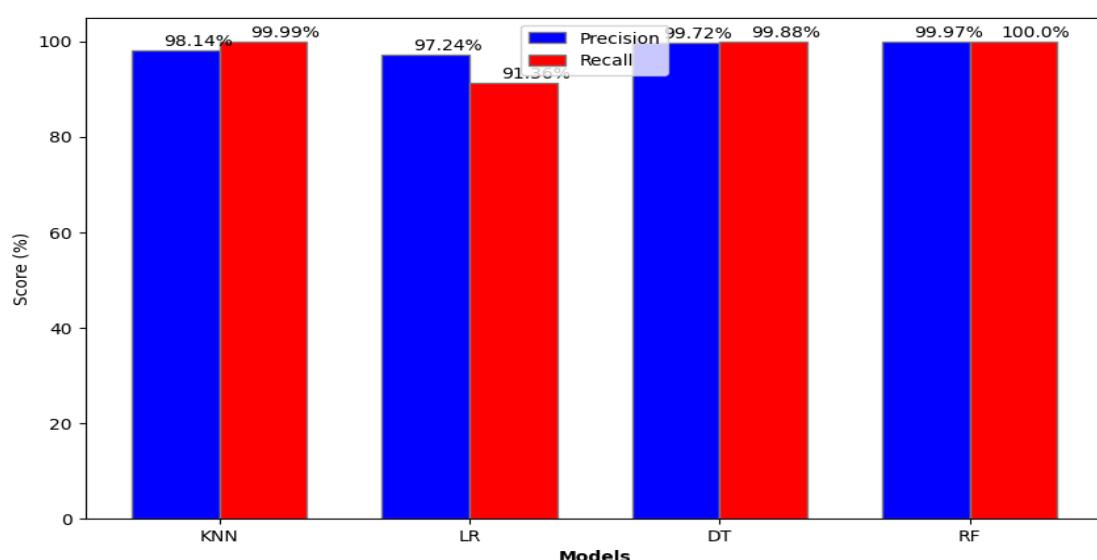
In the confusion matrix, the rows represent the actual values and the columns represent the predicted values. The first part is True Negative (negative predicted as negative), the second one is False Positive (Negative predicted as positive), the third part is False Negative (Positive predicted as Negative) and the lower right part is True Positive (Positive predicted as Positive). If not fraud is considered as negative and fraud is considered as positive, True negatives are not fraud predicted as not fraud, false positives are not fraud predicted as fraud, False Negatives are Fraud predicted as not fraud, and True Positives are Fraud predicted as Fraud.

Oversampling:

Bar chart showing the comparison between accuracy scores of all 4 models after oversampling the dataset:



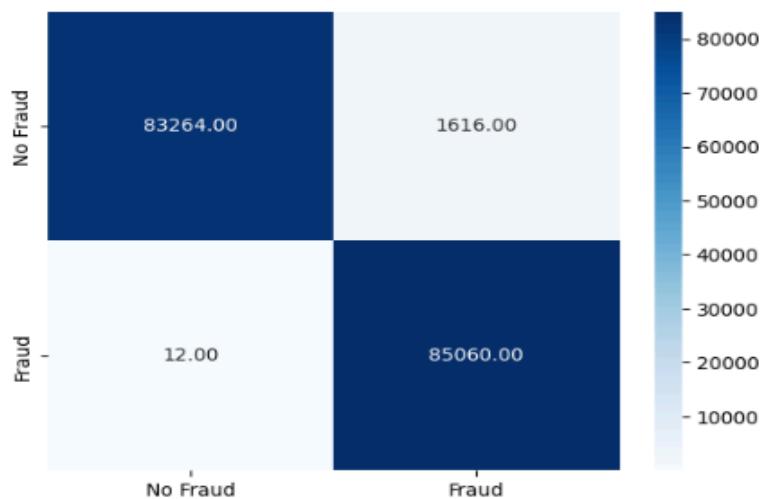
Bar chart showing the comparison between precision and recall scores of each model after oversampling:



After Oversampling, the classification report and confusion matrix of each model are as follows:

KNN Classifier:

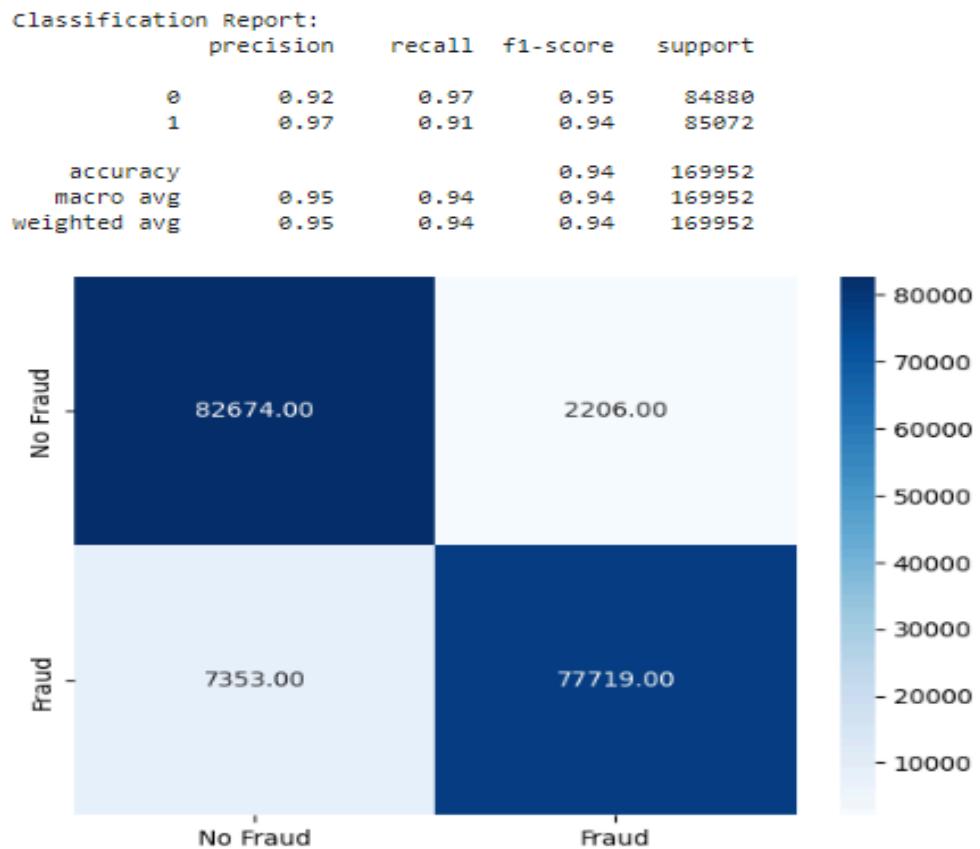
KNN Classification Report:					
	precision	recall	f1-score	support	
0	1.00	0.98	0.99	84880	
1	0.98	1.00	0.99	85072	
accuracy			0.99	169952	
macro avg	0.99	0.99	0.99	169952	
weighted avg	0.99	0.99	0.99	169952	



Decision Tree:

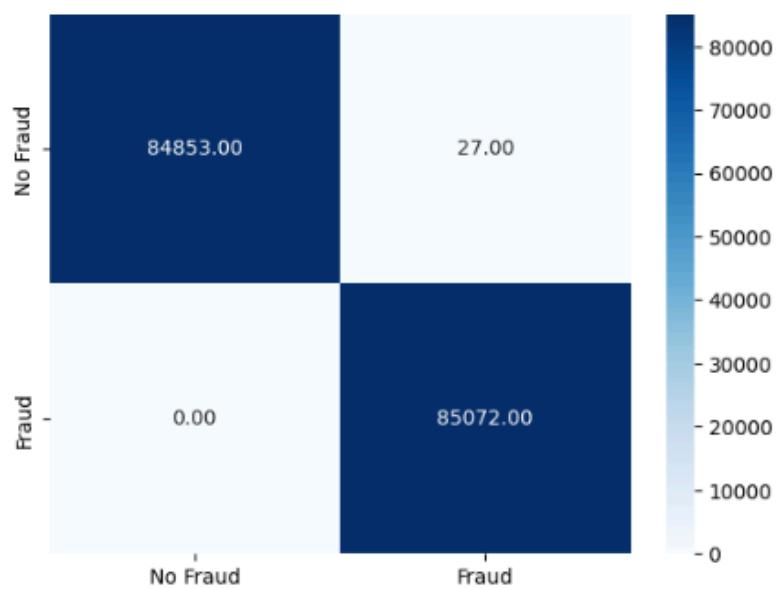
Classification Report:					
	precision	recall	f1-score	support	
0	1.00	1.00	1.00	84880	
1	1.00	1.00	1.00	85072	
accuracy			1.00	169952	
macro avg	1.00	1.00	1.00	169952	
weighted avg	1.00	1.00	1.00	169952	



Logistic regression:

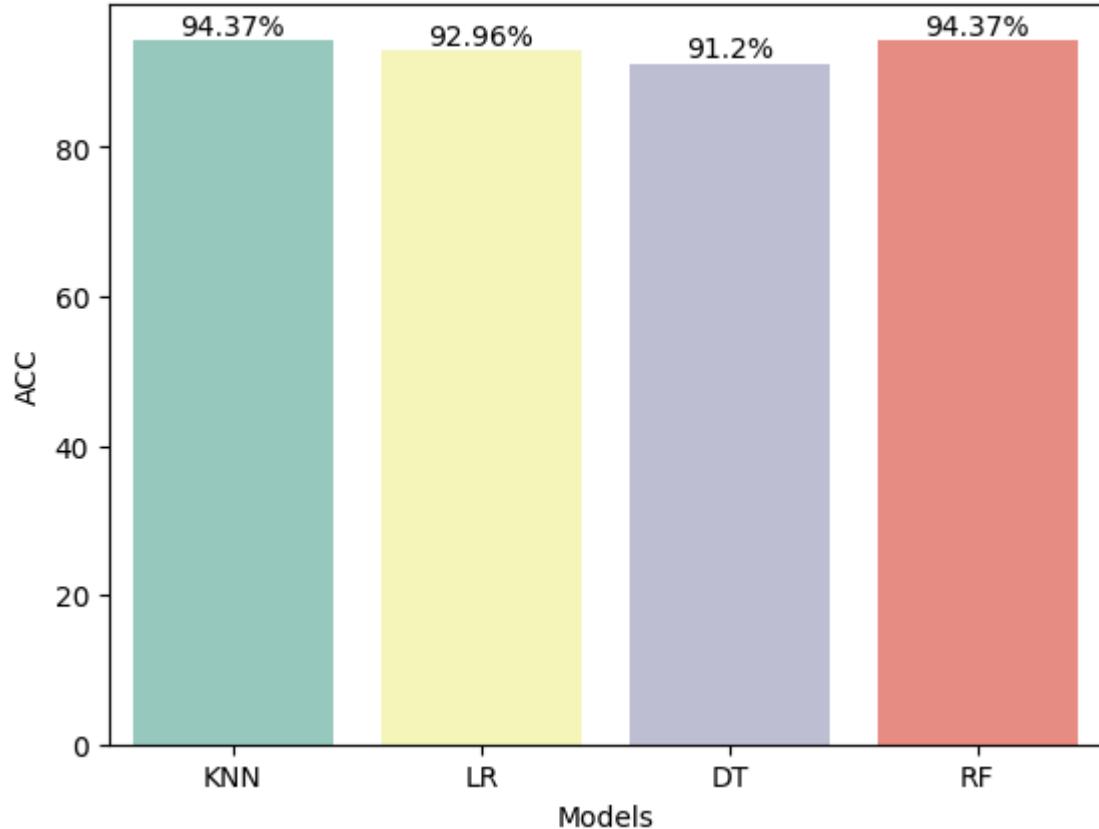
Random Forest:

```
Classification Report:  
precision    recall    f1-score   support  
  
          0       1.00      1.00      1.00     84880  
          1       1.00      1.00      1.00     85072  
  
    accuracy                           1.00     169952  
   macro avg       1.00      1.00      1.00     169952  
weighted avg       1.00      1.00      1.00     169952
```

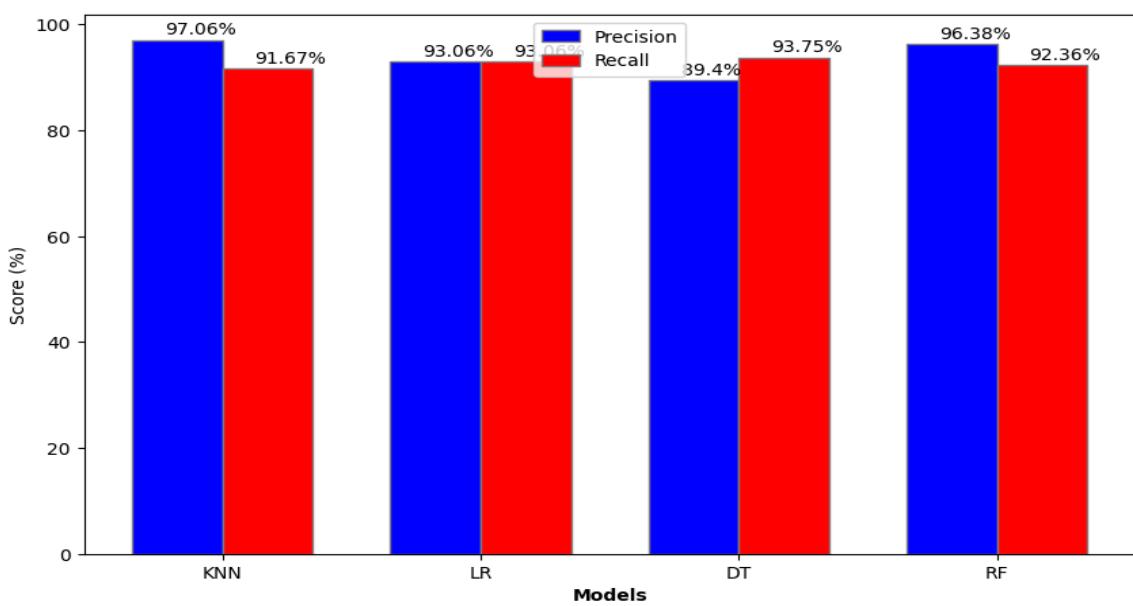


Undersampling:

Bar chart showing the comparison between accuracy scores of all 4 models after undersampling the dataset:



Bar chart showing the comparison between precision and recall scores of each model after undersampling:



After Undersampling, the classification report and confusion matrix of each model are as follows:

KNN:

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

          0       0.92      0.97      0.94     140
          1       0.97      0.92      0.94     144

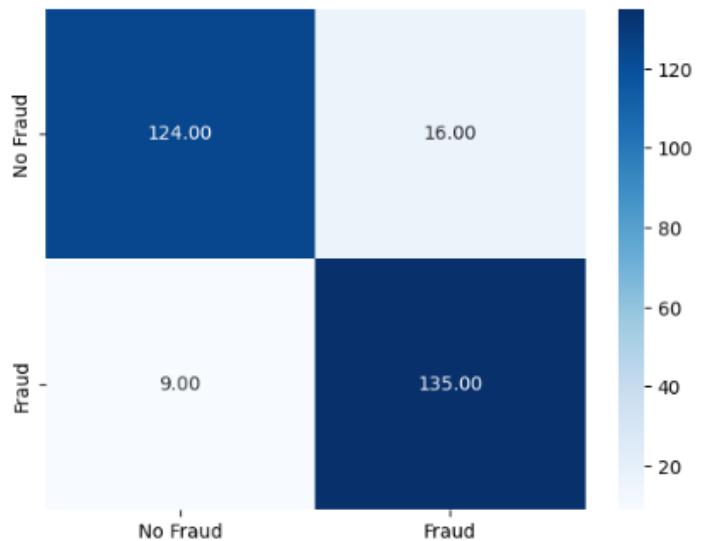
   accuracy                           0.94      284
  macro avg       0.94      0.94      0.94      284
weighted avg       0.95      0.94      0.94      284
```



Decision Tree:

```
Classification Report:
precision    recall   f1-score   support
          0       0.93      0.89      0.91      140
          1       0.89      0.94      0.92      144

accuracy                           0.91      284
macro avg       0.91      0.91      0.91      284
weighted avg    0.91      0.91      0.91      284
```



Logistic Regression:

```
Classification Report:
precision    recall   f1-score   support
          0       0.93      0.93      0.93      140
          1       0.93      0.93      0.93      144

accuracy                           0.93      284
macro avg       0.93      0.93      0.93      284
weighted avg    0.93      0.93      0.93      284
```



Random Forest:

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

          0       0.92      0.96      0.94     140
          1       0.96      0.92      0.94     144

   accuracy                           0.94      284
  macro avg       0.94      0.94      0.94      284
weighted avg       0.94      0.94      0.94      284
```



So from analysing our ML models' accuracy and other evaluation metrics, we can observe that Random Forest performs better than the other models for both oversampling and KNN performs better for undersampling. Both these models perform quite well for both undersampling and oversampling.

Conclusion

In our project, we have managed to train an imbalanced dataset of 2,84,807 data on credit card information. We have used different algorithms such as KNN, Decision Tree, Logistic Regression, and Random Forest. We have managed to accurately predict the target around more than 90% of the time.

We believe that by balancing the dataset and the proper evaluation metrics after training the models using the above algorithms we have managed to get a good accuracy.

In conclusion, this project showcases how we can use machine learning to solve important problems such as credit card fraud and help our financial sector flourish without this sort of distress.