Credit Card Fraud Detection

Mini Project

(Fundamentals of Machine Learning)

Submitted by:

Shriyanshi Srivastava(9919103158)

Department of CSE/IT
Jaypee Institute of Information Technology University, Noida

December 2021

Acknowledgement

I would like to place on record our deep sense of gratitude to *Dr. Mukesh Saraswat,* Associate Professor, Jaypee Institute of Information Technology, India for his generous guidance, help and useful suggestions.

I express my sincere gratitude to *Dr. Mukesh Saraswat,* Dept. of Computer Science and Engineering , India, for his stimulating guidance, continuous encouragement and supervision throughout the course of present work.

I also wish to extend our thanks to friends and other classmates for their insightful comments and constructive suggestions to improve the quality of this project work.

Abstract

The number of fraud cases has increased considerably as a result of the growing number of customers and businesses that utilise credit cards to complete financial transactions. The problem has been exacerbated by dealing with noisy and unbalanced data, as well as outliers. Credit card fraud arising from system abuse is described as the theft or misuse of a cardholder's credit card information for personal advantage without the cardholder's permission. It is critical to examine a user's usage patterns throughout the course of previous transactions in order to detect such frauds. The physical loss of a credit card or the loss of sensitive credit card information is referred to as credit card fraud. For detection, a variety of machine learning techniques can be applied. The use of artificial intelligence to detect fraud is proposed in this paper.

The proposed system uses logistic regression to build the classifier to prevent frauds in credit card transactions. To handle dirty data and to ensure a high degree of detection accuracy, a pre-processing step is used. The pre-processing step uses two novel main methods to clean the data: the mean-based method and the clustering-based method. Compared to two well-known classifiers, the support vector machine classifier and voting classifier, the proposed classifier shows better results in terms of accuracy, sensitivity, and error rate.

Index

	Page No.
1. Introduction	4
2. Background Study	5
3. Algorithm	7
4. Implementation	10
5. Experimental Result	19
6. Conclusion	22
7. References	23

1. Introduction

1.1 Problem Statement: To develop an effective fraud detection system to minimize the number of cases of fraud.Credit card fraud is the fraudulent use of credit card details to buy a product or service.

1.2 Objective: To recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase using Logistic Regression.

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase. Credit card fraud is often outlined because of the bootleg use of any system or criminal activity through the utilization of physical card or card info while not the information of the cardholder. The MasterCard is also physical or virtual in an exceedingly physical-card, the cardholder presents his or her card physically to a business person for creating a payment. To hold out deceitful transactions during this quiet purchase, a wrongdoer needs to steal the MasterCard.Machine learning algorithms are employed to analyse 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.

The investigators provide feedback to the automated system which is used to train and update the algorithm to eventually improve the fraud-detection performance over time. Fraud detection methods are continuously developed to defend criminals in adapting to their fraudulent strategies. The recent increase in credit card fraud has directly hit the financial sector hard. Losses due to credit card fraud mainly impact merchants because they bear all expenses, including the fees from their card issuer, administrative fees and other charges. All the losses are borne by the merchants, leading to increases in the prices of goods and decreases in discounts. Hence, reducing this loss is highly important. An effective fraud detection system is required to minimize the number of cases of fraud

2.Background Study

2.1 Prepare Data for Logistic Regression

The assumptions made by logistic regression about the distribution and relationships in our data are much the same as the assumptions made in linear regression. Much study has gone into defining these assumptions and precise probabilistic and statistical language is used.

2.1.1 Binary Output Variable: Logistic regression is intended for binary (two-class) classification problems. It will predict the probability of an instance belonging to the default class, which can be snapped into a 0 or 1 classification.

2.1.2 Remove Noise: Logistic regression assumes no error in the output variable (y), consider removing outliers and possibly misclassified instances from the training data. Gaussian distribution: Logistic regression is a linear algorithm. It does assume a linear relationship between the input variables with the output. Data transforms of input variables that better expose this linear relationship can result in a more accurate model. For example, we can use log, root, Box-Cox and other univariate transforms to better expose this relationship.

2.1.3 Remove Correlated Inputs: Like linear regression, the model can over fit if you have multiple highly correlated inputs. Consider calculating the pairwise correlations between all inputs and removing highly correlated inputs. Fail to Converge: It is possible for the expected likelihood estimation process that learns the coefficients to fail to converge. This can happen if there are many highly correlated inputs in your data or the data is very sparse (e.g. lots of zeros in your input data).

2.2 Logistic Function

Logistic regression is named for the function used at the core of the method, the logistic function. The logistic function, also called the sigmoid function, was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits. 1 / (1 + e^-value) Where e is the base of the natural logarithms (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that we want to transform. Below is a plot of the numbers between -10 and 10 transformed into the range 0 and 1 using the logistic function.

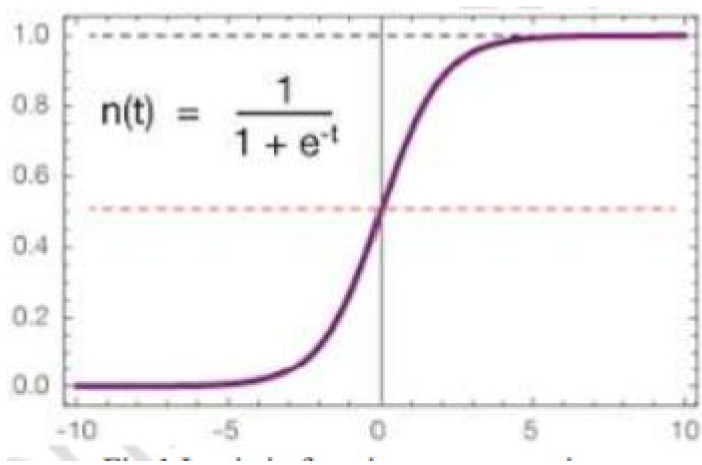


Fig 1. Logistic function representation

Fig 2. General Scenario Of Online Fraud

3.Algorithm

3.1 Building the Classifier

In the context of building the classifier, logistic regression is employed. Logistic regression is more advanced than linear regression. The reason for this is that linear regression cannot classify data that are widely distributed in a given space**.**

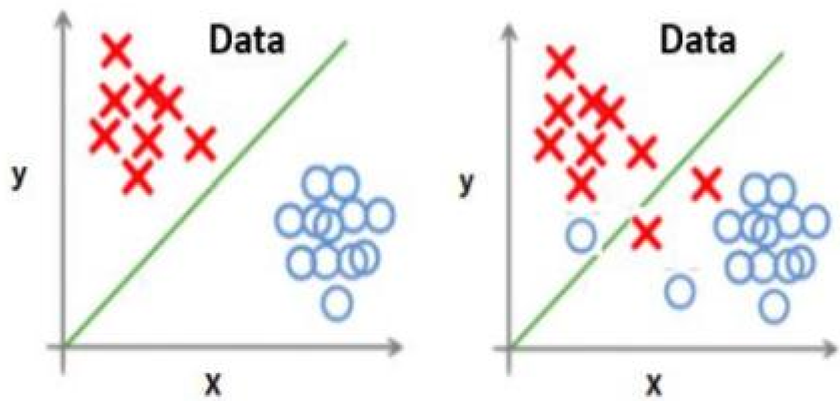


Fig 3. Limitations Of Linear Regression

As shown in Fig. 3, on the left side, the linear regression has the ability to classify the data, where the line can divide the given data into two main categories (or classes). The right side of Fig. 3 illustrates the limitation of linear regression. When the data overlap, the line cannot divide the data into two clear classes. This limitation is overcome by logistic regression. Fig. 4 provides a visual comparison between the linear regression and the logistic regression methods for the purpose of highlighting this limitation.

Logistic regression has the following advantages :

1) Logistic regression is easier to implement than linear regression and is very efficient to train.

2) It makes no assumptions about the distributions of classes in the feature space.

3) It can easily be extended to multiple classes (multinomial regression).

4) It is very efficient for classifying unknown records. The logistic regression equation can be obtained from the linear regression equation.

The mathematical steps to obtain logistic regression equations are given below: The equation of the straight line can be written as:

$y = a0 + a1 \times x1 + a2 \times x2 + \cdots ak \times xk$ (1)

In logistic regression, y can be between 0 and 1 only, so we divide the above equation by (1 − y):

y 1 - y | 0 for y = 0 and infinity for y = 1 (2)

As a result, the logistic regression equation is defined as:

$\log [y\ 1{-}y] = a0 + a1 \times x1 + a2 \times x2 + \cdots ak \times xk$ (3)

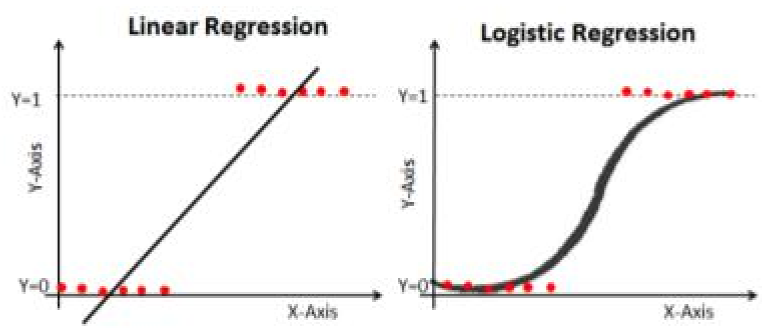


Fig 4. A Visual Comparison between Linear and Logistic Regression

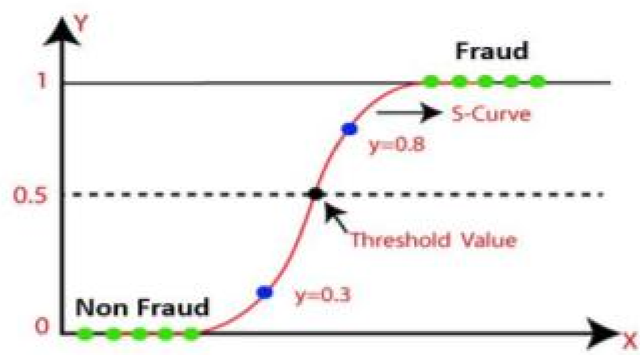


Fig 5. The Concept of Logistic Regression Classification

In other words, the fraud class takes the value "1", while the non-fraud class takes the value "0". A threshold of 0.5 is used to differentiate between the two classes, as shown in Fig. 5

3.2 Testing the Classifier

Since the cross-validation method divides the database into 10 parts, there are 10 testing data sets. Each testing data set is used to test one classifier (there are 10 classifiers). This in turn gives the model an advantage by allowing it to use the whole database for testing as well as for training. The testing process is tightly coupled with the accuracy of the model. Calculating the final accuracy involves calculating the accuracy of each classifier. Formally, let Acck C denote the accuracy of a given trained classifier, as shown in Fig. 6. Then, the final accuracy of the final classifier (ACCF C) is obtained based on the “average” mathematical operation.

$$ACC_F^C = \frac{\sum_{k=1}^{10} Acc_k^C}{k}$$

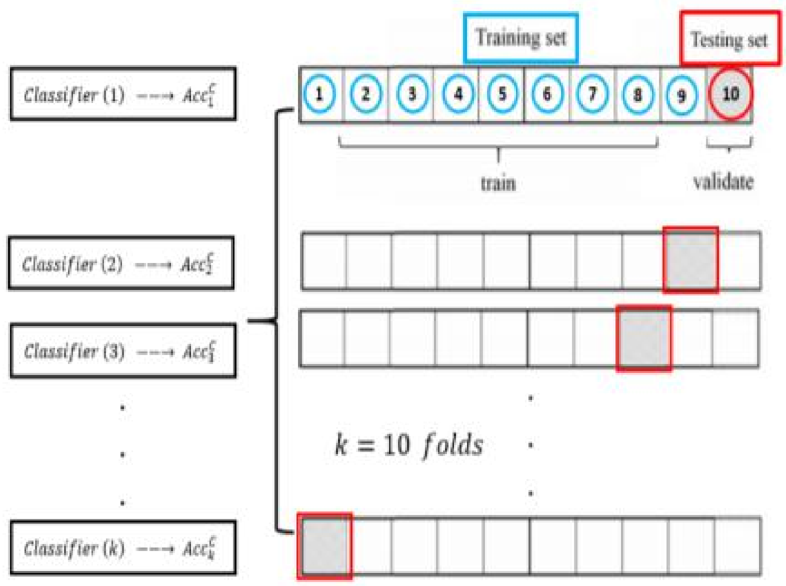


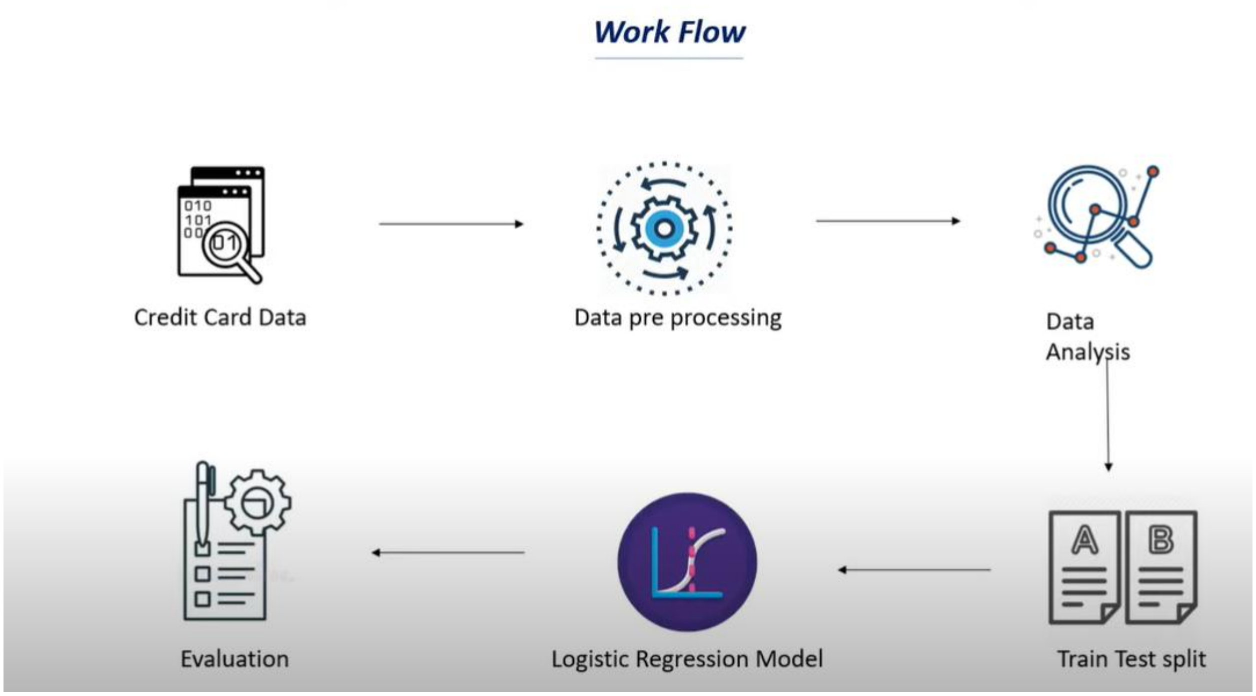
Fig 6. Classifiers with Corresponding Accuracies

3.3 Evaluating the Classifier

In general, a confusion matrix is an effective benchmark for analysing how well a classifier can recognize records of different classes [34]. The confusion matrix is formed based on the following terms: 1) True positives (TP): positive records that are correctly labelled by the classifier. 2) True negatives (TN): negative records that are correctly labelled by the classifier. 3) False positives (FP): negative records that are incorrectly labelled positive. 4) False negatives (FN): positive records that are mislabelled negative. Table III shows the confusion matrix in terms of the TP, FN, FP, and TN values. Relying on the confusion matrix, the accuracy, sensitivity, and error rate metrics are derived. For a given classifier, the accuracy can be calculated by considering the recognition rate, which is the percentage of records in the test set that are correctly classified (fraudulent or non-fraudulent). The accuracy is defined as:

$$Accuracy = (TP+TN) / number\ of\ all\ records\ in\ the\ testing\ set \quad (5).$$

4. Implementation



Below is the code implementation of the whole idea that is used in this project.

Importing the Dependencies

```
[ ]  import numpy as np
     import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import accuracy_score
```

```
[ ]  # loading the dataset to a Pandas DataFrame
     credit_card_data = pd.read_csv('/content/credit_data.csv')
```

```
[ ]  # first 5 rows of the dataset
     credit_card_data.head()
```

```
[ ] credit_card_data.tail()
```

	Time	V1	V2	V3	V4	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.3
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.8
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.6
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.3
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.0

```
[ ] # dataset informations
    credit_card_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 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
```

```
[ ]  <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 284807 entries, 0 to 284806
     Data columns (total 31 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
      20  V20     284807 non-null  float64
      21  V21     284807 non-null  float64
      22  V22     284807 non-null  float64
      23  V23     284807 non-null  float64
      24  V24     284807 non-null  float64
      25  V25     284807 non-null  float64
      26  V26     284807 non-null  float64
      27  V27     284807 non-null  float64
      28  V28     284807 non-null  float64
      29  Amount  284807 non-null  float64
      30  Class   284807 non-null  int64
     dtypes: float64(30), int64(1)
```

```
[ ] # checking the number of missing values in each column
    credit_card_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
V24       0
V25       0
V26       0
V27       0
```

```
[ ] # distribution of legit transactions & fraudulent transacti
    credit_card_data['Class'].value_counts()
```

```
0    284315
1       492
Name: Class, dtype: int64
```

This Dataset is highly unblanced

0 --> Normal Transaction

1 --> fraudulent transaction

```
[ ] # separating the data for analysis
    legit = credit_card_data[credit_card_data.Class == 0]
    fraud = credit_card_data[credit_card_data.Class == 1]
```

```
[ ] print(legit.shape)
    print(fraud.shape)
```

```
(284315, 31)
(492, 31)
```

```
[ ] # statistical measures of the data
    legit.Amount.describe()
```

```
count    284315.000000
mean         88.291022
std         250.105092
```

```
[ ] # statistical measures of the data
    legit.Amount.describe()
```

```
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
```

```
[ ] fraud.Amount.describe()
```

```
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
```

```
[ ] # compare the values for both transactions
    credit_card_data.groupby('Class').mean()
```

	Time	V1	V2	V3	
Class					
0	94838.202258	0.008258	-0.006271	0.012171	-0.

Under-Sampling

Build a sample dataset containing similar distribution of normal transactions and Fraudulent Transactions

Number of Fraudulent Transactions --> 492

```
[ ] legit_sample = legit.sample(n=492)
```

Concatenating two DataFrames

```
[ ] new_dataset = pd.concat([legit_sample, fraud], axis=0)
```

```
[ ] new_dataset.head()
```

```
[ ] new_dataset.tail()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850	0.697211	-2.0
280143	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170	0.248525	-1.1
280149	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739	1.210158	-0.6
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002	1.058733	-1.6
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050	-0.068384	0.5

```
[ ] new_dataset['Class'].value_counts()
```

```
1    492
0    492
Name: Class, dtype: int64
```

```
[ ] new_dataset.groupby('Class').mean()
```

Class	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	96783.638211	-0.053037	0.055150	-0.036786	-0.046439	0.077614	-0.023218	-0.000703	-0.057620
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731	0.570636

Splitting the data into Features & Targets

```
[ ] X = new_dataset.drop(columns='Class', axis=1)
    Y = new_dataset['Class']
```

```
[ ] print(X)
```

```
             Time        V1        V2  ...       V27       V28  Amount
203131  134666.0 -1.220220 -1.729458  ...  0.173995 -0.023852  155.00
95383    65279.0 -1.295124  0.157326  ...  0.317321  0.105345   70.00
99706    67246.0 -1.481168  1.226490  ... -0.546577  0.076538   40.14
153895  100541.0 -0.181013  1.395877  ... -0.229857 -0.329608  137.04
249976  154664.0  0.475977 -0.573662  ...  0.058961  0.012816   19.60
...          ...       ...       ...  ...       ...       ...     ...
279863  169142.0 -1.927883  1.125653  ...  0.292680  0.147968  390.00
280143  169347.0  1.378559  1.289381  ...  0.389152  0.186637    0.76
280149  169351.0 -0.676143  1.126366  ...  0.385107  0.194361   77.89
281144  169966.0 -3.113832  0.585864  ...  0.884876 -0.253700  245.00
281674  170348.0  1.991976  0.158476  ...  0.002988 -0.015309   42.53

[984 rows x 30 columns]
```

```
[ ] print(Y)
```

```
203131    0
95383     0
99706     0
153895    0
249976    0
         ..
```

Split the data into Training data & Testing Data

```
[ ] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
```

```
[ ] print(X.shape, X_train.shape, X_test.shape)
```

```
(984, 30) (787, 30) (197, 30)
```

Model Training

Logistic Regression

```
[ ] model = LogisticRegression()
```

```
[ ] # training the Logistic Regression Model with Training Data
    model.fit(X_train, Y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Model Evaluation

Accuracy Score

```
[ ] # accuracy on training data
    X_train_prediction = model.predict(X_train)
    training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
[ ] print('Accuracy on Training data : ', training_data_accuracy)
```

```
Accuracy on Training data :  0.9415501905972046
```

```
[ ] # accuracy on test data
    X_test_prediction = model.predict(X_test)
    test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
[ ] print('Accuracy score on Test Data : ', test_data_accuracy)
```

```
Accuracy score on Test Data :  0.9390862944162437
```

5.Experimental Result

Two classifiers are selected for a comparison with the classifier proposed in this work. They are the K-nearest neighbours (KNN) classifier and the voting classifier (VC). Below, a brief description of each selected classifier is presented. Fig. 7 shows the fundamental steps required to build the voting classifier.

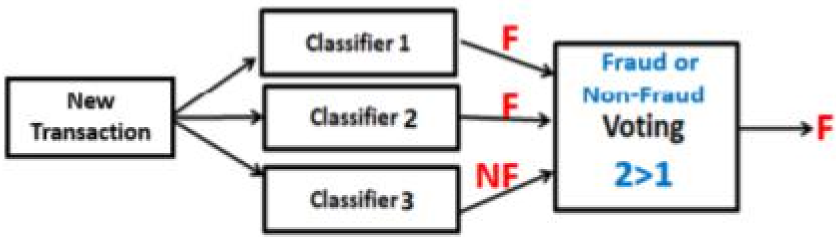


Fig 7. Basic Concept of the Voting Classifier

As shown in Fig. 7, there are many classifiers, and a voting step is required to produce the final output class. The voting step means that the final output of the classifier depends on the majority of the classes (predictions) that are generated by the classifiers. For example, there are three classifiers in Fig. 8. The final prediction is either Fraud (F) or Non-Fraud (NF). The voting process works as follows: 1) Obtain the outputs of the classifiers. 2) Calculate the number of classifiers that generate the F class (let us say 2 classifiers). 3) Calculate the number of classifiers that generate the NF class (let us say 1 classifier). 4) The majority is 2. Therefore, the final prediction is the F class. Fig. 8 shows the fundamental steps for building the KNN classifier. As shown in Fig. 8, there are two clusters (one for fraudulent transactions and one for non-fraudulent transactions). Each cluster has a centre, which is represented numerically by (-1) for non fraudulent transactions and (+1) for fraudulent transactions. For a given transaction, the KNN classifier processes the transaction and generates a corresponding number. Then, the distance between the generated value and the centre of each cluster is calculated. Finally, the transaction is assigned to the correct cluster (in the example, it is assigned to the non-fraud cluster). C. Results Since the cross-validation method is used to divide the database, we obtain ten sub-classifiers as mentioned previously. The process of calculating the final values of the AI-based metrics depends on the “average" mathematical operation. Table IV summarizes the obtained results. Table V summarizes the comparison of the logistic regression (LogR)-based classifier with both the KNN-based classifier and the VC-based classifier.

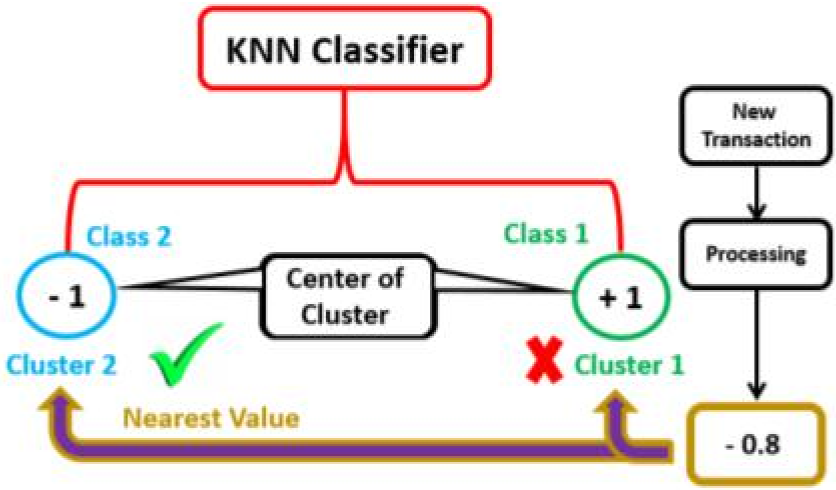


Fig 8. Basic Concept of the KNN Classifier

Table 1: Evaluating the proposed classifier

K-value	Accuracy	Sensitivity	Error rate
1	96%	97%	4%
2	98%	96%	2%
3	98%	97%	2%
4	96%	96%	4%
5	97%	98%	3%
6	96%	98%	4%
7	97%	96%	3%
8	98%	98%	2%
9	98%	98%	2%
10	98%	96%	2%
Average	97.2%	97%	2.8%

Table 2: Comparison of the classifiers

Classifier	Metrics		
	Accuracy	*Sensitivity*	*Error rate*
LogR classifier	97.2%	97%	2.8%
KNN classifier	93%	94%	7%
VC classifier	90%	88%	10%

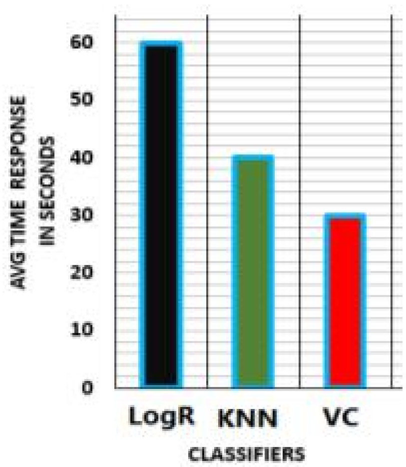


Fig 9. Performances of the Three Classifiers.

5.1 Analysis

Fig. 9 shows that the VC classifier achieves the best performance. This is because it depends only on a simple mathematical operation (the sum operation) to determine the classes and generate the final output. The KNN classifier comes second in terms of its response time. That is because this classifier must perform additional mathematical operations related to calculating the distances between the new value and the centre of each cluster, and these operations in turn consumes more time. Compared to the previous classifiers, the LogR classifier performs the worst. The reason for this is that the time required for database division and training the sub-classifiers is very high. In other words, training and testing ten sub-classifiers logically takes less time than training and testing one classifier (i.e., the KNN and VC classifiers). However, although the response time of the LogR classifier is the longest, it achieves the best accuracy. From the point of view of detecting fraud (or security), accuracy is more of a concern than performance. This issue will be taken into consideration in future work.

6. Conclusion

The detection of credit card fraud is a vital research field. This is because of the increasing number of fraud cases in financial institutions. This issue opens the door for employing artificial intelligence to build systems that can detect fraud. Building an AI-based system to detect fraud requires a database to train the system (or classifier). The data in reality are dirty and have missing values, noisy data, and outliers. Such issues negatively affect the accuracy rate of the system. To overcome these problems, a logistic regression-based classifier is proposed. The data are first cleaned using two methods: the mean-based method and clustering-based method. Second, the classifier is trained based on the cross validation technique (folds=10), which ensures that the whole database is used as both the training data set and testing data set. Finally, the proposed classifier is evaluated based on the accuracy, sensitivity, and error rate metrics. The proposed logistic regression-based classifier is compared to well-known classifiers, which are the K-nearest neighbours classifier and the voting classifier. The logistic regression-based classifier generates the best results (accuracy = 97.2%, sensitivity = 97%, and error rate = 2.8%).

6.1 Limitations

The performance of the proposed classifier suffers in terms of response time. In addition, it does not apply to data in real time.

6.2 Future work

In future work, we intend to enhance the performance and take the security and privacy of the data in real time into consideration.

7. References

[1] Yousefi, Niloofar, Marie Alaghband, and Ivan Garibay. "A Comprehensive Survey on Machine Learning Techniques and User Authentication Approaches for Credit Card Fraud Detection." arXiv preprint arXiv:1912.02629 (2019).

[2] Paschen, Jeannette, Jan Kietzmann, and Tim Christian Kietzmann. "Artificial intelligence (AI) and its implications for market knowledge in B2B marketing." Journal of Business & Industrial Marketing (2019).

[3] Abdallah, Aisha, Mohd Aizaini Maarof, and Anazida Zainal. "Fraud detection system: A survey." Journal of Network and Computer Applications 68 (2016): 90-113.

[4] Alladi, Tejasvi, et al. "Consumer IoT: Security vulnerability case studies and solutions." IEEE Consumer Electronics Magazine 9.2 (2020): 17-25.