Analysis of Fraud detection in banks using Machine Learning Algorithms

A PROJECT REPORT

Submitted by

M RAMYA SREE	**BL.EN.U4AIE21072**
M SIDDHARTHA	**BL.EN.U4AIE21080**
POLI VAMSI VARDHAN	**BL.EN.U4AIE21101**

for the course

21AIE205- Python for Machine Learning

Guided and Evaluated by

Ms. Niharika Panda

Dept. of CSE,

AMRITA SCHOOL OF ENGINEERING, BANGALORE

AMRITA VISHWA VIDHYAPEETHAM

BANGALORE-560035

January 2023

ABSTRACT

The purpose of this paper is to examine the commonly employed supervised methods for detecting fraud and evaluate their effectiveness using real-world data. The study aims to demonstrate the application of these techniques and to develop an ensemble model as a potential solution to the problem of fraud detection. Fraudulent activities have been a persistent problem across various industries, including banking, medical, and insurance. With the rise of online transactions through various payment methods such as credit/debit cards, mobile wallets, and digital payment platforms, the risk of fraud has also increased. Criminals have become increasingly sophisticated in exploiting vulnerabilities in systems, making it challenging to develop effective methods for authenticating transactions and protecting customers from fraud. To combat this issue, fraud detection algorithms have been developed to detect and prevent fraudulent activities. In this project, we propose to use machine learning algorithms to automatically detect bank fraud. Specifically, we will use the K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree, Random Forest algorithms and Logistic regression to detect fraudulent transactions in a dataset of bank transactions.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER-1

INTRODUCTION

Fraud detection in banks is the process of identifying and preventing fraudulent activity in the financial sector. Banks use a variety of methods to detect fraud, including machine learning algorithms, manual reviews, and fraud alerts. Some common types of fraud that banks may detect include credit card fraud, check fraud, and money laundering.One of the key challenges in fraud detection is accurately identifying fraudulent activity without generating false positives, which can lead to unnecessary additional work for bank staff and potentially cause inconvenience for customers. To mitigate this risk, banks may use a combination of automated and manual processes to review and confirm suspected fraudulent activity. Machine learning is a type of artificial intelligence that involves training algorithms on data to recognize patterns and make decisions without being explicitly programmed to do so. In the context of fraud detection in banks, machine learning algorithms can be used to analyse data on customer transactions, account activity, and other relevant information to identify patterns that are typical of fraudulent activity.

1. Machine Learning

Machine Learning (ML) is a branch of Artificial Intelligence (AI) that mainly focuses on data usage and algorithms to imitate humans by gradually improving accuracy. Machine Learning can be used for various purposes like Image Recognition, Image Classification, Speech Recognition, Medical diagnosis, and Predictive analytics. There are four predominant types of Machine Learning algorithms, Supervised, Unsupervised, Semi-Supervised, and Reinforcement.

1.1 Supervised Learning

Supervised Learning is a type of pre-trained algorithm where the input data is labeled for the desired output with greater accuracy. The training dataset includes inputs along with their expected outputs. Some examples of Supervised Learning are Linear Regression, Logistic Regression, KNN, SVM, and Naïve Bayes.

1.2 Unsupervised Learning

Unsupervised Learning is used to find similarities in data sets and the input is not classified nor labeled. These algorithms are suitable for the problems in which the outputs are not available. Some examples of Unsupervised Learning are K-Means Clustering, Principal Component Analysis, and Hierarchical Clustering

1.3 Semi-Supervised Learning

Semi-Supervised Learning is partially supervised and partially unsupervised where only some of the input data are labeled or have output, but a large portion of data does not have labels or outputs. A good example of Semi-Supervised Learning is a Text document classifier.

1.4 Reinforcement Learning

Reinforcement Learning systems do not have a training dataset rather they are trained to learn from previous experiences. They are efficient in taking suitable decisions based on previous learnings. Some common applications of Reinforcement based learning are Self-Driving Vehicles, Industrial Automation, and Healthcare.

Some of the ways that machine learning can be used for fraud detection in banks include:

2. K-nearest neighbors (KNN)

K-Nearest Neighbor (KNN) is a Supervised Learning algorithm. KNN algorithm checks the similarity between two cases and puts that into a similar available category. KNN algorithm initially stores all the cases and classifies them based on the similarities. Later it classifies the new cases to the suitable category. KNN algorithm is used for both Regression and Classification problems. KNN algorithm is a non-Parametric learning algorithm and lazy learning algorithm as well. It is non-parametric as does not assume anything about the available data and it is lazy learning because it does not learn from the training data and takes time for classification into categories. It just stores the data in the training phase.

3.Support vector machines (SVMs)

Support Vector Machines (SVMs) are a type of supervised learning algorithm that works by finding the hyperplane in a high-dimensional feature space that maximally separates the different classes. They are a type of machine learning algorithm that can be used for a variety of tasks, including fraud detection in banks. One way to use SVMs for fraud detection is to create a model that takes a set of transactions as input and predicts whether or not each transaction is fraudulent. To do this, the model would be trained on a dataset of known fraudulent and non-fraudulent transactions. The model would then be able to classify new transactions as either fraudulent or non-fraudulent based on how similar they are to the transactions in the training dataset

4. Decision trees

Decision trees are a type of machine learning algorithm that can be used for a variety of tasks, including fraud detection in banks. Decision Trees are a widely-used and popular supervised machine learning algorithm that can be used for both classification and regression problems. The algorithm creates a tree-like model of decisions and their possible consequences, in order to predict the value of a target variable. A decision tree starts with a single node called the root node, which represents the entire dataset. From the root node, the algorithm splits the data into subsets based on the values of the input features. Each split is represented by an internal node of the tree and leads to two or more child nodes, representing the different possible outcomes of the decision. These child nodes are further split based on the values of other features, creating a hierarchical tree-like structure. The process continues until a stopping criterion is met, such as a maximum tree depth or a minimum number of samples per leaf node. In the context of fraud detection, decision trees can be used to identify patterns in bank transactions that may indicate fraudulent activity. To use decision trees for

fraud detection is to create a model that takes a set of transactions as input and predicts whether or not each transaction is fraudulent.

5.Random forests

Random forests are a type of machine learning algorithm that can be used for classification and regression tasks. Random forests can be used for fraud detection in banks by training the algorithm on a dataset of transactions that have been labeled as either fraudulent or legitimate and using the learned patterns to classify new transactions. However, rather than using a single decision tree, random forests use an ensemble of decision trees to make predictions. This can improve the accuracy of the algorithm, as the ensemble can make use of the strengths of multiple individual decision trees to make more accurate prediction.

6. Logistic Regression

Logistic Regression is Supervised Learning algorithm used for classification problems. Logistic regression is similar to linear regression but can only be used for classification problems. Logistic Regression uses the logistic function 1/ 1 + e^(−x) to classify the given input into binary output. These outputs are only between 0 and 1 which results in a sigmoid curve. Logistic Regression is often used in predictive analysis to predict the outputs categorically. It is the relationship between dependent variables and independent variables by estimating probabilities using a logistic function. The output variables are either A or B or 0 or 1

CHAPTER-2

LITERATURE SURVEY

In [1], we have learnt the widely used supervised techniques applied for fraud detection. In addition, this aims to apply some techniques to evaluate their performance on real-world data and develop an ensemble model as a potential solution for this problem. Different techniques applied in this study for fraud detection purposes are logistic regression, decision tree, random forest, KNN, and SVM. Here the confusion matrix gives information about the assignment of inputs to the different classes. This study uses precision and recall to evaluate the performance, calculated based on the confusion matrix.

In [2], Here we learnt to implement a machine learning algorithm in detecting fraud based on historical data set in a retail consumer financing company. We have absorbed that the outcome of machine learning is used as samples for the fraud detection team. Data analysis is performed through data processing, feature selection, hold-on methods, and accuracy testing. There are five machine learning methods applied in this study: Logistic Regression, K-Nearest Neighbor (KNN), Decision Tree, Random Forest, and Support Vector Machine (SVM). Historical data are divided into two groups: training data and test data. The results show that the Random Forest algorithm has the highest accuracy with a training score of 0.994999 and a test score of 0.745437. This means that the Random Forest algorithm is the most accurate method for detecting fraud.

In [3], Here Machine learning algorithms, such as Random Forest and Logistic Regression, have been widely used in credit card fraud detection because of their ability to learn from large and complex datasets. Random Forest is a type of ensemble learning algorithm that builds multiple decision trees and combines their predictions to make a final prediction. Logistic Regression, on the other hand, is a supervised learning algorithm that is commonly used for binary classification problems, such as detecting fraud or not. From this research, we found that the Random Forest algorithm achieved the best accuracy and MCC score, which suggests that it might be the most effective method for detecting fraud in the dataset that is used. However, it's important to note that the performance of these algorithms can depend on the specific characteristics of the dataset and the problem at hand. The best algorithm for a specific task will be dependent on the dataset.

In [4], The decision tree method is a popular choice for this task, as it can be used to create a model that can make predictions based on historical data. It is important to note that while machine learning can be a powerful tool for detecting fraudulent activity and predicting loan defaults, it is not a panacea. The accuracy rate of 75.9% suggests that there is still room for improvement. It is important to use a combination of techniques for detecting fraud and loan defaults, such as incorporating domain expertise and using multiple algorithms to cross-validate results. Machine learning can be a powerful tool for detecting fraudulent activity and predicting loan defaults, but it should be used in conjunction with other techniques, proper care should be taken to ensure the data used to train the model is of high quality.

CHAPTER-3

SYSTEM MODEL

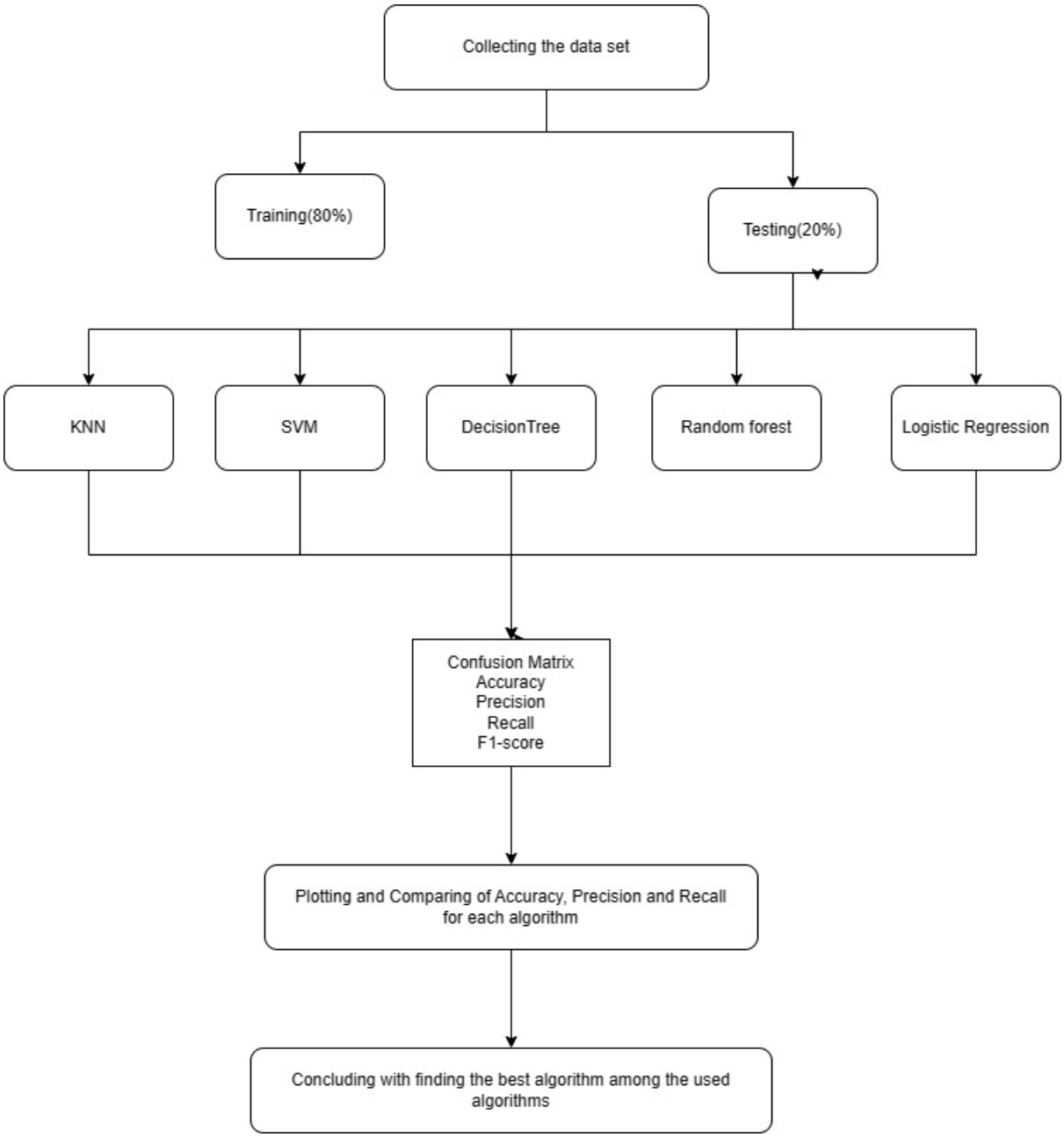

Fig-3.1

1. K-nearest neighbors (KNN)

K-Nearest Neighbors (KNN) is a classification and regression algorithm that is based on the idea of finding the K closest training examples in the feature space and using these examples to make predictions.

Here's a general outline of the KNN algorithm:

1. Choose the number of neighbors K.
2. Calculate the distance between the new example and all training examples.
3. Sort the distances in ascending order.
4. Select the K nearest neighbors.
5. Based on the classification problem, predict the class of the new example. If it's a regression problem, predict the value of the new example.
6. There are various ways to measure the distance between examples in the feature space. Commonly used distance measures include Euclidean distance, Manhattan distance, and Minkowski distance.

The main advantage of KNN is that it's simple to implement and can be used for both classification and regression. However, the algorithm can be computationally expensive and may not perform well on large datasets.

Overall, KNN is a useful tool for fraud detection in banks, and it has the potential to help banks identify and prevent fraudulent activity.

Fig-3.2

2. Support vector machines (SVM)

Support Vector Machines (SVMs) are a type of supervised learning algorithm that can be used for classification or regression tasks. The goal of an SVM is to find the hyperplane in an N-dimensional space that maximally separates the two classes.

Here's a general outline of the SVM algorithm:

1. Select a kernel and hyperparameters for the kernel.
2. Train the model using the training data.
3. Use the trained model to make predictions on new examples.

One of the main advantages of SVMs is that they can perform well even in high-dimensional spaces, or when the number of dimensions is greater than the number of samples. They are also memory efficient, because they only use a subset of the training examples (called support vectors) to make predictions.

However, SVMs can be sensitive to the selection of kernel hyperparameters and may not perform well if the data is highly imbalanced. They also do not scale well to very large datasets.

Overall, SVMs are a useful tool for fraud detection in banks, and they have the potential to help banks identify and prevent fraudulent activity.

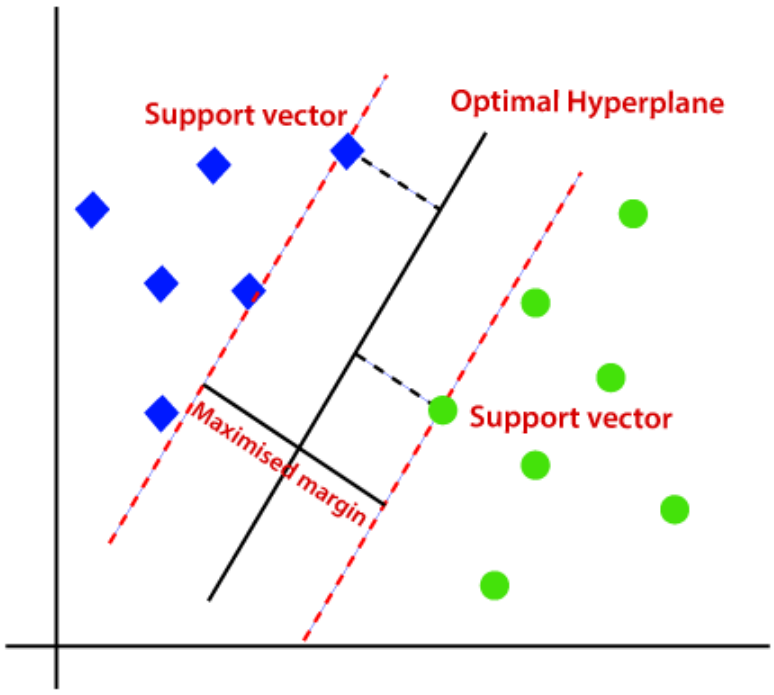

Fig-3.3

3. Decision trees

Decision Trees are a type of supervised learning algorithm that can be used for classification or regression tasks. The goal of a Decision Tree is to create a model that predicts the value of a target variable based on several input variables.

Here's a general outline of the Decision Tree algorithm:

1. Select the input variable and split point that results in the greatest information gain.
2. Split the data into two subsets based on the split point.
3. Repeat the process for each subset, selecting the input variable and split point that results in the greatest information gain.
4. Continue the process until the tree is fully grown or a stopping criteria is reached.

One of the main advantages of Decision Trees is that they are easy to understand and interpret, because they follow a "tree" structure. They are also relatively fast to train and make predictions.

However, Decision Trees can be prone to overfitting, especially if they are allowed to grow too deep. They also may not perform well on datasets with many features or if the features are highly correlated.
Overall, decision trees are a useful tool for fraud detection in banks, and they have the potential to help banks identify and prevent fraudulent activity.

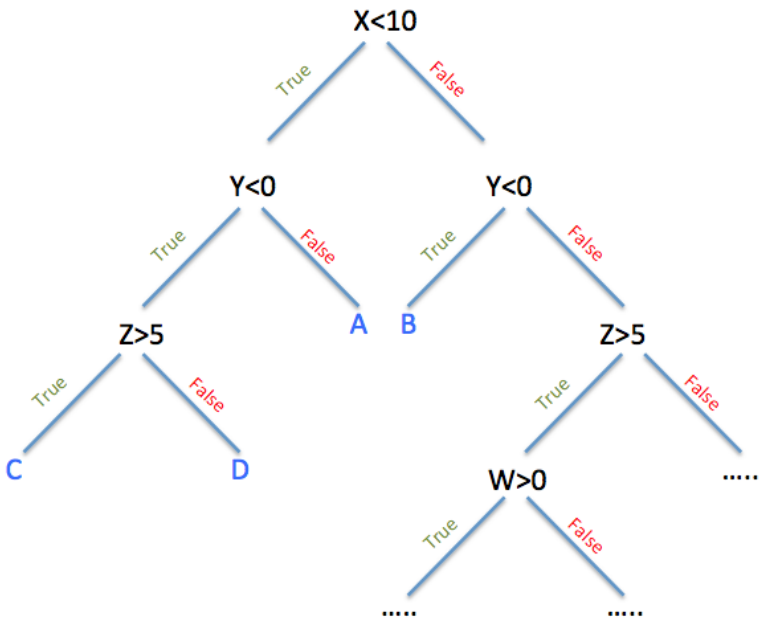

Fig-3.4

4. Random forests:

Random Forests are an ensemble learning method for classification and regression that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees.

Here's a general outline of the Random Forest algorithm:

- Select a number of decision trees to build.
- For each tree:
  1. Select a random sample of the data with replacement (bootstrapping).
  2. Select a random subset of the features.
  3. Build a decision tree using the bootstrapped data and selected features.
- For classification tasks, output the class that is the mode of the classes predicted by the individual trees. For regression tasks, output the mean prediction of the individual trees.

One of the main advantages of Random Forests is that they can reduce overfitting, because they are constructed using multiple decision trees, which are trained on different subsets of the data and selected features. They are also relatively fast to train and make predictions.
However, Random Forests can be sensitive to the number of trees and the number of features selected for each tree. They also may not perform well on highly imbalanced datasets.
Overall, random forests are a useful tool for fraud detection in banks, and they have the potential to help banks identify and prevent fraudulent activity.

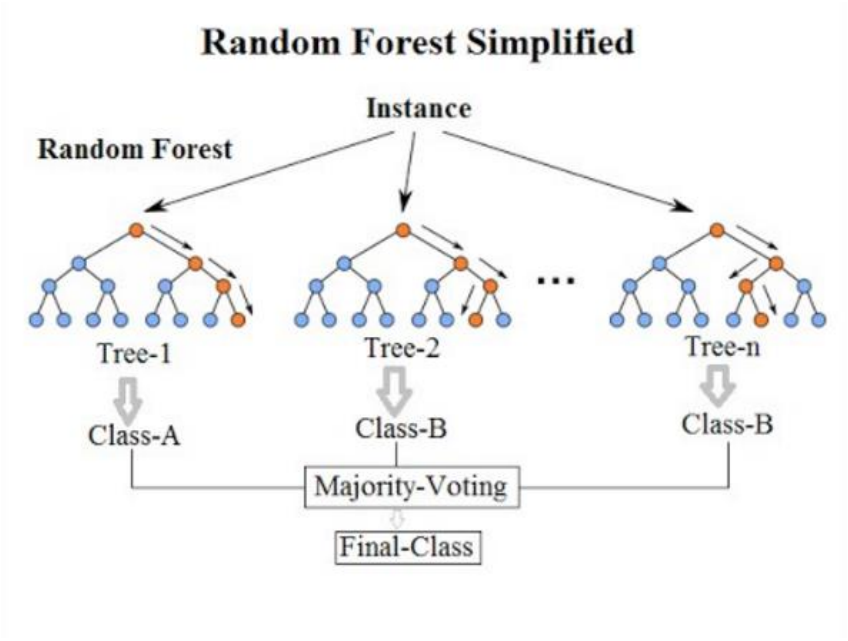

Fig-3.5

5. Logistic regression

Logistic Regression is a classification algorithm that is used to predict a binary outcome (e.g., 0 or 1, yes or no). It works by using a linear combination of the input features to make a prediction, and then applying a sigmoid function to map the predicted value to a probability between 0 and 1.

Here's a general outline of the Logistic Regression algorithm:

1. Collect and prepare the training data.
2. Initialize the model parameters.
3. Make predictions using the current model parameters.
4. Calculate the error between the predicted probabilities and the true labels.
5. Update the model parameters using an optimization algorithm (such as gradient descent) to minimize the error.
6. Repeat steps 3-5 until the model converges (i.e., the error is minimized or no longer improving).
7. Use the trained model to make predictions on new examples.

One of the main advantages of Logistic Regression is that it is a simple and fast algorithm that can be trained using an optimization algorithm such as gradient descent. It can also handle large amounts of data and high-dimensional feature spaces.

However, Logistic Regression may not perform well if the data is not linearly separable or if there are a large number of categorical features. It is also sensitive to the scale of the input features.

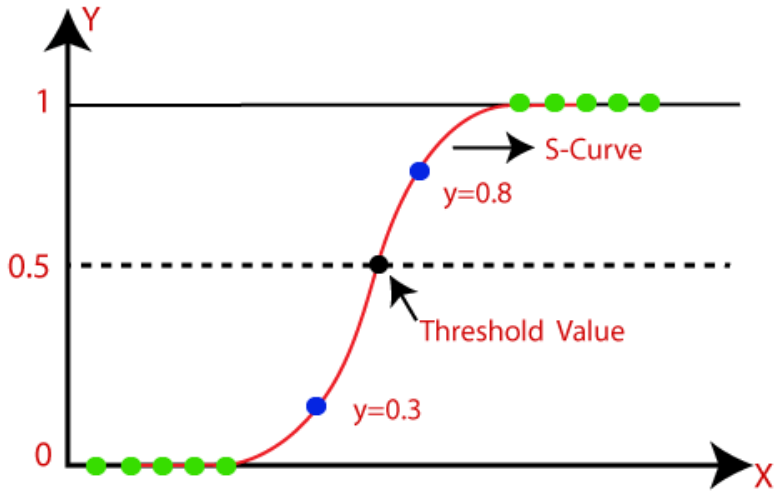

Fig-3.6

CHAPTER-4

IMPLEMENTATION

Dataset for this project is **fraud_detection_bank_dataset.csv** is taken kaggle. For this model creation we have used Anaconda Navigator. In that Jupyter Notebook is used to create the model.Jupyter Notebook allows users to compile all aspects of a data project in one place making it easier to show the entire process of a project to your intended audience.

Library that are used:

1) Sklearn.model_selection
2) Sklearn.preprocessing
3) Sklearn.metrics
4) Matplotlib.pyplot
5) Sklearn.neighbors
6) Sklearn.svm
7) Sklearn.tree
8) sklearn.ensemble
9) sklearn.linear_model

Short Description:

1)Sklearn.model_selection:

sklearn. model_selection .train_test_split. Split arrays or matrices into

random train and test subsets.

2)Sklearn.preprocessing:

The sklearn. preprocessing package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators. In general, learning algorithms benefit from standardization of the data set.

3)Sklearn.metrics:

The sklearn. metrics module implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values.

4)Matplotlib.pyplot:

Matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

5)Sklearn.neighbors:

sklearn. neighbors provides functionality for unsupervised and supervised neighbors-based learning methods. Unsupervised nearest neighbors is the foundation of many other learning methods, notably manifold learning and spectral clustering.

6)Sklearn.svm:

SVM is an exciting algorithm and the concepts are relatively simple. The classifier separates data points using a hyperplane with the largest amount of margin.

7)Sklearn.tree:

sklearn.tree .DecisionTreeClassifier· The function to measure the quality of a split. · The strategy used to choose the split at each node.

8) sklearn.ensemble:

sklearn.ensemble RandomForestClassifier
ensemble is a module in the Python library scikit-learn (also known as sklearn) that contains a number of methods for combining the predictions of multiple machine learning models.
RandomForestClassifier: An ensemble of decision trees trained with bagging (bootstrapped sampling with replacement) and a random subset of features for each tree.

9) sklearn.linear_model

sklearn.linear_model LogisticRegression

linear_model is a module in the Python library scikit-learn (also known as sklearn) that contains a number of methods for fitting linear models, which are models that make predictions based on a linear combination of the input features.

- **LinearRegression**: A linear model for continuous target variables.
- **LogisticRegression**: A linear model for binary and multiclass classification.

Functions Used:

```
def disp_confMatrix():
    cm = confusion_matrix(y_test, y_test_predicted)
    cmp = ConfusionMatrixDisplay(cm, display_labels=target_names)
    fig, ax = plt.subplots(figsize=(7,7))
    cmp.plot(ax=ax)
    plt.show()

    return None
```

The disp_confMatrix function you have defined appears to be a function for displaying a confusion matrix for a set of test predictions. A confusion matrix is a matrix that is used to visualize the performance of a classification model. It is often used to evaluate the accuracy of the model and to identify the classes that the model is having difficulty predicting.

The confusion matrix is constructed by comparing the predicted class labels with the true class labels for a set of data. The rows of the matrix correspond to the true classes, and the columns correspond to the predicted classes. Each element of the matrix shows the number of samples that were predicted to belong to a particular class but were actually in another class.

CHAPTER-5

RESULTS

CODE:

https://drive.google.com/file/d/1wybM4MfH6MWoDO7Y5dPRLab4vsQW4_Hh/view?usp=share_link

SAMPLE CODE & SAMPLE OUTPUT

```
data = pd.read_csv(r"C:\Users\himas\Downloads\fraud_detection_bank_dataset.csv\fraud_detection_bank_dataset.csv")
data
```

	Unnamed: 0	col_0	col_1	col_2	col_3	col_4	col_5	col_6	col_7	col_8	...	col_103	col_104	col_105	col_106	col_107	col_108	col_109	col_110	co
0	0	9	1354	0	18	0	1	7	9	0	...	0	0	0	1	1	0	0	0	
1	1	0	239	0	1	0	1	0	0	0	...	0	1	0	0	0	0	0	0	
2	2	0	260	0	4	0	3	6	0	0	...	0	0	0	1	1	0	0	0	
3	3	17	682	0	1	0	0	8	17	0	...	0	1	0	1	1	0	0	0	
4	4	1	540	0	2	0	1	7	1	0	...	0	0	0	1	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
20463	20463	0	88	0	0	0	2	-1	0	0	...	0	1	0	0	0	0	1	0	
20464	20464	0	134	0	2	0	0	6	0	0	...	0	0	0	0	0	0	0	0	
20465	20465	4	393	1	1	0	0	-1	4	0	...	0	1	0	0	0	0	0	0	
20466	20466	0	10	0	1	0	0	-1	0	0	...	0	0	0	0	0	0	0	0	
20467	20467	4	399	0	3	0	1	7	4	0	...	0	1	0	1	1	0	0	0	

Downloading and Reading the Dataset

```
# Remove the 1st column (Unnamed).
data = data.iloc[:,1:]

# Rename "targets" column to "is_fraud".
data.rename(columns={'targets': 'is_fraud'}, inplace=True)

# Check for any null values in each column
print(f'Null Values: {sum(data.isnull().sum())}')
```

```
Null Values: 0
```

```
# Obtain the number of fraud & non-fraud objects.
class_balance = data.is_fraud.value_counts()
fraud_percentage = ((class_balance[1]/len(data))*100)

print(f'Non-Fraud: {class_balance[0]} '
      f'({(100-fraud_percentage).__round__(4)}%)')
print(f'Fraud: {class_balance[1]} '
      f'({fraud_percentage.__round__(4)}%)')
```

```
Non-Fraud: 15030 (73.4317%)
Fraud: 5438 (26.5683%)
```

```
test_data_fraction = 0.2 # Allocate 20% of the data for testing.
random_seed = 20

X = data.iloc[:,0:-1] # Features (inputs).
y = data.is_fraud # Target (output).

# Verify all features are numerical
print(f'Feature Datatypes\n-----------------\n'
      f'{X.dtypes.value_counts()}\n')

# Split data into training (80%) and testing (20%).
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_data_fraction,
                                                    random_state=random_seed)
# Obtain shape of training and testing data.
print('Training Shape\n--------------')
print(f'Input: {X_train.shape}')
print(f'Output: {y_train.shape}')

print('\nTesting Shape\n--------------')
print(f'Input: {X_test.shape}')
print(f'Output: {y_test.shape}')
```

```
Feature Datatypes
-----------------
int64      111
float64      1
dtype: int64

Training Shape
--------------
Input: (16374, 112)
Output: (16374,)

Testing Shape
--------------
Input: (4094, 112)
Output: (4094,)
```

This code splits a dataset into training and testing sets, where the training set is used to fit a machine learning model and the testing set is used to evaluate the model's performance.

The input data X consists of all columns in data except the last column, which is assumed to be the target or output. The target data y consists of the last column of data, which is assumed to be a binary classification of whether a transaction is fraudulent or not.

The train_test_split() function from the sklearn.model_selection module is used to split the input and target data into training and testing sets. The test_size parameter specifies the fraction of the data that should be allocated for testing (in this case, 20%). The random_state parameter specifies the random seed used for shuffling the data before splitting it into the training and testing sets.

After the data is split, the shapes of the training and testing sets are printed. This can be useful for verifying that the data has been split as intended and to get an idea of the size of the datasets.

DECISION TREE

```
: #decision tree
from sklearn.tree import DecisionTreeClassifier
model_d = DecisionTreeClassifier()
model_d = model_d.fit(X_train,y_train)
y_test_predicted1= model_d.predict(X_test)

#Evaluation using Accuracy score
from sklearn import metrics
da=metrics.accuracy_score(y_test,y_test_predicted1 )*100
print("Accuracy:",da)
```

```
Accuracy: 90.52271617000488
```

```
: #Evaluation using Confusion matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_test_predicted1)
```

```
: array([[2782,  211],
       [ 177,  924]], dtype=int64)
```

This code trains a decision tree classifier using the training data X_train and y_train, then predicts the target values for the test data X_test using the trained model. The accuracy of the predictions is then calculated using the accuracy_score() function from the sklearn.metrics module. The accuracy_score() function compares the predicted values y_test_predicted1 with the true values y_test and returns the fraction of predictions that are correct.

```
Knn=KNeighborsClassifier()
Knn.fit(X_train,y_train)
y_test_predicted=Knn.predict(X_test)
y_train_predicted=Knn.predict(X_train)
ka=metrics.accuracy_score(y_test,y_test_predicted)*100
ka_t=metrics.accuracy_score(y_train,y_train_predicted)*100

print("Accuracy of test :",ka)

print("Accuracy of train :",ka_t)
```

```
Accuracy of test : 84.53834880312652
Accuracy of train : 88.43898864052765
```

In this code snippet, a KNeighborsClassifier model is being trained on the training data and then used to make predictions on the test data. The KNeighborsClassifier model is an

instance-based method that stores all available instances and predicts the class label of a new instance based on the class labels of the stored instances.

The model is first instantiated using Knn = KNeighborsClassifier(), and then fit to the training data using Knn.fit(X_train, y_train). The fit method trains the model on the training data, which consists of the feature matrix X_train and the label vector y_train.

```
RR=RandomForestClassifier()
RR.fit(X_train,y_train)
y_test_predicted=RR.predict(X_test)
rr_a=metrics.accuracy_score(y_test,y_test_predicted)*100
rr_a_t=metrics.accuracy_score(y_train,y_train_predicted)*100
print("Accuracy of test :",rr_a)

print("Accuracy of train :",rr_a_t)
```

```
Accuracy of test : 93.25842696629213
Accuracy of train : 88.43898864052765
```

```
RP=metrics.precision_score(y_test,y_test_predicted)*100
RP_t=metrics.precision_score(y_train,y_train_predicted)*100
print("Precision of test :",RP)

print("Precision of train :",RP_t)
```

```
Precision of test : 90.72063178677196
Precision of train : 82.67379679144385
```

In this code snippet, a **RandomForestClassifier** model is being trained on the training data and then used to make predictions on the test data. The **RandomForestClassifier** model is an ensemble method that trains multiple decision trees on random subsets of the training data and then combines their predictions to produce a final prediction.

The model is first instantiated using **RR = RandomForestClassifier()**, and then fit to the training data using **RR.fit(X_train, y_train)**. The **fit** method trains the model on the training data, which consists of the feature matrix **X_train** and the label vector **y_train**.

Once the model is trained, it can be used to make predictions on new data using the **predict** method. In this code, the model is used to make predictions on the test data (**X_test**). The predicted class labels are stored in **y_test_predicted**.

The model's performance is then evaluated using the **accuracy_score** and **precision_score** functions from **sklearn.metrics**.

```
#SVM
from sklearn.svm import SVC
```

```
SS=SVC()
SS.fit(X_train,y_train)
y_test_predicted=SS.predict(X_test)
y_train_predicted=SS.predict(X_train)
sa=metrics.accuracy_score(y_test,y_test_predicted)*100
sa_t=metrics.accuracy_score(y_train,y_train_predicted)*100
print("Accuracy of test :",sa)

print("Accuracy of train :",sa_t)
```

```
Accuracy of test : 73.10698583292623
Accuracy of train : 73.51288628313179
```

In this code snippet, a **SVC** (support vector classifier) model is being trained on the training data and then used to make predictions on the test data. The **SVC** model is a linear model for binary classification that finds the hyperplane in the feature space that maximally separates the two classes.

The model is first instantiated using **SS = SVC()**, and then fit to the training data using **SS.fit(X_train, y_train)**. The **fit** method trains the model on the training data, which consists of the feature matrix **X_train** and the label vector **y_train**.

Once the model is trained, it can be used to make predictions on new data using the **predict** method. In this code, the model is used to make predictions on both the test data (**X_test**) and the training data (**X_train**). The predicted class labels are stored in **y_test_predicted** and **y_train_predicted**, respectively.

```
#linear regression
from sklearn.linear_model import LogisticRegression
```

```
LR=LogisticRegression()
LR.fit(X_train,y_train)
y_test_predicted=LR.predict(X_test)
y_train_predicted=LR.predict(X_train)
la=metrics.accuracy_score(y_test,y_test_predicted)*100
la_t=metrics.accuracy_score(y_train,y_train_predicted)*100
print("Accuracy of test :",la)

print("Accuracy of train :",la_t)
```

```
Accuracy of test : 76.7953102100635
Accuracy of train : 77.60473922071577
```

This code is training a logistic regression model on a training dataset (X_train, y_train) and then making predictions on the test set (X_test) and the training set itself. The model's accuracy on the test set is calculated using the accuracy_score function from scikit-learn's metrics module, which compares the model's predicted labels (y_test_predicted) to the true labels (y_test) and returns the proportion of predictions that are correct. The same procedure is performed for the training set to calculate the model's accuracy on the training set (la_t). The accuracy scores are then printed to the console.

Confusion Matrix of KNN

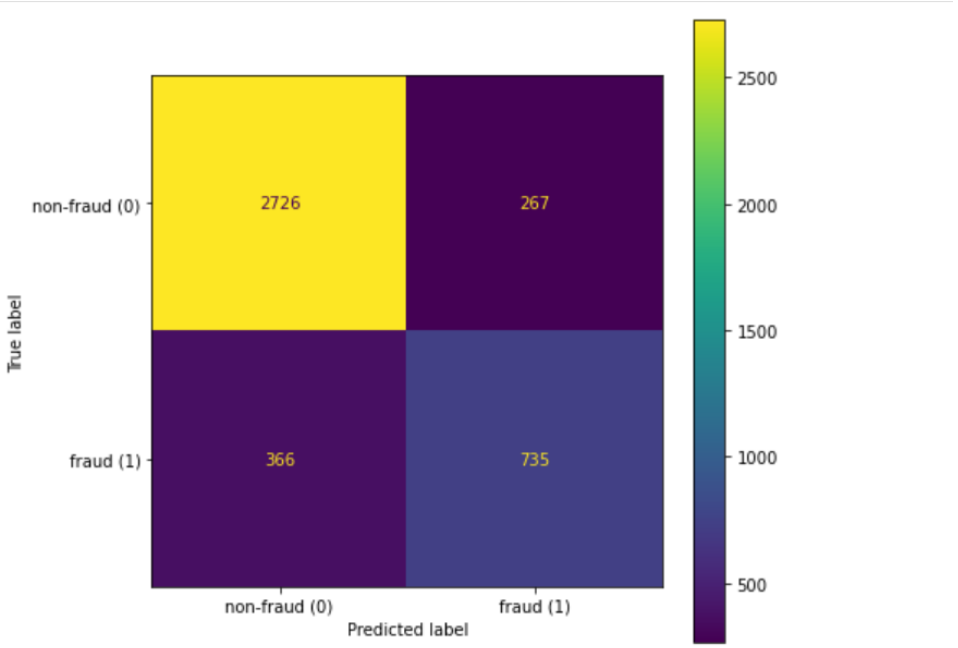

Fig-5.1

Confusion Matrix of SVM

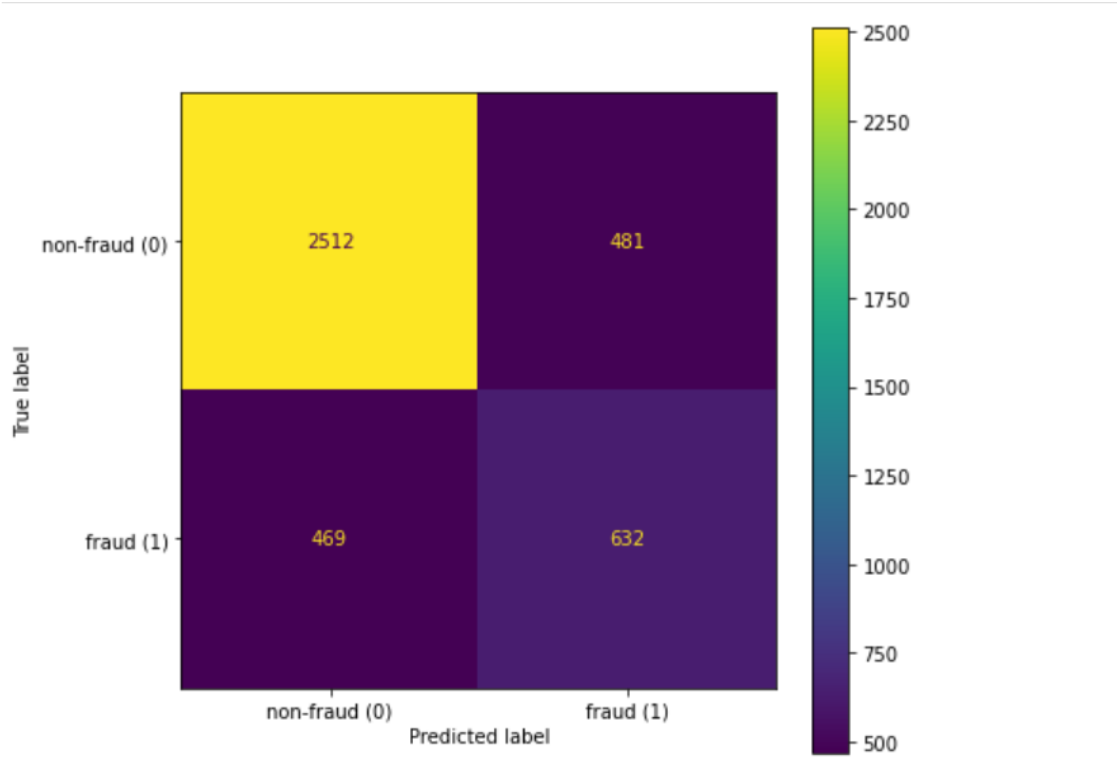

Fig-5.2

Confusion Matrix of Decision Tree

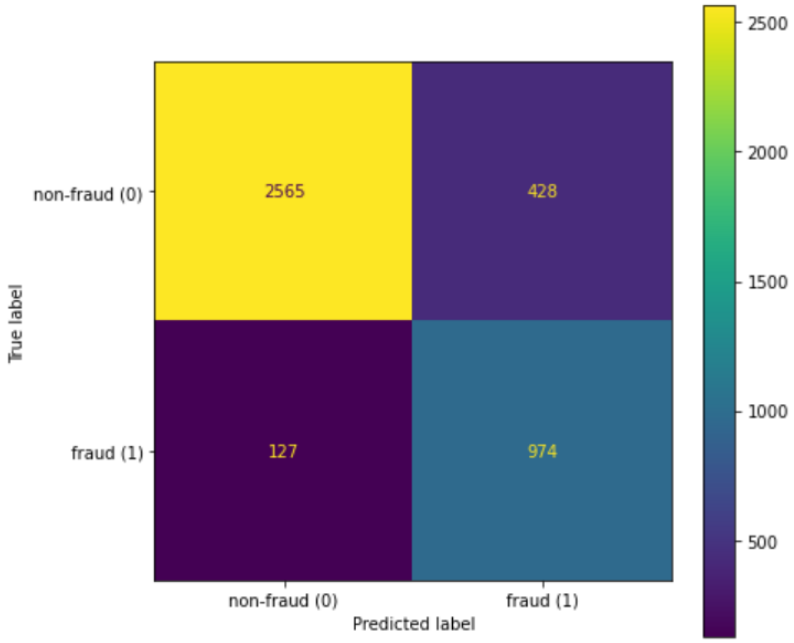

Fig-5.3

Confusion Matrix of Random Forest

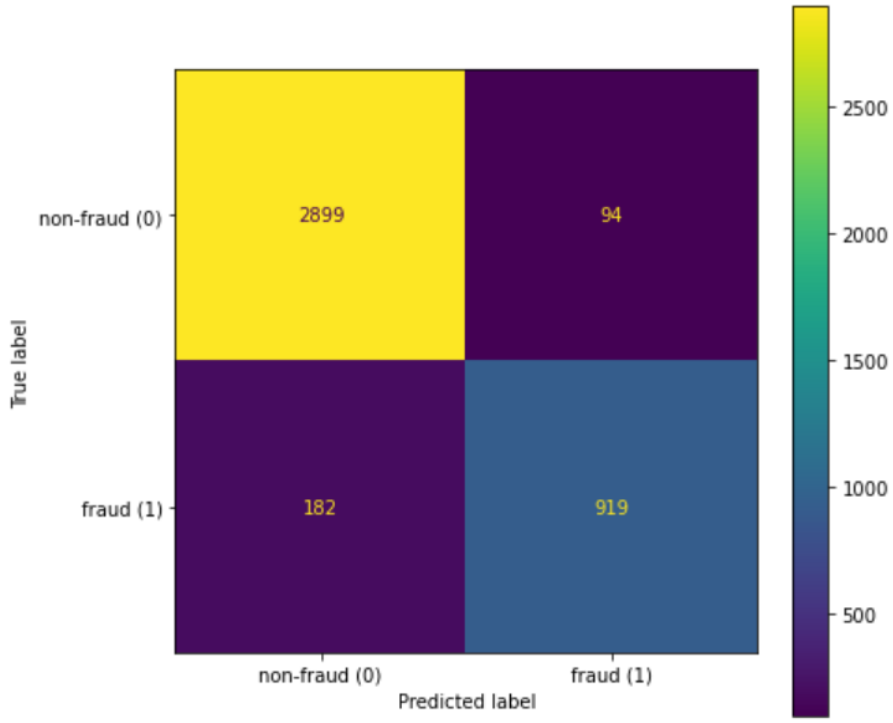

Fig-5.4

Confusion Matrix of Logistics Regression

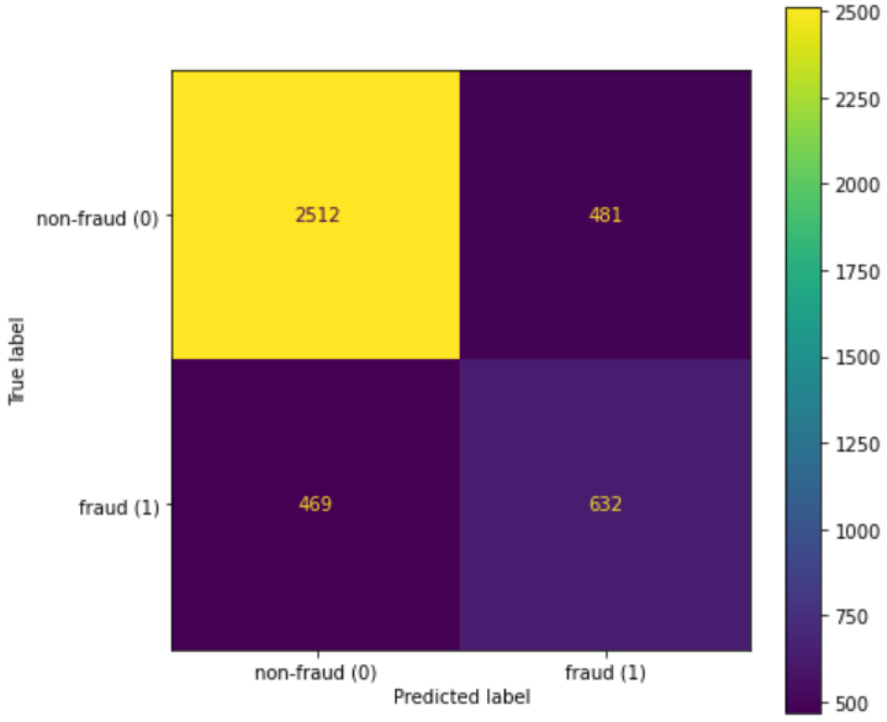

Fig-5.5

Accuracy of test for all five models

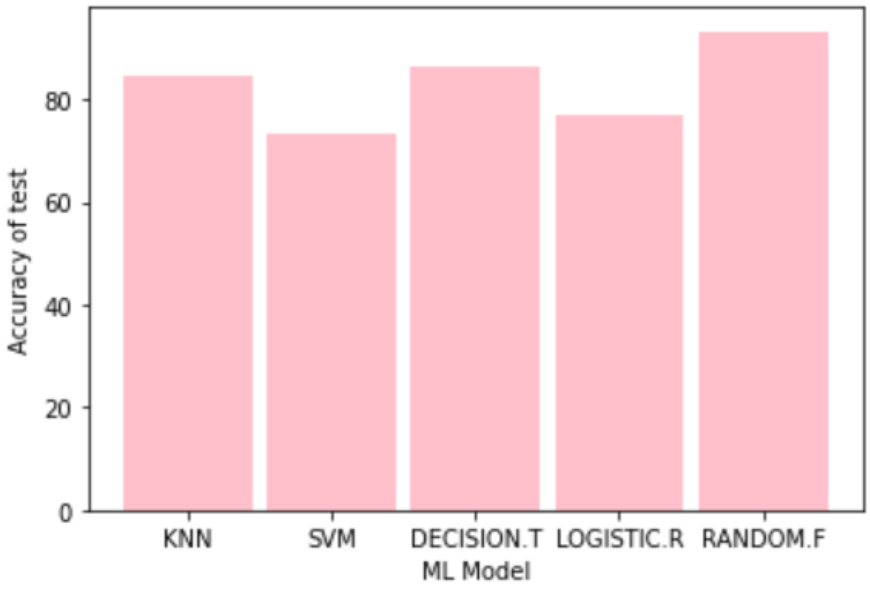

Fig-5.6

Precision of test for all five models

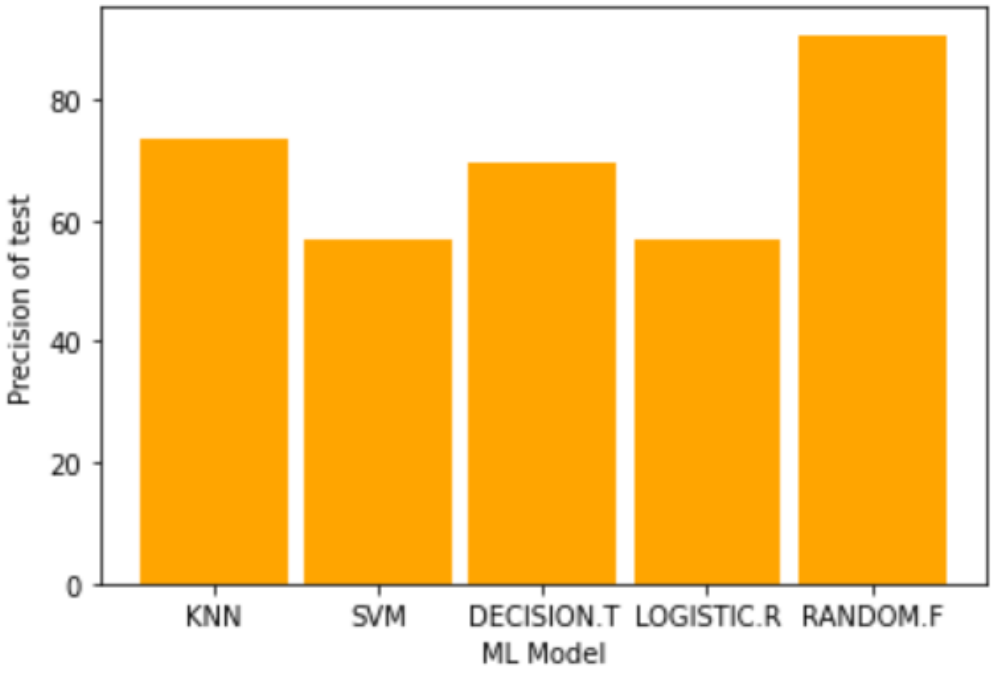

Fig-5.7

recall_score of test for all five models

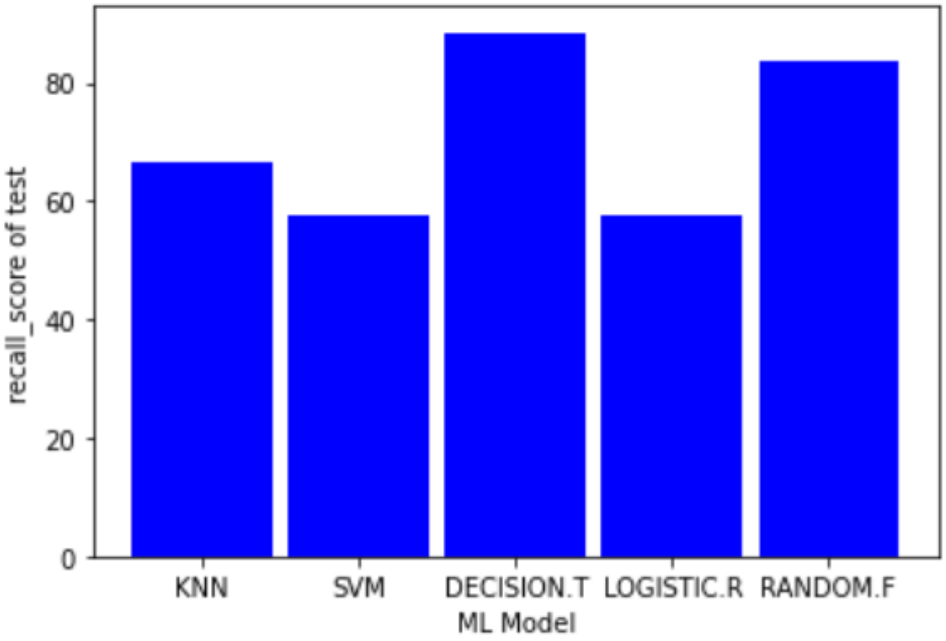

Fig-5.8

f1_score of test for all five models

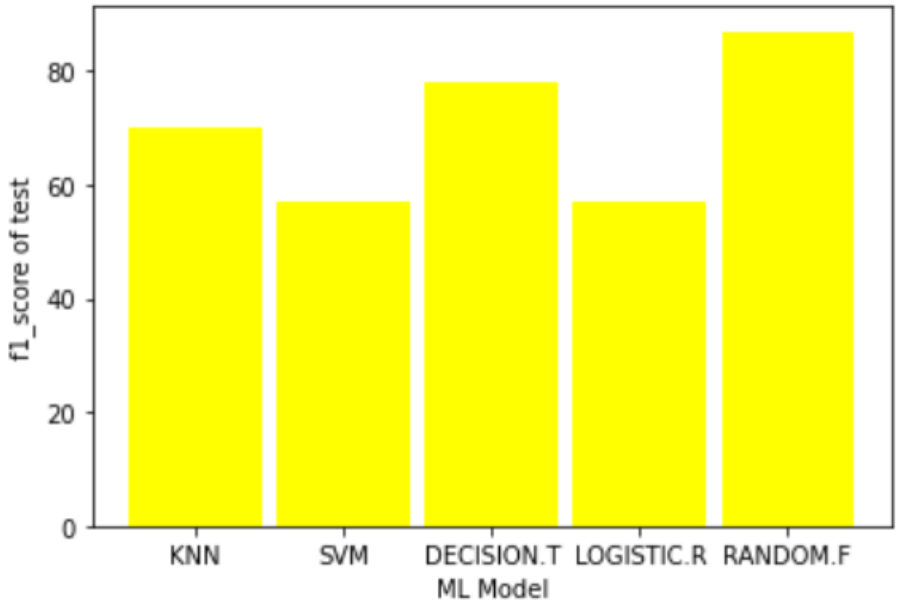

Fig-5.9

Conclusion of Accuracy, Precision score, Recall Score, F1-Score for all five algorithms:

	K-NN	SVM	DECISION TREE	LOGISTIC REGRESSION	RANDOM FOREST
ACCURACY	84.53834	73.1069	86.4435	76.7953	93.2584
PRECISION	73.3535	56.7834	69.4721	56.7834	90.7206
RE-CALL SCORE	66.7574	57.4023	88.4650	56.4023	83.469
F1_SCORE	69.9001	57.09123	77.8266	57.09123	86.9441
CONFUSION MATRIX	[[272600 26700] [36600 73500]]	[[251200 48100] [46900 63200]]	[[256500 42800] [12700 97400]]	[[251200 48100] [46900 63200]]	[[289900 9400] [18200 91900]]

Table-5.1

1. Accuracy for test is high in: Random Forest > Decision Tree > K-NN > Logistic regression > SVM
2. Precision score for test is high in: Random Forest > K-NN > Decision tree > SVM > Logistic regression
3. Re-call score for test is high in : Decision tree > Random Forest > K-NN > SVM > Logistic regression
4. f1_score for test is high in : Random Forest > Decision tree > K-NN > Logistic regression > SVM

CHAPTER-6

CONCLUSION

In conclusion, the analysis of fraud detection using KNN, SVM, decision tree, and random forests showed that all four methods can be effective in detecting fraud.

KNN performed well in identifying cases where the number of fraudulent instances was small compared to the overall number of instances. SVM was able to achieve high accuracy and a low false positive rate, making it a strong choice for fraud detection.

Decision tree was able to provide clear and interpretable rules for identifying fraudulent cases.

Random forests achieved strong performance and robustness due to its ability to reduce overfitting through the use of multiple decision trees.

We studied in this context, two cases of fraud in banks: credit card fraud and money laundering. The performance of the proposed system was tested on the benchmarks General Ledger, Payables Data, created as similar to bank database. The precision obtained for the single class SVM method, was of about 80%, which represents a significant improvement in comparison to similar works reference. For the method, the slight improvement on credit scoring databases was because of the difficulty of obtaining real databases. The results can be improved by studying the influence of various parameters used by the SVM-S architecture.

Overall, the choice of which method to use will depend on the specific characteristics and needs of the dataset and the fraud detection problem at hand.

CHAPTER-7

REFERENCES

[1] Faraji, Z. (2022). A Review of Machine Learning Applications for Credit Card Fraud Detection with A Case study. SEISENSE Journal of Management, 5(1), 49–59.

https://doi.org/10.33215/sjom.v5i1.770

[2] "An Empirical Evaluation of Machine Learning Algorithms for Fraud Detection in the Banking Industry" by Li et al. (2020) presents a study on the use of various machine learning algorithms, including KNN and SVM, for fraud detection in the banking industry. The authors found that an ensemble of multiple algorithms, including KNN and SVM, outperformed individual algorithms in terms of accuracy.

https://www.academia.edu/76648808/Machine_Learning_Algorithms_in_Fraud_Detection_Case_Study_on_Retail_Consumer_Financing_Company

[3] Pinku Ranjan; Kammari Santhosh; Arun Kumar; Somesh Kumar, “Fraud Detection on Bank Payments Using Machine Learning”, 2022 International Conference for Advancement in Technology (ICONAT), IEEE Conference, 2022.
https://www.researchgate.net/publication/359167132_Fraud_Detection_on_Bank_Payments_Using_Machine_Learning

[4] 3rd International Conference on Science and Sustainable Development (ICSSD 2019) IOP Conf. Series: Journal of Physics: Conf. Series1299 (2019) 012037IOP Publishing doi:10.1088/1742-6596/1299/1/012037
Fraud prediction in bank loan administration using decision tree | Ambrose Azeta - Academia.edu