**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**

Bank fraud is a growing problem that impacts individuals and financial institutions worldwide. Traditional methods of detecting fraud, such as manual review of suspicious activity, are time-consuming and prone to human error. Banking Fraud has been an issue with huge consequences to banks and customers alike, both in terms of financial losses, trust and credibility. As per the Nilson report, it is anticipated that card frauds alone would amount to a whopping $30 billion worldwide by 2020. Also, with the technology disruption in both banking and, the number of transactions has increased exponentially in recent years. Fraudsters have also become extremely smart, adopting innovatory fraudulent tactics. As a result, it has compounded the problem.

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. These algorithms will be implemented in Python and evaluated based on their ability to accurately classify fraudulent and non-fraudulent transactions. The results of this study will be used to provide recommendations for improving the detection of bank fraud using machine learning techniques.

Firstly, topre-process the dataset, we will start by handling missing values. This may involve imputing missing values with a suitable substitute such as the mean or median of the available values, or dropping rows or columns with too many missing values. Next, we will perform feature engineering to extract useful features from the raw data that may help improve the performance of our machine learning algorithms. This may involve creating new features by combining existing ones, or selecting a subset of the most relevant features. Once the dataset is cleaned and prepared, we will train and test each of the four algorithms on it, using appropriate evaluation metrics such as precision, recall, and F1 score to compare their performance. The goal is to identify the algorithm that performs the best on the dataset, and fine-tune its hyperparameters to further improve its performance.

Finally, after training and testing the four algorithms on the pre-processed dataset, we will compare their results and identify the one that performs the best on the task of bank fraud detection. This will be our recommended model for use in detecting and preventing bank fraud. Our hope is that this project will contribute to the development of more effective methods for detecting and preventing bank fraud, ultimately leading to a safer and more secure financial system. The goal of this project is to accurately identify fraudulent transactions in order to protect both financial institutions and consumers from financial loss. By developing a better understanding of the patterns and characteristics of fraudulent activity, we can work towards building a system that is better able to detect and prevent fraudulent transactions from occurring. This will benefit everyone involved by helping to ensure that financial transactions are secure and that financial institutions and consumers are protected from financial loss.

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

K-nearest neighbors (KNN): KNN is a type of supervised learning algorithm that can be used for classification tasks. It works by finding the K nearest neighbors of a given data point, and then classifying the point based on the majority class of its neighbors. For example, in the context of fraud detection, a KNN classifier could be trained on a dataset of transactions labeled as either "fraudulent" or "legitimate", and then used to classify new transactions as one of these two classes.

Support vector machines (SVMs): SVM is another type of supervised learning algorithm that can be used for classification tasks. It works by finding the hyperplane in a highdimensional space that maximally separates different classes. For example, in the context of fraud detection, an SVM classifier could be trained on a dataset of transactions labeled as either "fraudulent" or "legitimate", and then used to classify new transactions as one of these two classes.

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. 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.

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.

**LITERATURE SURVEY**

1. **Tiltle:** A Review of Machine Learning Applications for Credit Card Fraud Detection with A Case study
   1. **Authors:** Dr. Zahra Faraji
   2. **Year Of Publish:** 2022
   3. **Abstract:** This paper aims to highlight the widely used supervised techniques applied for fraud detection. In addition, this paper 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.
2. **Tiltle:** Credit Card Fraud Detection Based on Deep Neural Network Approach
   1. **Authors:** Khalid I. Alkhatib,Ahmad I. Al-Aiad, Mothanna H. Almahmoud, Omar N. Elayan
   2. **Conference:** 12th International Conference on Information and Communication Systems (ICICS)
   3. **Year Of Publish:** 2021
   4. **Abstract:** From this paper we learnt that fraud is a serious problem for financial sector as it provides critical services, and there is a big need for more developments to prevent attackers from harming customers. This paper basically aims on research where a dataset is used for transactions of well-known company Vesta, in order to build a new model for credit card fraud detection.
3. **Tiltle:** Comparative Study of Machine Learning Algorithms for Fraud Detection in Blockchain
   1. **Author:** Madhuparna Bhowmik
   2. **Conference:** 5th International Conference on Computing Methodologies and Communication (ICCMC)
   3. **Year Of Publish:** 2021
   4. **Abstract:** This paper compares the performance of KNN, SVM, and several other machine learning algorithms on a dataset of bank transactions. The authors found that SVM outperformed KNN in terms of accuracy, but KNN was faster to train.

**SYSTEM MODEL**

1. **K-nearest neighbors (KNN):**

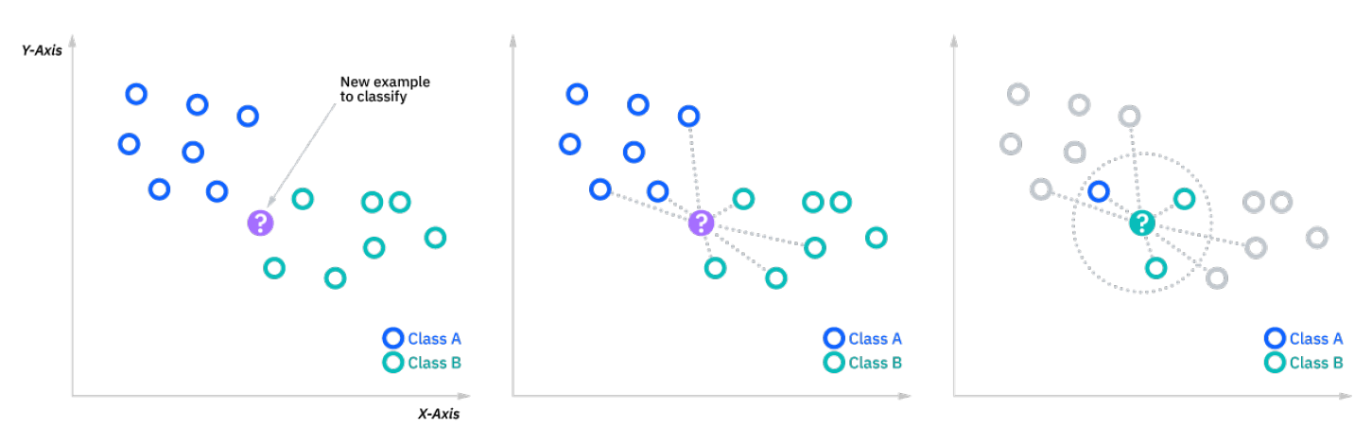
K-Nearest Neighbors (KNN) is a classification and regression algorithm that is based on the idea of finding the K closest training examples in the feature space and using these examples to make predictions.

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.



**2). Support vector machines (SVMs):**

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.



**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.

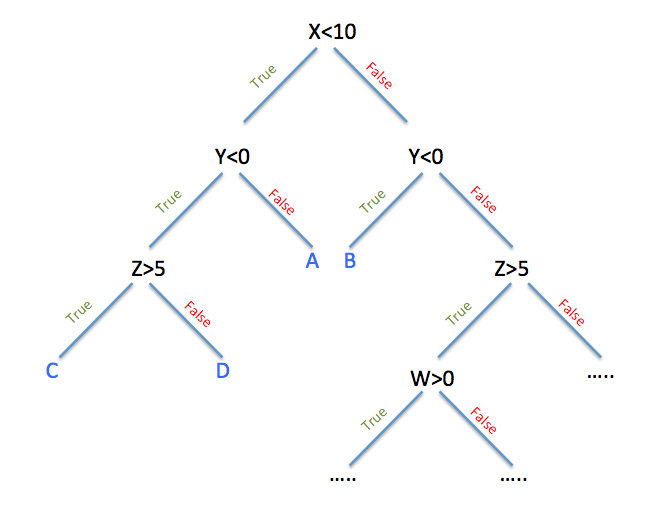
Here's a general outline of the Decision Tree algorithm:

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.



**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:

1. Select a number of decision trees to build.
2. 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.
3. 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.



1. **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:

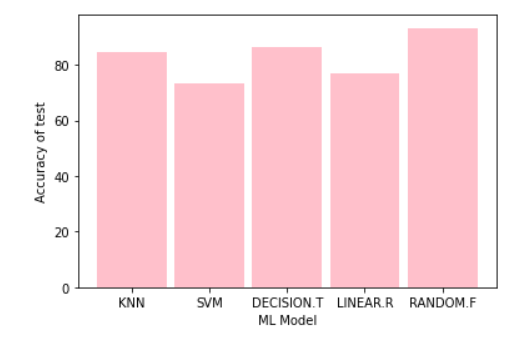
1. Collect and prepare the training data.
2. Initialize the model parameters.
3. Make predictions using the current model parameters.
4. Calculate the error between the predicted probabilities and the true labels.
5. Update the model parameters using an optimization algorithm (such as gradient descent) 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.

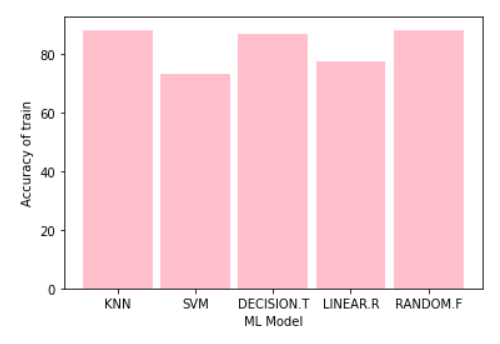
However, Logistic Regression may not perform well if the data is not linearly separable or if there are a large number of categorical features. It is also sensitive to the scale of the input features.



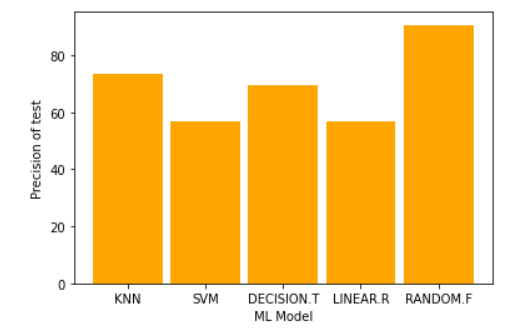
**Accuracy of test for all five models**

****

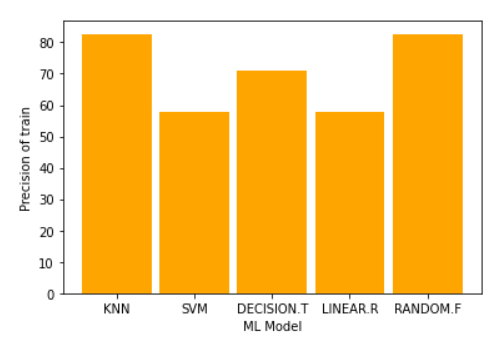
**Accuracy of train** **for all five models**



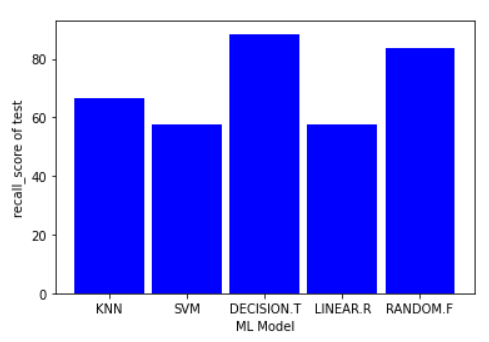
**Precision of test for all five models**

****

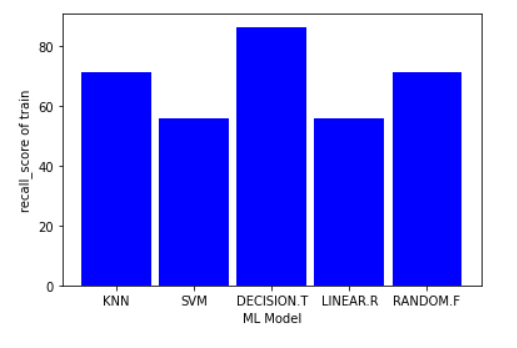
**Precision of train for all five models**

****

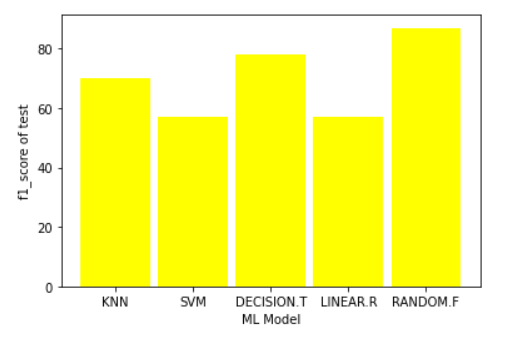
**recall\_score of test for all five models**

****

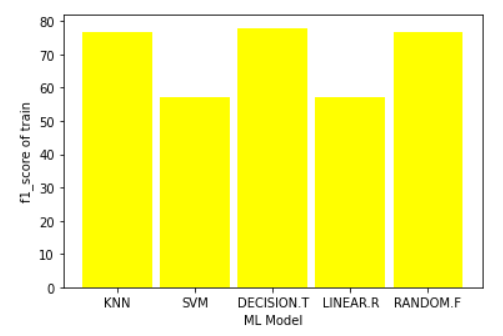
**recall\_score of train for all five models**

****

**f1\_score of test for all five models**

****

**f1\_score of train for all five models**

****

**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.

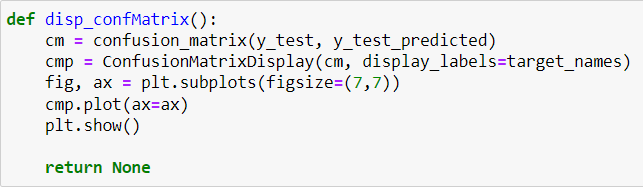
**9) sklearn.linear\_model**

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.

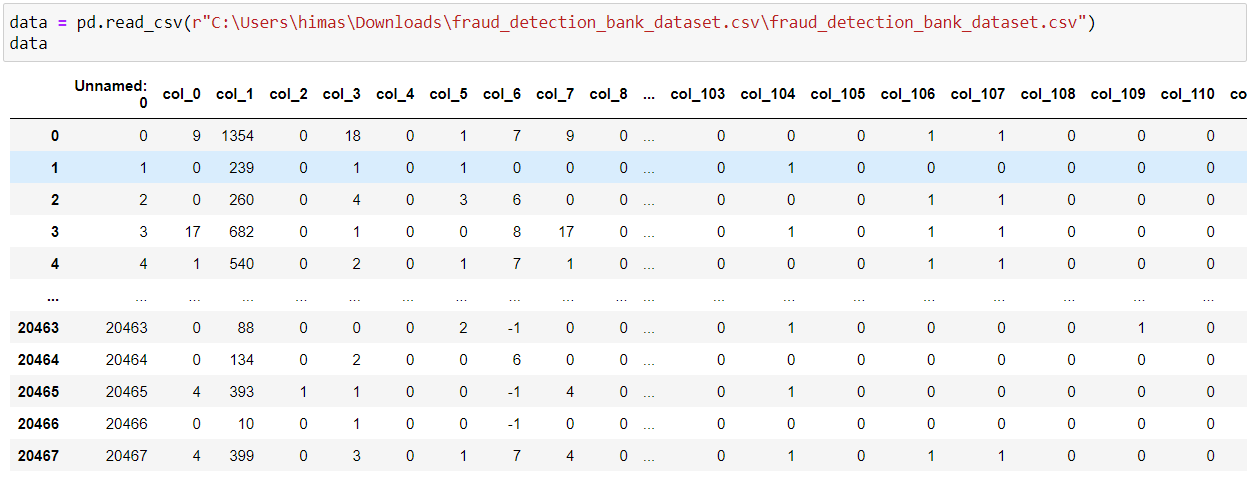
**Functions Used:**

****

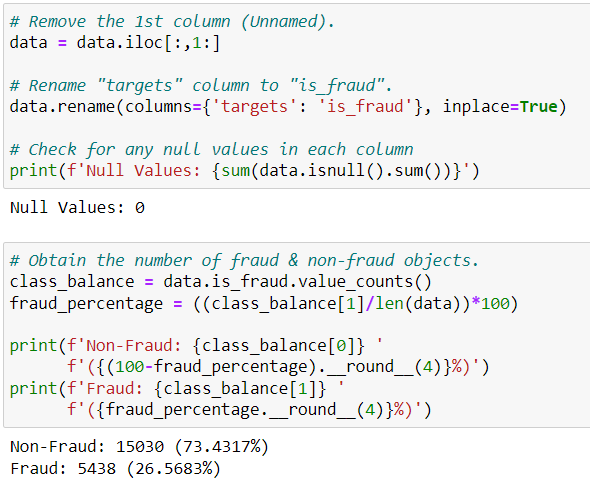
The disp\_confMatrix function you have defined appears to be a function for displaying a confusion matrix for a set of test predictions. A confusion matrix is a matrix that is used to visualize the performance of a classification model. It is often used to evaluate the accuracy of the model and to identify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

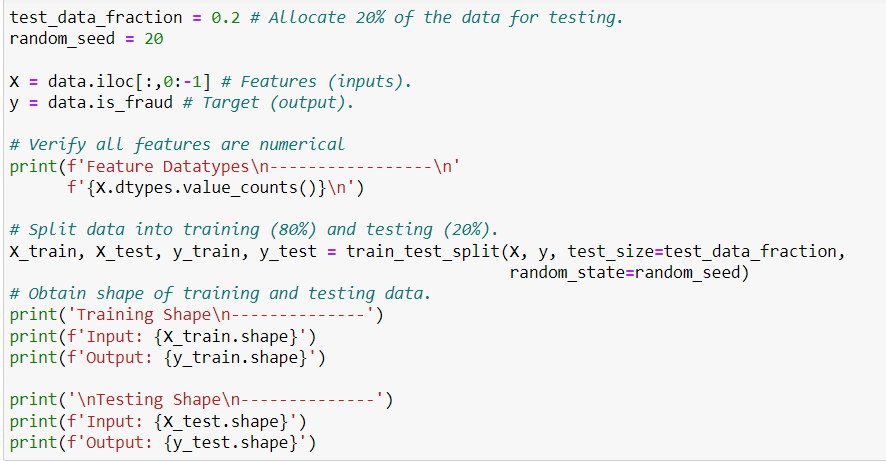
The confusion matrix is constructed by comparing the predicted class labels with the true class labels for a set of data. The rows of the matrix correspond to the true classes, and the columns correspond to the predicted classes. Each element of the matrix shows the number of samples that were predicted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

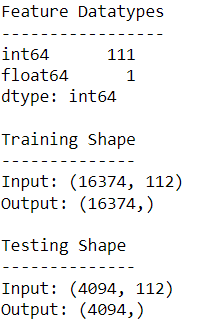
**SAMPLE CODE & SAMPLE OUTPUT**

****

Downloading and Reading the Dataset

****

****

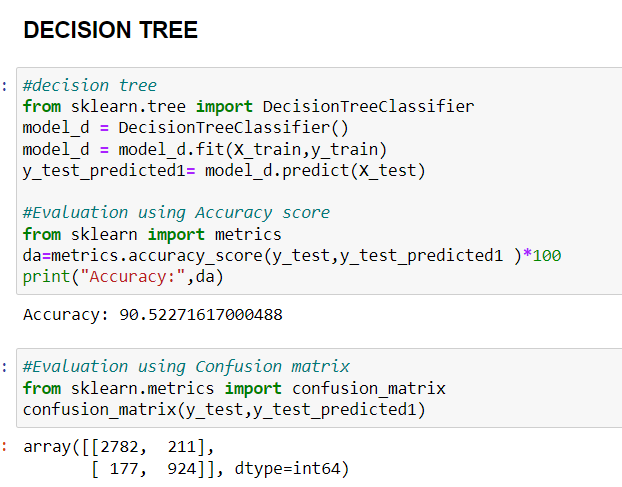
****

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.

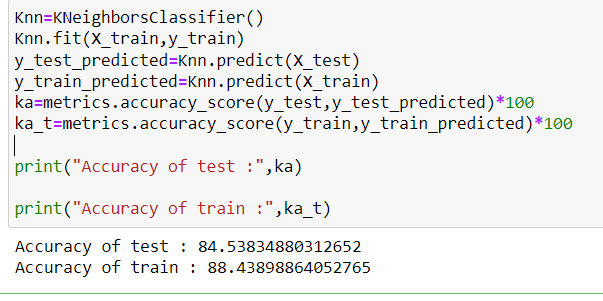
The input data X consists of all columns in data except the last column, which is assumed to be the target or output. The target data y consists of the last column of data, which is assumed to be a binary classification of whether a transaction is fraudulent or not.

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.

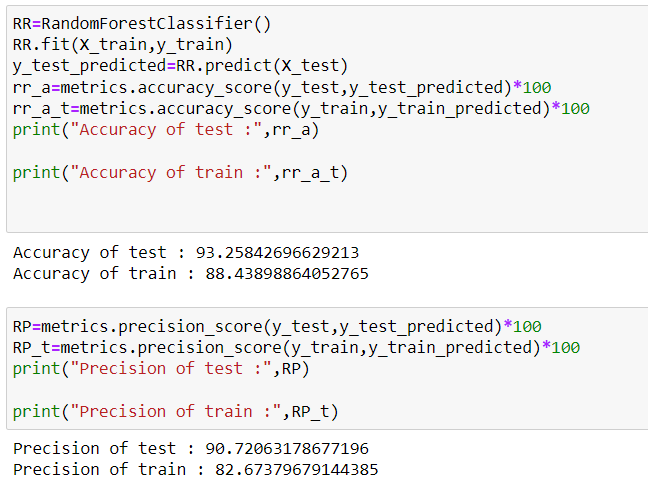


This code trains a decision tree classifier using the training data X\_train and y\_train, then predicts the target values for the test data X\_test using the trained model. The accuracy of the predictions is then calculated using the accuracy\_score() function from the sklearn.metrics module. The accuracy\_score() function compares the predicted values y\_test\_predicted1 with the true values y\_test and returns the fraction of predictions that are correct.

****

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.

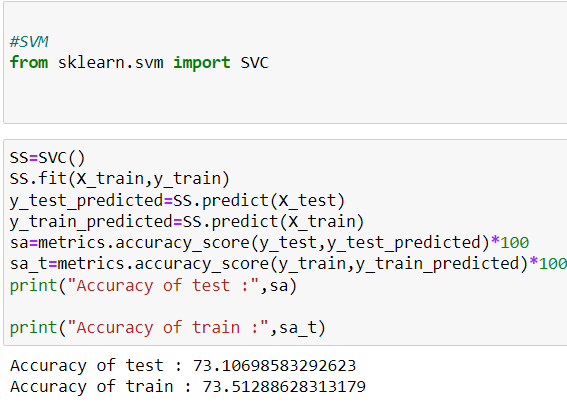


In this code snippet, a **RandomForestClassifier** model is being trained on the training data and then used to make predictions on the test data. The **RandomForestClassifier** model is an ensemble method that trains multiple decision trees on random subsets of the training data and then combines their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

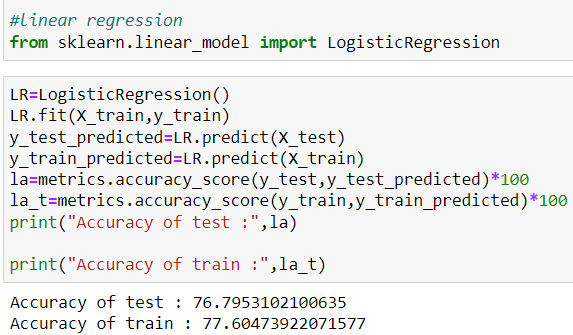
The model's performance is then evaluated using the **accuracy\_score** and **precision\_score** functions from **sklearn.metrics**.



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.

****

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.

**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.

**REFERENCES**

[1] Alkhatib, K. I.-A. (2021). Credit Card Fraud Detection Based on Deep Neural Network Approach. 12th International Conference on Information and Communication Systems (ICICS) (pp. 153-156). IEEE. DOI: <https://doi.org/10.1109/ICICS52457.2021.9464555>

[2] 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.

[3] <https://www.kaggle.com/>

[4] Alkhatib, K. I.-A. (2021). Credit Card Fraud Detection Based on Deep

Neural Network Approach. 12th International Conference on Information

and Communication Systems (ICICS) (pp. 153-156). IEEE. DOI

[5] "A Comparative Study of Machine Learning Algorithms for Fraud Detection in the Banking Industry" by Hossain et al. (2020) compares the performance of KNN, SVM, and several other machine learning algorithms on a dataset of bank transactions. The authors found that SVM outperformed KNN in terms of accuracy, but KNN was faster to train.