B. P. Poddar Institute of Management & Technology

Department of Computer Science Engineering

Course Name: M. Tech - Data Science

Academic Year: 2023-24, Semester: Odd

MINI PROJECT

Paper Name: Machine Learning Laboratory Code: PGIT(DS)194B

Mini Project Title: **UPI Fraudulent Transaction Operation Recognition using Machine Learning Model**

Students Details

Name of Student	**University Roll Number**
Subarna Paul	11561723008

<Please do not write anything below the dotted line>

……

Marks awarded

Marks awarded	
Total Marks	

Signature of Faculty with date _____

Contents

Topic:

UPI Fraudulent Transaction Operation Recognition using Machine Learning Model.

Abstract:

This report focuses on the task of applying different machine learning (ML) models to recognize ever so popular Unified Payments Interface (UPI) fraudulent. On-going studies emphasizes the utility of machine learning algorithms for the analysis and recognition of online banking transactions. The report outlines diverse approaches to enhance detection precision, such as addressing imbalances in datasets, transforming features, and engaging in feature engineering. The outcomes of distinct algorithms are presented visually, with logistic regression and several others demonstrating comparable performance, achieving an accuracy score of 80%.

Introduction:

Unified Payments Interface (UPI) is a real-time payment system widely used in India. It enables seamless money transfers between bank accounts through mobile devices. Users can link multiple bank accounts to a single mobile application, simplifying transactions. This simplification of peer-to-peer transaction has made the very method of money transfer more vulnerable and open to threat for attackers. In recent years, there has been notable attention given to the advancement of artificial intelligence in identifying fraudulent UPI transactions. This heightened focus is attributed to the increasing occurrences of fraudulent activities within the banking industry, leading to substantial financial losses for both banks and their customers. Our project centred on the utilization of machine learning models to detect fraudulent UPI transactions. Employing pre-processing techniques, we compared and selected the most effective outcomes from the bank data. To achieve our objectives, we executed the following steps:

1. Developed multiple machine learning models employing diverse methodologies and strategies.

2. Assessed and compared the models from the preceding stage using both quantitative and visual criteria.

3. Analysed the obtained results and formulated conclusions regarding the research objective.

Integrating ML technology for identifying fraudulent banking operations presents several challenges, notably the lack of transparency and interpretability in the employed algorithms. Understanding the intricacies of these algorithms can be challenging, hindering the prompt identification and rectification of errors. Also, lack of recent reliable data sources and inconsistency in the existing data may contribute towards the accuracy score of the model.

Working Dataset:

The dataset consists of 2000 records with various features related to UPI transactions. Each record includes details such as transaction hour, day, month, and year, transaction category, UPI number, age of the user, transaction amount, state, and a binary indicator for fraud risk (1 for fraud, 0 for valid). The dataset has been balanced with 1180 fraud cases and 820 valid cases. The 'id' column serves as the unique identifier for each record. The dataset has undergone pre-processing, including the introduction of missing values and subsequent imputation.

	id	trans_hour	trans_day	trans_month	trans_year	category	upi_number	age	trans_amount	state	fraud_risk
0	0	6	26	1	2022	10	9957000271	32	3096.74	5	1.0
1	1	19	8	1	2022	9	9957000387	63	1556.14	28	0.0
2	2	14	4	4	2022	20	9957000093	32	438.11	19	1.0
3	3	10	16	3	2022	8	9957000966	21	463.77	21	1.0
4	4	7	30	10	2022	10	9957000529	18	4616.89	24	1.0
5	5	20	23	12	2022	19	9957000154	40	3449.50	28	1.0
6	6	6	10	2	2022	3	9957000405	19	644.24	18	1.0
7	7	18	12	12	2022	24	9957000840	74	1790.05	21	1.0
8	8	22	14	3	2022	5	9957000149	69	2966.49	22	1.0
9	9	10	14	11	2022	3	9957000889	30	4179.50	10	1.0
10	10	10	2	11	2022	16	9957000267	49	227.87	24	0.0

Figure 1: Tabular structure of working dataset

Material and Methods:

In pursuit of the objectives defined in this study, we utilized classification algorithms. These algorithms leverage features to ascertain the class of a given object. Machine learning depends on labelled training data for categorizing new observations. To ensure precise predictions for forthcoming observations, these algorithms analyze a dataset comprising examples with features and corresponding classes. They fall under the category of supervised learning techniques, as they map input variables (x) to discrete output functions (y) representing categories, as opposed to numerical values. Classification algorithms produce discrete outputs rather than continuous ones. These algorithms learn from labelled input data, where each input data point is associated with a specific output or class.

The objective of classification algorithms is to categorize a given dataset, and they are extensively employed to predict the output for categorical data. The research workflow, as depicted in the figure, illustrates the stages of a high-level algorithm for a machine learning program solution. This encompasses dataset selection and loading, feature standardization, random under sampling, model fitting, model testing, and outputting the best model.

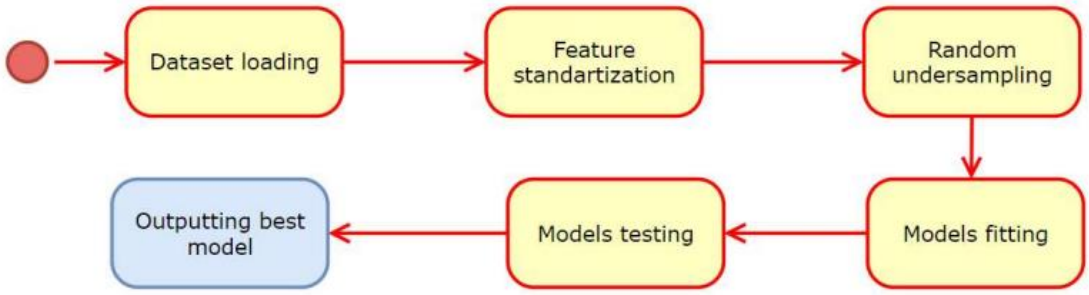

Figure 2: Program flow of the project

Framework to build the set of machine learning algorithms:

• Selecting the dataset and loading the dataset into the program (using a library such as pandas or NumPy to load the dataset into the program).

• Splitting the dataset into training and testing sets (using a library such as scikit-learn) and standardizing the features in the training and testing sets using a standard scalar.

• Imbalanced learning to randomly under sample the majority class in the training set to balance the class distribution.

• For each model, fitting the model to the training data using hyperparameter tuning (using a library such as scikit-learn).

• Using a library such as scikit-learn to evaluate each model on the testing data using an appropriate evaluation metric and selecting the model with the best evaluation metric on the testing data.

• Outputting the best model and its hyperparameters and evaluation metric for further use in production or research.

Machine Learning Algorithms:

The machine learning algorithms chosen to be used here were:

1) Logistic Regression (LR)
2) K-Nearest Neighbours (KNN)
3) Support Vector Machine (SVM)
4) Naive Bayes (NB)
5) Decision Tree (DT)
6) Random Forest (RF)

Logistic regression (LR), stands as a frequently employed statistical model in classification and predictive analytics. It estimates the probability of an event based on a specified set of independent variables. Utilizing a logistic transformation, logistic regression converts the odds, representing the logarithm of the odds or the natural logarithm of the odds.

K-nearest neighbours (KNN), serves as a nonparametric classifier in supervised learning. This classifier relies on proximity to categorize or predict the grouping of an individual data point. Commonly employed for classification, KNN assigns a class label based on the majority vote among nearby data points.

Support vector machine (SVM), is a supervised learning algorithm utilized for both classification tasks and regression assignments. SVM establishes a decision boundary, or hyper plane, to segregate n-dimensional space into distinct classes.

Naive Bayes classifier (NB), a probabilistic machine learning model designed for classification tasks, relies on Bayes' theorem. This classifier operates under the assumption that all predictors or features are independent, implying that the presence of one feature does not influence the others.

Decision tree (DT), is a hierarchical structure resembling a flowchart. Each internal node in the tree signifies an attribute check, and each branch corresponds to the outcome of that check. The terminal nodes, or end nodes, contain class labels. The decision tree is trained by dividing the initial dataset into subsets based on the values of attributes during the attribute-checking process.

Random Forest (RF) is a technique widely applied by specialists for classification and regression tasks. The potency of the forest grows with an increased number of trees. This algorithm constructs decision trees using randomly selected data samples, generates predictions from each tree, and determines the best solution through a voting mechanism.

Comparative Study of Algorithms:

The results of the analysis of the strengths and weaknesses of each algorithm are presented in the following table. The efficacy of these algorithms varies based on the particular problem and dataset under consideration.

Algorithm name	Pros	Cons
Logistic Regression (LR)	Performs well in handling high-dimensional data with complex relationships, missing values, and outliers	Computationally expensive for massive datasets and challenging to interpret.
K-Nearest Neighbors	Easy to implement and works well on small datasets, handling nonlinear relationships.	Computationally expensive for large datasets sensitive to irrelevant features and distance metrics.
Support Vector Machine (SVM)	Performs well on many problems, including nonlinear data, and handles high-dimensional data.	Computationally expensive for large datasets. Sensitive to choose kernel function and hyper-parameters.
Naive Bayes (NB)	Fast and straightforward, performing well in handling categorical features.	Assumes independence between features. may not perform well on highly nonlinear data.
Decision Tree (DT)	Easy to interpret and handles nonlinear relationships. Good performance on small to medium datasets.	Easily over fits the data and is sensitive to small changes in data.
Random Forest (RF)	Performs well in handling high-dimensional data with complex relationships, missing values, and outliers.	Computationally expensive for massive datasets and challenging to interpret.

Table 1: Comparative study of Pros and Cons of the listed algorithms

Algorithm Design:

The technical implementation of the task described in this paper was carried out using the high-level programming language Python. Python is acknowledged for its simplicity, consistency, flexibility, robust AI and ML libraries, platform independence, and the extensive Python community, making it an ideal choice for machine learning and AI-driven projects. The development environment employed for this project was Jupyter Notebook. Notebooks consist of cells that can be formatted in Markdown for text or a programming language. In the technical implementation of this task, the following libraries were employed:

1) Scikit-learn is an open-source machine learning library designed for the Python programming language. It encompasses a range of algorithms for classification, regression, and clustering, including support-vector machines.
2) Pandas is a Python open-source software library crafted for data manipulation and analysis. It provides data structures and operations tailored for manipulating numerical tables and time series data.
3) Matplotlib is a plotting library designed for the Python programming language and its numerical mathematics extension NumPy. It furnishes an object-oriented API for incorporating plots into applications.

The dataset was partitioned into training and testing sets, comprising 80% and 20%, respectively, to assess generalization performance. Logistic Regression (LR), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Naive Bayes (NB), Decision Tree (DT), and Random Forest (RF) algorithms were employed to address the classification task.

Program Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

dataset = pd.read_csv('fraud_detection_dataset.csv')
dataset.head()
dataset.shape
x = dataset.iloc[:, :10].values
y = dataset.iloc[:, 10].values
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 2)
x_train.shape
x_test.shape
fraud = np.count_nonzero(y_train == 1)
valid = np.count_nonzero(y_train == 0)
print("Fraud cases in training data: ", fraud)
print("Valid cases in training data: ", valid)

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

LR:
from sklearn.linear_model import LogisticRegression
LR_model = LogisticRegression(random_state = 2)
LR_model.fit(x_train, y_train)
y_pred = LR_model.predict(x_test)
acc_lr = accuracy_score(y_test,y_pred)
print(acc_lr)

KNN:
from sklearn.neighbors import KNeighborsClassifier
KNN_model = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
KNN_model.fit(x_train, y_train)
y_pred = KNN_model.predict(x_test)
acc_knn = accuracy_score(y_test,y_pred)
print(acc_knn)

SVM:
from sklearn.svm import SVC
SVM_model = SVC(kernel = 'linear', random_state = 0)
SVM_model.fit(x_train, y_train)
y_pred = SVM_model.predict(x_test)
acc_svm = accuracy_score(y_test,y_pred)
print(acc_svm)

SVM_model = SVC(gamma = 0.001)
SVM_model.fit(x_train, y_train)
```

```
y_pred = SVM_model.predict(x_test)
acc_svm = accuracy_score(y_test,y_pred)
print(acc_svm)

SVM_model = SVC(C=5)
SVM_model.fit(x_train, y_train)
y_pred = SVM_model.predict(x_test)
acc_svm = accuracy_score(y_test,y_pred)
print(acc_svm)

SVM_model = SVC(kernel='rbf')
SVM_model.fit(x_train, y_train)
y_pred = SVM_model.predict(x_test)
acc_svm = accuracy_score(y_test,y_pred)
print(acc_svm)

NB:
from sklearn.naive_bayes import GaussianNB
NB_model = GaussianNB()
NB_model.fit(x_train, y_train)
y_pred = NB_model.predict(x_test)
acc_nb = accuracy_score(y_test,y_pred)
print(acc_nb)

DT:
from sklearn.tree import DecisionTreeClassifier
DT_model = DecisionTreeClassifier (criterion = 'entropy')
DT_model.fit(x_train, y_train)
y_pred = DT_model.predict(x_test)
acc_dt = accuracy_score(y_test,y_pred)
print(acc_dt)

RF:
from sklearn.ensemble import RandomForestClassifier
RF_model = RandomForestClassifier()
RF_model.fit(x_train, y_train)
y_pred = RF_model.predict(x_test)
acc_rf = accuracy_score(y_test,y_pred)
print(acc_rf)

ACCURACY:

scores = [
    acc_lr * 100,
    acc_knn * 100,
    acc_svm * 100,
    acc_nb * 100,
    acc_dt * 100,
    acc_rf * 100,
]

names = ["Logistic Regression",
```

```
"K-Nearest Neighbors",
"Support Vector Machine",
"Naive Bayes",
"Decision Tree",
"Random Forest"]

df = pd.DataFrame()
df['Algorithm Name'] = names
df ['Accuracy Score (%)'] = scores
df = df. sort_values('Accuracy Score (%)', ascending = False)

df

plt.figure(figsize=(10, 6))
plt.bar(df['Algorithm Name'], df['Accuracy Score (%)'], color='blue')
plt.xlabel('Algorithm Name')
plt.ylabel('Accuracy Score (%)')
plt.title('Accuracy Scores of Different Algorithms')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```

Result Analysis:

The results of each of the classification algorithms are plotted using matplotlib plotting graph and shown in the below figure. As we can observe, Logistic regression model along with SVM and Naive Bayes model performed satisfactory result from our dataset.

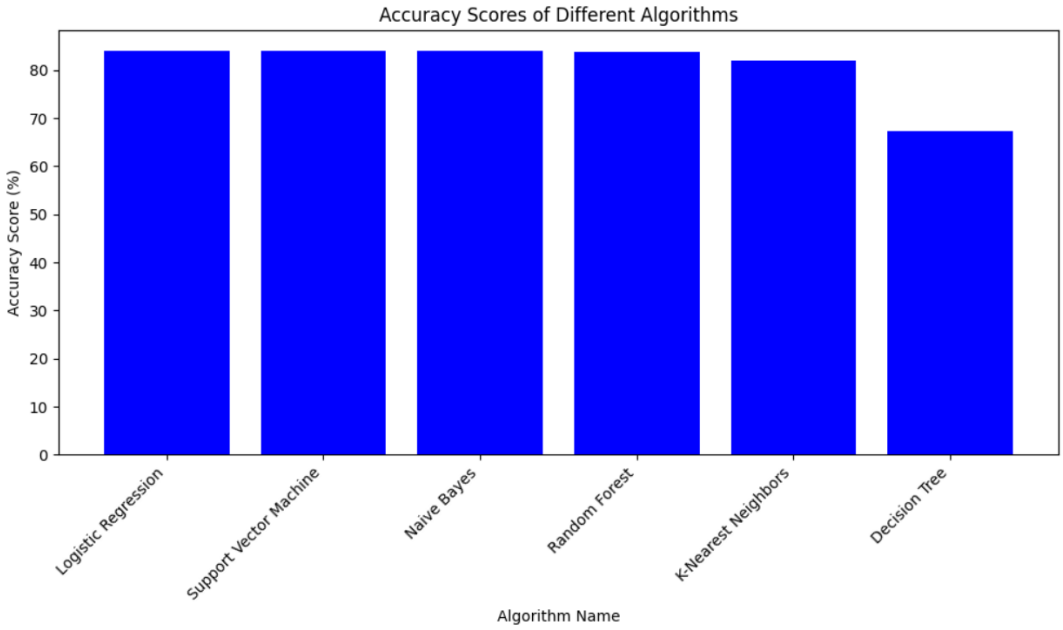

Figure 3: Graphical Representation of Accuracy

Conclusion:

This report underscores the significance of employing artificial intelligence for the detection of fraudulent UPI transactions. We put forth several classification algorithms capable of categorizing transactions based on specific features.

Following the training and testing phases, each chosen algorithm exhibited remarkable and consistent results, with accuracy scores consistently surpassing 75% in all cases. The test runs of the showcased algorithms led to the conclusion that all of them, to a similar extent, effectively handle the recognition of fraudulent transactions.

This project holds significance as it employs artificial intelligence for the detection of fraudulent banking transactions, a crucial aspect in the expanding online transaction market, especially in a nation like India. With over 70% of the 85 million MSME sellers connected to the internet, and an anticipated 80% internet penetration by 2027, the relevance of such initiatives is evident. Currently, 60% of MSMEs have embraced e-payments, including methods like net banking and UPI, with projections indicating a rise to 70% e-payment penetration by 2027.

References:

[1] Nitu Kumari, S. Kannan and A. Muthukumaravel - "Credit Card Fraud Detection Using Genetic-A Survey" published by MiddleEast Journal of Scientific Research, IDOSI Publications, 2014

[2] V. N. Dornadula and S. Geetha ―Credit Card Fraud Detection using Machine Learning Algorithms, Procedia Comput. Sci., vol. 165, pp. 631–641, 2019, doi: 10.1016/j.procs.2020.01.057.

[3] The python standard library (https://docs.python.org/3/library/index.html#the-python-standard-library)