PROJECT REPORT

COMPARISON OF DIFFERENT SUPERVISED MACHINE LEARNING ALGORITHM TO DETECT PAYMENT FRAUD

INDRAJIT PANDA	21009213
ROHAN ARORA	21041350
KHUSHI MEHTA	21008231

Introduction

Online digital payment is currently the backbone of any industry to make it more sustainable and scalable, as the Digital transaction continues to surge, the credit risk and fraudulent transaction has now become the critical concern for any financial institutions. Earlier using traditional approach to mitigate the risk associated was less efficient with consistent leakage. With advancement of technology and development of various machine learning algorithm, it has now become convenient and easier to detect such transaction. Machine learning algorithm provides an automated and safeguard approach to predict any fraudulent transaction and helps the financial institution to monitor it and take necessary action. This report illustrates a comprehensive investigation and compare the performance of various popular machine learning algorithm such as Logistic regression, k neighbors classifier decision tree classifier, random forest, light GBM, XGBoost, for identifying online digital payment fraud. The study compares and evaluates these algorithms using three distinct datasets, each representing a different aspect of digital payment fraud. To ensure the validity and diversity of the conclusion and efficacy of the algorithm, the three datasets are collected from real-world digital platforms having various transaction attributes, user behavior and fraud patterns. The research involves feature selection and engineering where attributes are carefully chosen and transformed to create robust predictive models. Subsequently, all the algorithm are trained on the preprocessed datasets and hyperparameter tuning are performed to optimize their performance. Further evaluation of models' performance is done using confusion matrices. By calculating key metrics such as accuracy, precision, recall, F1-score. Below Fig (i) shows the flow diagram on the entire research methodology performed.

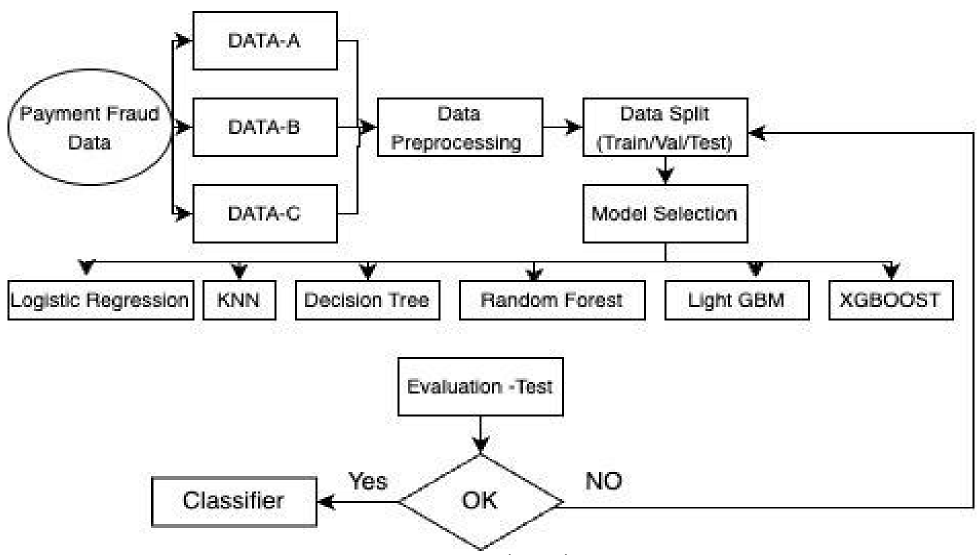


Fig – i

Literature Review

Supervised Machine Learning algorithm

Machine learning algorithms utilized programmed algorithm that learn and optimize their operations by analyzing input data to make prediction effective and accurate but withing the acceptable range. Supervised machine learning classification models play a pivotal role in addressing a wide array of real-world problems that involve predicting categorical outcomes. By leveraging labeled training data, these models learn to map input features to predefined classes, enabling accurate classification of new, unseen instances.

In this research, we have considered various supervised Machine Learning classifiers as well as regression models to understand the most suitable model for each type of transaction data. **Decision Tree**: The decision tree is a supervised machine learning predictive algorithm that consists of a flow-chart-like structure consisting of a root node that branches out to make various decisions at the internal nodes and provides all the possible outcomes at the leaf nodes. Decision trees are also used to classify data based on the various features and attributes present in the dataset by using entropy to segregate the data into different groups to reduce confusion and improve accuracy by generating correct predictions.

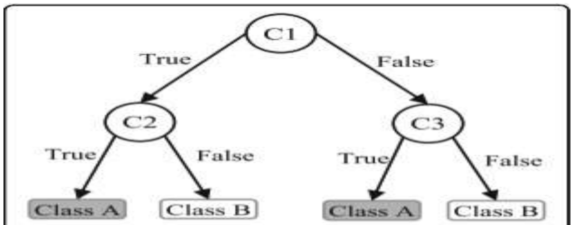


Fig. 3 An illustration of a Decision tree. Each variable (C1, C2, and C3) is represented by a circle and the decision outcomes (Class A and Class B) are shown by rectangles. In order to successfully classify a sample to a class, each branch is labelled with either 'True' or 'False' based on the outcome value from the test of its ancestor node

K – Nearest Neighbors: The K – nearest neighbors (KNN) is a supervised, predictive, and classifying machine learning algorithm that uses Euclidean, Manhattan, Cosine, and Jaccard distance metrics amongst many for data categorization, prediction, and creating labels for similar data points. Training and learning of dataset in advance, is non-essential in this algorithm which makes it an instance-based algorithm. KNN makes predictions based on the similarity of the test data points to its nearest neighbors in the training dataset.

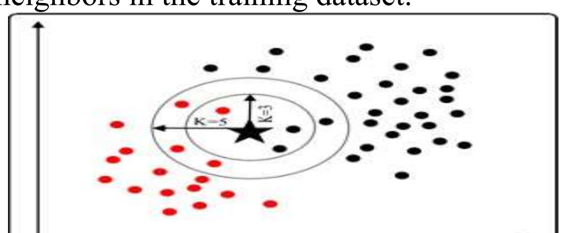


Fig. 6 A simplified illustration of the K-nearest neighbour algorithm. When K = 3, the sample object ('star') is classified as 'black' since it gets more 'vote' from the 'black' class. However, for K = 5 the same sample object is classified as 'red' since it now gets more 'vote' from the 'red' class

Random Forest: As the name suggests, it is a parallel processing of data assigned randomly to various decision trees, that combines multiple decisions formed by each random tree to arrive at the final decision. The randomness of using different subsets of training data in each decision tree makes this method more accurate to generate predictions. This method is also suitable to handle big and complex datasets since there are fewer chances of overfitting as it accommodates both categorical as well as numeric data.

Logistic Regression: Logistic regression is a classifying and predicting statistical model that analyses various factors and assigns weights to these factors to predict the probability or the likelihood of the outcome. The obtained probability can then be compared with the threshold value to determine the final decision. Since this model uses weights for various input factors, it makes this model flexible enough to develop relationships amongst these factors for further exploratory analysis.

Extreme Gradient Boosting Machine (XGBM): This machine learning algorithm is densely used as a classifier and regression model. XGBM uses various decision trees to rectify the errors made by the previous decision trees. This process continues to iterate until a position is reached where the trees are combined to generate the best output or provide the final prediction. The XGBM algorithm is also suitable to handle big and complex datasets due to it's high efficiency and accuracy.

Light Gradient Boosting Machine (LGBM): This algorithm as the name suggests also categorizes itself under the gradient boosting frameworks. LGBM like the above-mentioned XGBM also uses multiple decision trees to rectify the errors made by its predecessors until a single strong output is generated. The reason why this gradient boosting framework is accompanied by "Light" is due to its high efficiency, ability to handle large datasets, reduce memory usage as compared to other gradient boosting framework

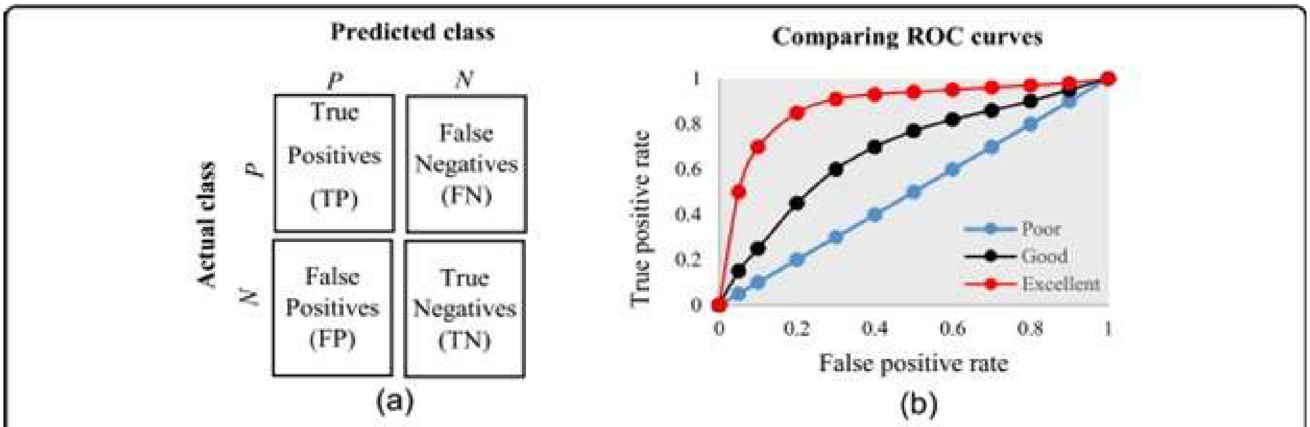


Fig. 11 a The basic framework of the confusion matrix and (**b**) A presentation of the ROC curve

Data Summary

Data-A

This dataset has been collected from Kaggle website; this contains historical information about fraudulent transactions which can be used to detect fraud in online payments. The dataset contains 6.3 million records having 10 attributes and 1 variable which contains flag 1 for fraud and 0 for not fraud. Below table represents the type of attributes and its basic description.

Variable Name	Variable description	DataType	Min	Mean	Max
Step	type of online transaction	varchar			
Type	type of online transaction	varchar			
Amount	the amount of the transaction	Float	0	179862	9244552
nameOrig	customer starting the transaction	varchar			
oldbalanceOrg	balance before the transaction	float	0	833883	5958504
newbalanceOrig	balance after the transaction	float	0	855113	4958504
nameDest	recipient of the transaction	varchar			
oldbalanceDest	initial balance of recipient before the transaction	float	0	1100702	3560159
newbalanceDest	the new balance of recipient after the transaction	float	0	1290820	3561793
isFraud	fraud transaction	boolean	Binary 1/0		
isFlaggedFraud	Flagged Fraud	boolean			

Data Cleaning, Pre-processing, Outlier Treatment and Exclusions and Transformation

At first, my objective was to clean the dataset. I closely examined each variable through visual inspection using Microsoft Excel and python and then performed the following steps.

Pre-cleaning Exercise

1) Cross checked in case any missing values and removed in case any.
2) Checked in case any junk character in the varchar column and removed.
3) Using boxplot, removed numbers which are completely irrelevant and out of the logic.

Normal distribution check

Using QQ plot and normal distribution curve we checked if the attributes are normally distributed or not, based on the curve we see that the skewness exists in the data, but as the target analysis is on identifying fraud transaction there would be existence of skewed values. Additionally considering the size of the dataset which is more than 30 we take an assumption that the values are normally distributed and proceed accordingly.

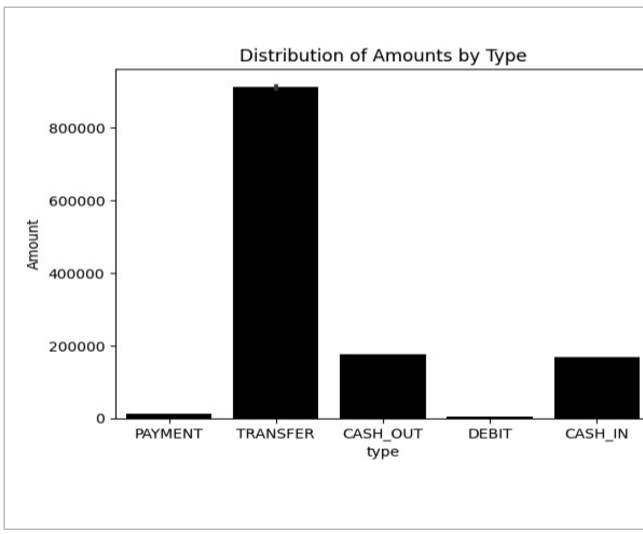
Exclusion and Transformation

To optimize the efficiency of the model performance, new variables have been defined and Some variables have been eliminated.

1) Removed Step attribute as all the row has value as 1 and do not add any information to the model.
2) New attribute "Original_balance_diff" has been added, this variable is the difference in amount based on old balance origination and new balance origination.

   Original_balance_diff= ['oldbalanceorg']-['newbalanceorig']
3) oldbalanceorg' and newbalanceorig' has been removed as new variable has been added.
4) New attribute "Destination_balance_diff" has been added, this variable is the difference in amount based on old balance origination and new balance origination.

   destination_balance_diff= ['oldbalanceDest']-['newbalanceDest']
5) In order to identify the destination where the transaction is done multiple times which can be a potential fraud account, we have defined a new variable as Frequency which has flag 1 when the number of transactions is more than 20 to that particular account.
6) Using One-hot encoding we have added attributes for each type of transaction i.e., CASH_IN, CASH_OUT, DEBIT, PAYMENT, TRANSFER .Type attribute has been removed.



Type_CASH_IN	Type_CASH_OUT	Type_DEBIT	Type_PAYMENT	Type_TRANS
0	0	0	1	
0	0	0	1	
0	0	0	0	
0	1	0	0	
0	0	0	1	

7) Created a binary flag as 'nameOrig_flag which determine 1 in case the origin starts with 'C' and 0 if it starts with 'M'. Similarly , Created a binary flag as 'nameDest_flag which determine 1 in case the origin starts with 'C' and 0 if it starts with 'M'.
8) To trigger large amount in a transaction which can be the one with high indicators of being fraud we have considered anything beyond 75^{th} percentile of the data as surge indicator.

Correlation Matrix

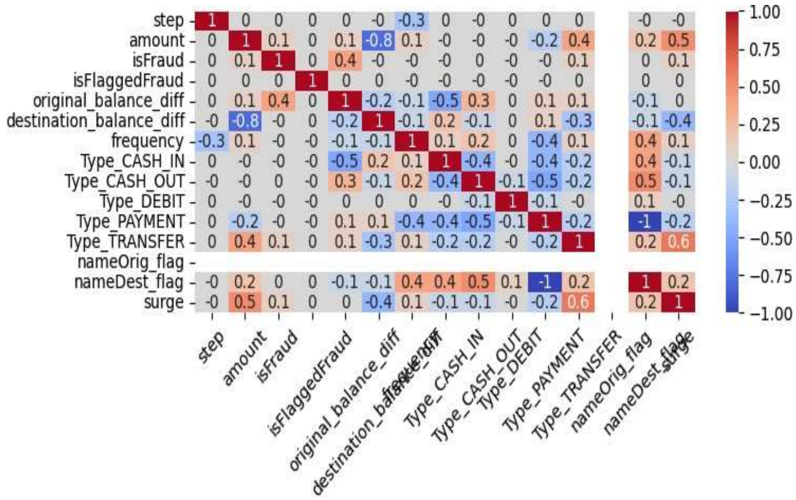


Fig -ii

Fig-ii shows the correlation matrix between the predictor variables.
From the matrix it is evident that amount is negatively correlated with original balance diff with correlation parameter as -0.8. Similarly, destination balance diff is negatively correlated with amount with correlation parameter as -0.8. Hence amount attribute have been removed.

Methodology

Data-A

After the pre-processing and correlation check is done, the final dataset has 10 attributes and 1 target variable as below.

Predictor Variable	isFlaggedFraud	original_balance_diff	destination_balance_diff
	Type_CASH_IN	Type_CASH_OUT	Type_DEBIT
	Type_PAYMENT	Type_TRANSFER	nameOrig_flag
	nameDest_flag		
Target Variable	isFraud		

Target Variable distribution

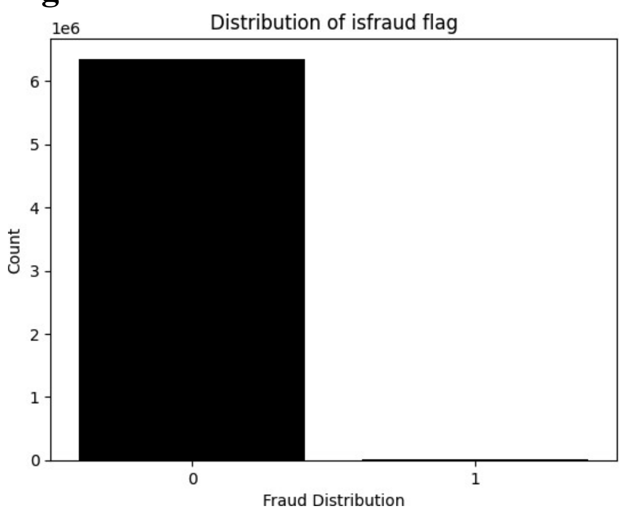


Fig -III

Fig-III shows the distribution of the target variable is Fraud

Target Variable	1	0
isFraud	8213	6354407

Based on the distribution it is clear that the target variable is highly unbalanced. In order to address the class imbalance in the target variable and improve the predictive performance of the model, the Synthetic Minority Over-sampling Technique (SMOTE) was employed to effectively oversample the data, ensuring a more representative and balanced distribution of the target classes.

Training and Validation of the models

Further with the balanced dataset, we split the data into three section Train -60%, Validation -20% and Test -20% to ensure generalisation of the model. We have used four different classification algorithms: Decision Tree, K-Nearest Neighbours, Random Forest, and Logistic Regression. Each model was trained using the resampled training data generated through SMOTE. The validation set was used for fine-tuning the models to improve their performance.
Additionally, two ensemble models, XGBoost and LightGBM, were trained to further improve the fraud detection accuracy. To optimize the hyperparameters of the XGBoost model, Randomized Search with cross-validation was performed, considering hyperparameters like the number of estimators, maximum depth, and learning rate.

Please refer the below code snippet and the appendix section for the Python Code.

Classification Model

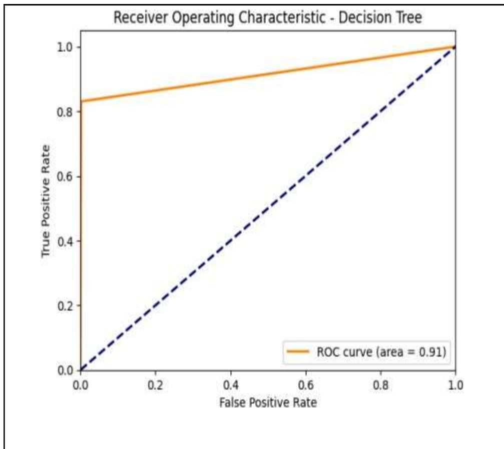
```python
# This one function can be used to do the whole metrics process we tried in Notebook #1
def Model_with_SMOTE(df):
    # Split the dataframe into train, test, and validation sets
    X = df.drop(['isFraud','amount'], axis=1)
    y = df['isFraud']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
    X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=42)

    # Use SMOTE technique to resample the unbalanced data in the training set
    smote = SMOTE(random_state=42)
    X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

    # Fit models using DecisionTreeClassifier, KNeighborsClassifier, and RandomForestClassifier
    models = [
        ("Decision Tree", DecisionTreeClassifier()),
        ("K-Nearest Neighbors", KNeighborsClassifier()),
        ("Random Forest", RandomForestClassifier()),
        ("LogisticRegression", LogisticRegression())
    ]

    for name, model in models:
        print(f"Training {name}...")
        model.fit(X_train_resampled, y_train_resampled)

        # Fine-tune the model using the validation dataset
        y_val_pred = model.predict(X_val)

        # Predict using the test dataset
        y_test_pred = model.predict(X_test)
```

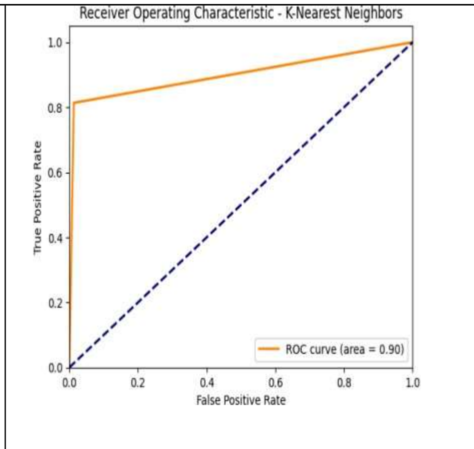
Ensemble Model

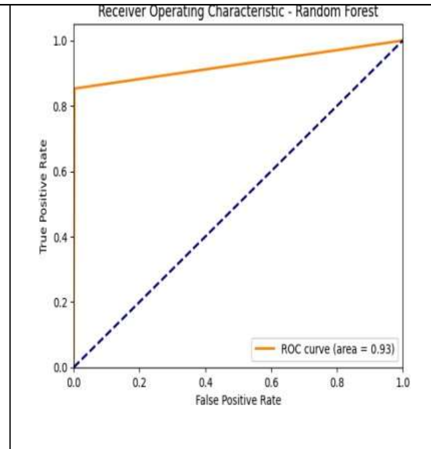
```python
# Step 2: Fit models and predict
def fit_xgboost_light_GBM_predict(X_train, X_val, X_test, y_train, y_val, y_test):
    # XGBoost
    xgb_model = xgb.XGBClassifier()
    xgb_model.fit(X_train, y_train)
    xgb_pred = xgb_model.predict(X_test)

    # LightGBM
    lgb_model = lgb.LGBMClassifier()
    lgb_model.fit(X_train, y_train)
    lgb_pred = lgb_model.predict(X_test)

    # Randomized Search for XGBoost
    xgb_params = {
        'n_estimators': [100, 200, 300],
        'max_depth': [3, 5, 7],
        'learning_rate': [0.1, 0.01, 0.001]
    }
    xgb_random_search = RandomizedSearchCV(xgb_model, xgb_params, n_iter=10, scoring='accuracy', cv=3)
    xgb_random_search.fit(X_train, y_train)
    xgb_random_pred = xgb_random_search.predict(X_test)

    return xgb_pred, lgb_pred,xgb_random_pred

# You can choose any prediction array here
```

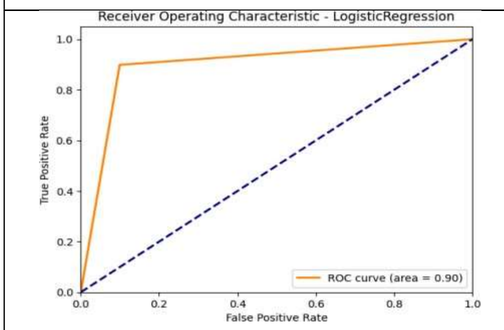
Result

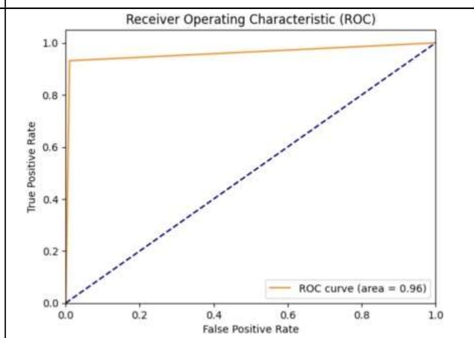
ROC curve and the Classification report for Data -A are as follows: -

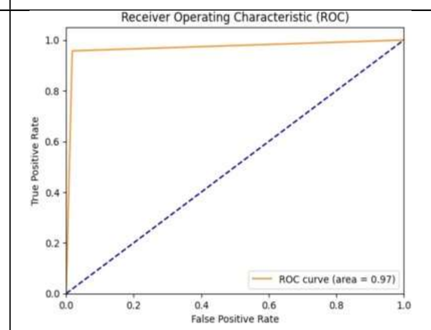












Classification Report for Decision Tree:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1906351
1	0.36	0.83	0.50	2435
accuracy			1.00	1908786
macro avg	0.68	0.91	0.75	1908786
weighted avg	1.00	1.00	1.00	1908786

Classification Report for K-Nearest Neighbors:

	precision	recall	f1-score	support
0	1.00	0.99	0.99	1906351
1	0.07	0.81	0.14	2435
accuracy			0.99	1908786
macro avg	0.54	0.90	0.56	1908786
weighted avg	1.00	0.99	0.99	1908786

Classification Report for Random Forest:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1906351
1	0.31	0.85	0.45	2435
accuracy			1.00	1908786
macro avg	0.65	0.93	0.72	1908786
weighted avg	1.00	1.00	1.00	1908786

Classification Report for LogisticRegression:

	precision	recall	f1-score	support
0	1.00	0.90	0.95	1906351
1	0.01	0.90	0.02	2435
accuracy			0.90	1908786
macro avg	0.51	0.90	0.49	1908786
weighted avg	1.00	0.90	0.95	1908786

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.99	0.99	1270904
1	0.10	0.93	0.19	1620
accuracy			0.99	1272524
macro avg	0.55	0.96	0.59	1272524
weighted avg	1.00	0.99	0.99	1272524

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.98	0.99	1270904
1	0.06	0.96	0.12	1620
accuracy			0.98	1272524
macro avg	0.53	0.97	0.56	1272524
weighted avg	1.00	0.98	0.99	1272524

Confusion Matrix

Predicted Values \ Actual Values	Yes	No
Yes	1902737	3614
No	413	2022

Decision Tree

Predicted Values \ Actual Values	Yes	No
Yes	1881473	24878
No	454	1981

K-Nearest Neighbour

Predicted Values \ Actual Values	Yes	No
Yes	1901625	4726
No	358	2077

Random Forest

Predicted Values \ Actual Values	Yes	No
Yes	1717027	189324
No	247	2188

Logistic Regression

Predicted Values \ Actual Values	Yes	No
Yes	1257721	13183
No	111	1509

XGBOOST

Predicted Values \ Actual Values	Yes	No
Yes	1248273	22631
No	69	1551

Light GBM

Data B

This dataset has been collected from data. World website, and contains credit card transactions done by European cardholders within a duration of 2 days in September 2013. The dataset contains more than 2 lakh 80 thousand records having 30 attributes and 1 attribute named 'Class' that contains 0 for non-fraud and 1 for fraud transactions. The data contains only numeric values that have resulted from PCA transformation, and due to confidentiality, the original features are not made available to the public. The 2 features that have been in the original state are 'Time' and 'Amount'. The table showcased below comprises the attributes and their basic description

Variable Name	Datatype	count	mean	min	max
Time	float64	284807	94813.85958	0	172792
V1	float64	284807	1.17E-15	-56.4075096	2.45492999
V2	float64	284807	3.42E-16	-72.7157276	22.057729
V3	float64	284807	-1.38E-15	-48.3255894	9.38255843
V4	float64	284807	2.07E-15	-5.6831712	16.875344
V5	float64	284807	9.60E-16	-113.743307	34.8016659
V6	float64	284807	1.49E-15	-26.1605059	73.3016255
V7	float64	284807	-5.56E-16	-43.5572416	120.589494
V8	float64	284807	1.21E-16	-73.2167185	20.0072084
V9	float64	284807	-2.41E-15	-13.4340663	15.5949946
V10	float64	2.85E+05	2.24E-15	-2.46E+01	2.37E+01
V11	float64	2.85E+05	1.67E-15	-4.80E+00	1.20E+01

V12	float64	2.85E+05	-1.25E-15	-1.87E+01	7.85E+00
V13	float64	2.85E+05	8.19E-16	-5.79E+00	7.13E+00
V14	float64	2.85E+05	1.21E-15	-1.92E+01	1.05E+01
V15	float64	2.85E+05	4.89E-15	-4.50E+00	8.88E+00
V16	float64	2.85E+05	1.44E-15	-1.41E+01	1.73E+01
V17	float64	2.85E+05	-3.77E-16	-2.52E+01	9.25E+00
V18	float64	2.85E+05	9.56E-16	-9.50E+00	5.04E+00
V19	float64	2.85E+05	1.04E-15	-7.21E+00	5.59E+00
V20	float64	-0.211721365	-0.062481092	284807	0.77092502
V21	float64	-0.228394947	-0.029450168	284807	0.73452401
V22	float64	-0.542350373	0.006781943	284807	0.72570156
V23	float64	-0.161846345	-0.01119293	284807	0.6244603
V24	float64	-0.354586136	0.040976056	284807	0.60564707
V25	float64	-0.317145054	0.016593502	284807	0.52127807
V26	float64	-0.326983926	-0.052139108	284807	0.48222701
V27	float64	-0.070839529	0.001342146	284807	0.40363249
V28	float64	-0.052959793	0.011243832	284807	0.33008326
Amount	float64	284807	88.34961925	0	25691.16
Class	int64	284807	0.001727486	0	1

Data Cleaning, Pre-processing, Outlier Treatments

To check the quality of the dataset, the first step was to identify if there are any missing values in the dataset. In this dataset, no missing values were identified. A ratio of fraud to non-fraud transactions was done, with a value of 0.173% concluding that the data was highly imbalanced with only 492 fraud and 284315 non-fraud cases respectively.

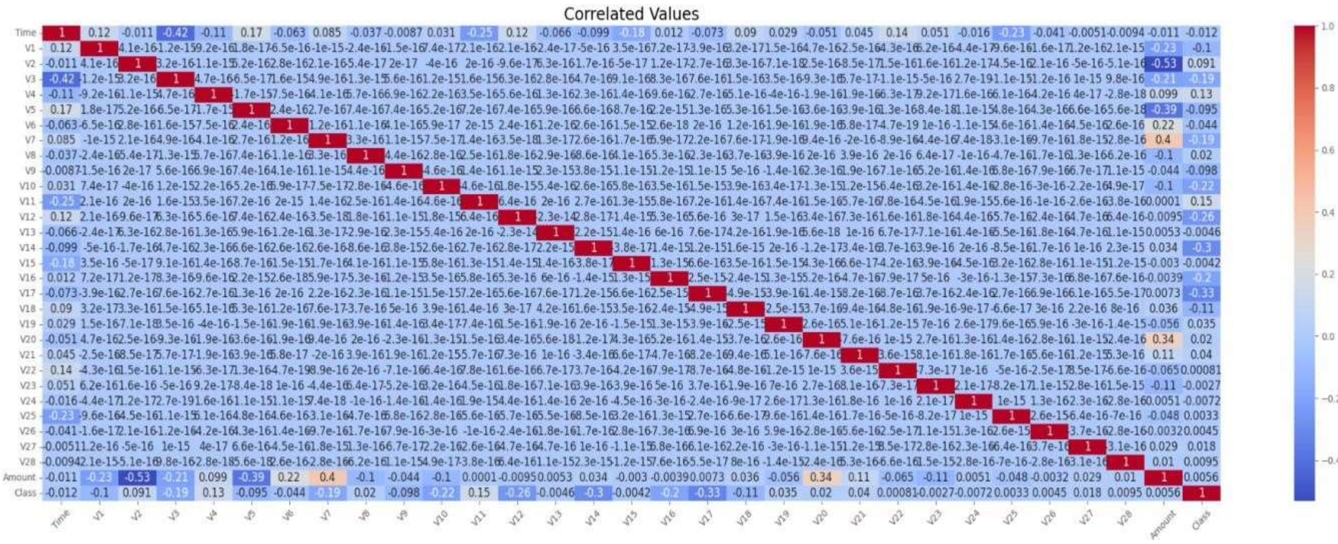
Normal distribution check:

Using the QQ plot and normal distribution curve to check if the attributes are normally distributed or not, it was observed that the data were highly skewed, but since the target analysis was on identifying fraud transactions, there would be the existence of skewed values. Furthermore, taking into consideration the size of the dataset, which is more than 30, it could be assumed that the values are normally distributed and hence, further operations have been carried out accordingly.

Exclusion and Transformation:

Since, the data has already been PCA transformed, and all the values were numeric, no further transformation was done. Additionally, as mentioned above, since the properties of variables were unknown due to confidentiality, all the variables were considered to generate the co-relation matrix to detect the outliers with the target variable as ‘Class’.

Correlation Matrix:



In this case, it can be observed that amount and V2 are negatively correlated with a correlation parameter of -0.53. If V2 is removed amongst the two, there is no change in the fraud to non-fraud ratio, however, when the amount attribute is removed, we observe that there is a minute change in the fraud to non-fraud ratio.

Methodology:

From the outlier detection and removal, it could be observed that the number of fraud cases reduced from 492 to 418 and the non-fraud ones reduced from 284315 to 261754. This states that the ratio of fraud to non-fraud post outlier removal is 0.16% which previously was 0.173%. Hence, no marginal difference was observed.

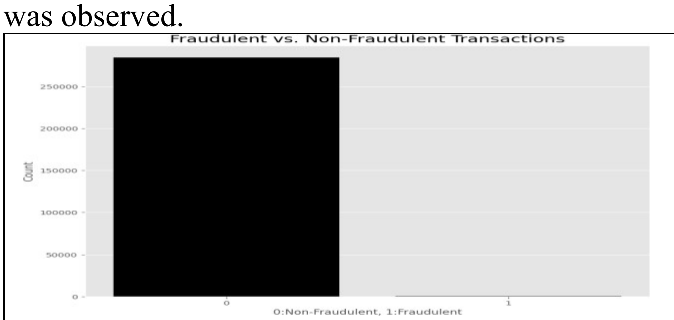


Fig. 3(b) shows the distribution of fraud v/s non-fraud transactions

Based on the distribution, the target variable is highly unbalanced. To address this high imbalance in the target variable and to improve the predictive performance of the models, a Synthetic Minority Over-sampling Technique (SMOTE) was employed to effectively oversample the data, to ensure a balanced distribution of the target class.

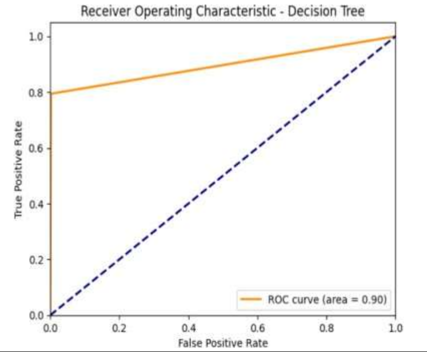
<u>Training and Validation of the models</u>

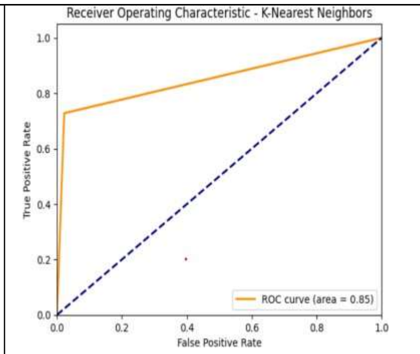
Once a balanced dataset was generated, the data was split into three sections namely training (60%), validation (20%), and testing (20%) set, to ensure the generalization of the model. To draw fair comparisons, four classification algorithms were considered such as decision tree, K-nearest neighbour (KNN), random forest, and logistic regression. Each model was trained using the resampled training data generated through SMOTE. The validation set was used for fine-tuning the models to improve their performance.

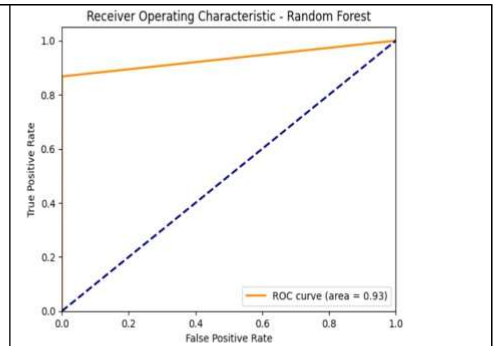
Additionally, two ensemble models, XGBoost and LightGBM, were trained to further improve fraud detection accuracy. To optimize the hyperparameters of the XGBoost model, a randomized search with cross-validation was performed, considering hyperparameters like the number of estimators, maximum depth, and learning rate.

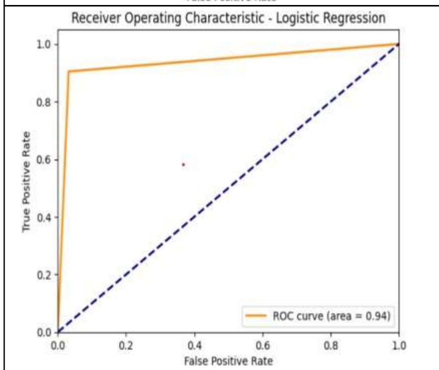
Result:

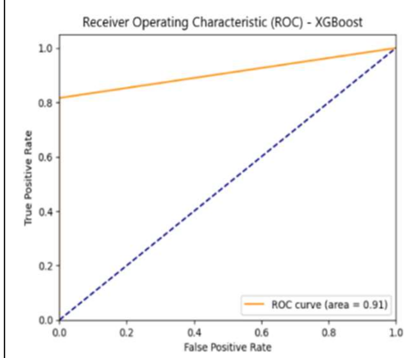
ROC curve and the Classification report for Data -B are as follows: -

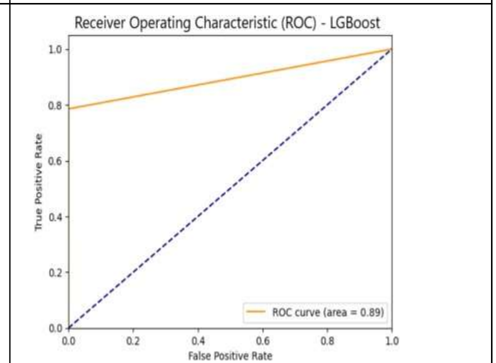












Classification Report for Decision Tree

	precision	recall	f1-score	support
0	1.00	1.00	1.00	85307
1	0.41	0.79	0.54	136
accuracy			1.00	85433
macro avg	0.71	0.9	0.77	85433
weighted avg	1.00	1.00	1.00	85433

Classification Report for K-Nearest Neighbours

	precision	recall	f1-score	support
0	1.00	0.98	0.99	85307
1	0.05	0.73	0.09	136
accuracy			0.98	85433
macro avg	0.52	0.85	0.54	85433
weighted avg	1.00	0.98	0.99	85433

Classification Report for Random Forest

	precision	recall	f1-score	support
0	1.00	1.00	1.00	85307
1	0.81	0.87	0.84	136
accuracy			1.00	85433
macro avg	0.91	0.93	0.92	85433
weighted avg	1.00	1.00	1.00	85433

Classification Report for Logistic Regression

	precision	recall	f1-score	support
0	1.00	0.97	0.98	85307
1	0.04	0.9	0.08	136
accuracy			0.97	85433
macro avg	0.52	0.94	0.53	85433
weighted avg	1.00	0.97	0.98	85433

Classification Report for XGBoost

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.98	0.82	0.89	98
accuracy			1.00	56962
macro avg	0.99	0.91	0.94	56962
weighted avg	1.00	1.00	1.00	56962

Classification Report for LGBoost

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.89	0.79	0.83	98
accuracy			1.00	56962
macro avg	0.94	0.89	0.92	56962
weighted avg	1.00	1.00	1.00	56962

Confusion Matrix

		Actual Values: Yes	Actual Values: No
Predicted Values	Yes	85152	155
	No	28	108

		Actual Values: Yes	Actual Values: No
Predicted Values	Yes	83408	1899
	No	37	99

		Actual Values: Yes	Actual Values: No
Predicted Values	Yes	85280	27
	No	18	118

		Actual Values: Yes	Actual Values: No
Predicted Values	Yes	82597	2710
	No	13	123

		Actual Values: Yes	Actual Values: No
Predicted Values	Yes	56862	2
	No	18	80

		Actual Values: Yes	Actual Values: No
Predicted Values	Yes	56854	10
	No	21	77

DATA – C

The dataset for default of credit card clients in Taiwan [1] contains 30,000 instances spread across 25 attributes constituting 1 outcome variable "def_pay" which specifies the default payments and 24 predictor variables that enlist the demographic characteristics, credit data, payment history, and bill statements of credit card clients in Taiwan from September 2005 to April 2005.

Attribute	Attribute description	Datatype	Min	Max	Mean	Std Deviation
ID	ID of each client	Integer	1	30000	15000.5	8660.398374
LIMIT_BAL	Amount of given credit in NT dollars (includes individual and family/supplementary credit	Float	10000	1000000	167484.3227	129747.6616
SEX	Gender (1=male, 2=female)	Integer	1	2	1.603733333	0.489129196
EDUCATION	Level of education (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)	Integer	0	6	1.853133333	0.79034866
MARRIAGE	Marital status (1=married, 2=single, 3=others)	Integer	0	3	1.551866667	0.521969601
AGE	Age in years	Integer	21	79	35.4855	9.217904068
PAY_1	Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, … 8=payment delay for eight months, 9=payment delay for nine months and above)	Integer	-2	8	-0.0167	1.123801528
PAY_2	Repayment status in August, 2005 (scale same as above)	Integer	-2	8	-0.133766667	1.197185973
PAY_3	Repayment status in July, 2005 (scale same as above)	Integer	-2	8	-0.1662	1.196867568
PAY_4	Repayment status in June, 2005 (scale same as above)	Integer	-2	8	-0.220666667	1.169138622
PAY_5	Repayment status in May, 2005 (scale same as above)	Integer	-2	8	-0.2662	1.133187406
PAY_6	Repayment status in April, 2005 (scale same as above)	Integer	-2	8	-0.2911	1.149987626
BILL_AMT1	Amount of bill statement in September, 2005 (NT dollar)	Float	-165580	964511	51223.3309	73635.86058
BILL_AMT2	Amount of bill statement in August, 2005 (NT dollar)	Float	-69777	983931	49179.07517	71173.76878
BILL_AMT3	Amount of bill statement in July, 2005 (NT dollar)	Float	-157264	1664089	47013.1548	69349.38743
BILL_AMT4	Amount of bill statement in June, 2005 (NT dollar)	Float	-170000	891586	43262.94897	64332.85613
BILL_AMT5	Amount of bill statement in May, 2005 (NT dollar)	Float	-81334	927171	40311.40097	60797.15577
BILL_AMT6	Amount of bill statement in April, 2005 (NT dollar)	Float	-339603	961664	38871.7604	59554.10754
PAY_AMT1	Amount of previous payment in September, 2005 (NT dollar)	Float	0	873552	5663.5805	16563.28035
PAY_AMT2	Amount of previous payment in August, 2005 (NT dollar)	Float	0	1684259	5921.1635	23040.8704
PAY_AMT3	Amount of previous payment in July, 2005 (NT dollar)	Float	0	896040	5225.6815	17606.96147
PAY_AMT4	Amount of previous payment in June, 2005 (NT dollar)	Float	0	621000	4826.076867	15666.15974
PAY_AMT5	Amount of previous payment in May, 2005 (NT dollar)	Float	0	426529	4799.387633	15278.30568
PAY_AMT6	Amount of previous payment in April, 2005 (NT dollar)	Float	0	528666	5215.502567	17777.46578
def_pay	Default payment (1=yes, 0=no)	Integer	0	1	0.2212	0.415061806

Data Cleaning, Pre-processing, Outlier Treatments:

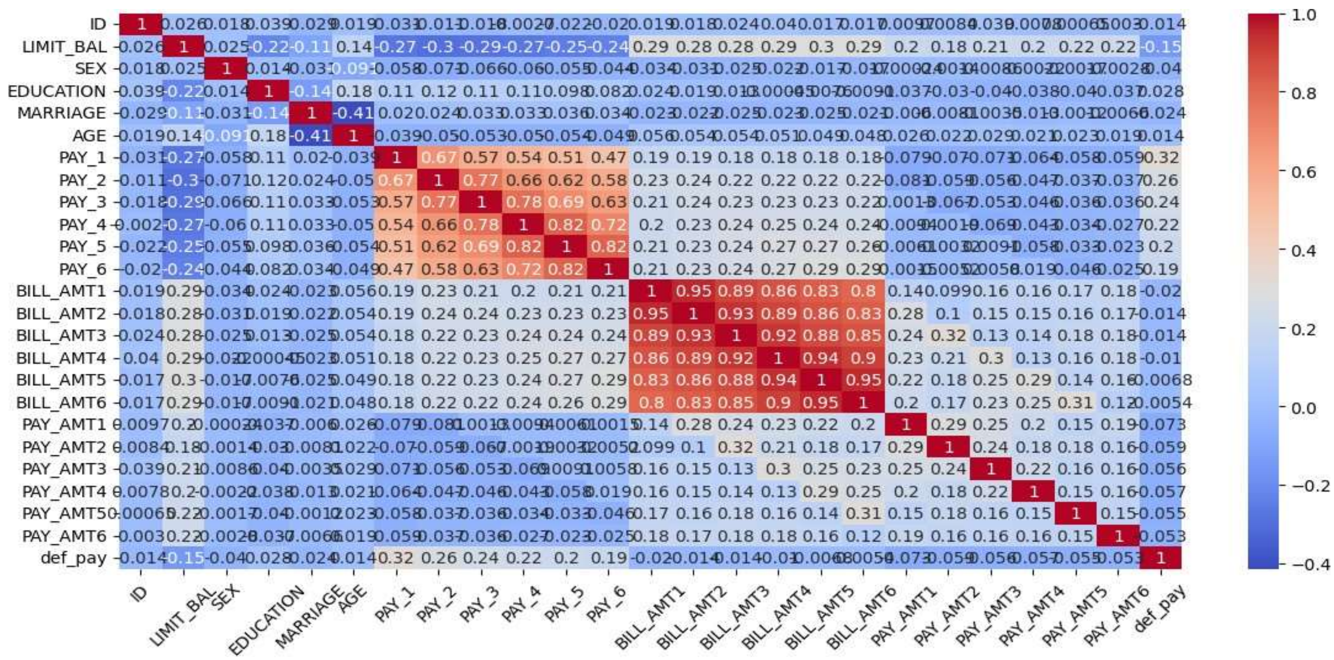
There wasn't any missing data for this dataset. To mitigate any issues that might arise downstream, I renamed my outcome variable from "default.payment.next.month" to "def_pay", and corrected sequence issues by renaming "PAY_0" To "PAY_1".

Normal distribution check:

I checked the normality of data by visual inspection of histogram and Q-Q plots and based on the curves found signs of skewness in some of the attributes, although considering our data is related to credit card payments, it's natural to see some skewness. Furthermore, the size of our dataset is greater than 30, thus as per considerations provided under the Central Limit Theorem, we go ahead with our assumption of normally distributed data.

Through the various plots and distributions, I was able to interpret a 78% v/s 22% data distribution among those who default payments and those who don't, which signified no signs of class imbalance in our dataset. Additionally, upon performing an outlier analysis, we couldn't see much variability in our data thus we proceed ahead without removing any outliers.

Correlation Matrix:



The above figure shows the correlation matrix among attributes of the dataset. Since the PAY_*, BILL_AMT*, and PAY_AMT* are similar kinds of features thus they are found to be significantly correlated with each other. Therefore, even after the visible presence of collinearity, we cannot blindly remove these features as they are important for the final model prediction. Hence, we proceed with our full dataset.

Methodology

Post data pre-processing, normality check, and correlation analysis, the final dataset comprises 24 predictor variables and 1 outcome variable, as enlisted in the below table:

Outcome	Predictor variables
default_payment_next_month	ID, LIMIT_BAL, SEX, EDUCATION, MARRIAGE, AGE, PAY_1, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6, BILL_AMT1, BILL_AMT2, BILL_AMT3, BILL_AMT4, BILL_AMT5, BILL_AMT6, PAY_AMT1, PAY_AMT2, PAY_AMT3, PAY_AMT4, PAY_AMT5, PAY_AMT6

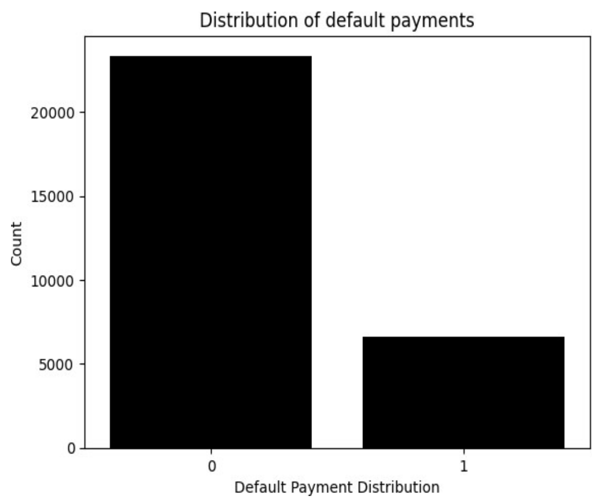


Fig -III

Fig-III shows our outcome variable's default v/s non-default payment distribution (default_payment_next_month).

Outcome variable	1	0
default_payment_next_month	6636	23364

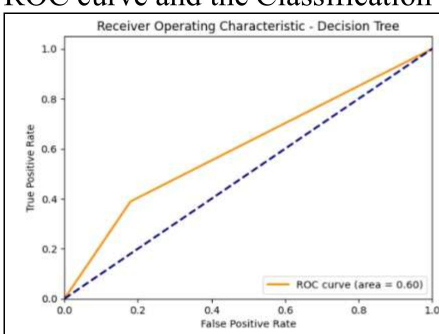
Based on the distribution, it is evident that the target variable is moderately balanced, in the ratio of 78% to 22%.

Training and Validation of the models

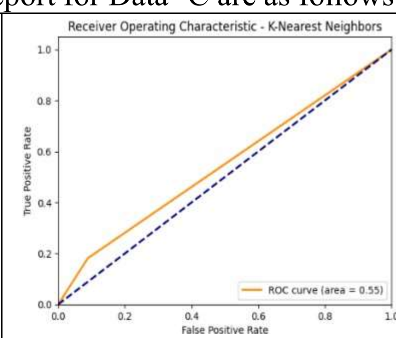
The dataset was split into three sections namely training (60%), validation (20%), and testing (20%) to ensure generalisation of the model. We have used four different classification algorithms: Decision Tree, K-Nearest Neighbours, Random Forest, and Logistic Regression. The validation set was used for fine-tuning the models to improve their performance. Additionally, two ensemble models, XGBoost and LightGBM, were trained to further improve fraud detection accuracy. To optimize the hyperparameters of the XGBoost model, a Randomized Search with cross-validation was performed, considering hyperparameters like the number of estimators, maximum depth, and learning rate.

Result

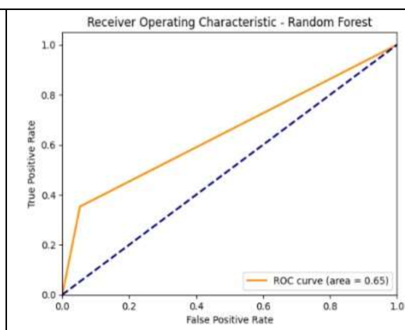
ROC curve and the Classification report for Data -C are as follows: -



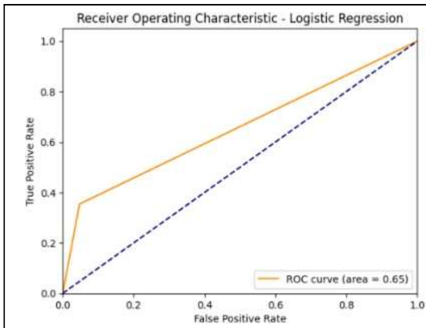
ROC – Decision Tree



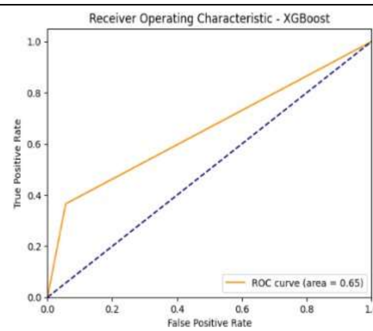
ROC - KNN



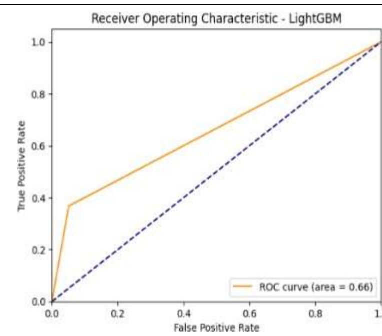
ROC – Random Forest



ROC – Logistic Regression



ROC - XGBoost



ROC - LightGBM

```
Classification Report for Decision Tree:
              precision    recall  f1-score   support

           0       0.83      0.82      0.82      4687
           1       0.38      0.39      0.38      1313

    accuracy                           0.73      6000
   macro avg       0.60      0.60      0.60      6000
weighted avg       0.73      0.73      0.73      6000
```

```
Classification Report for K-Nearest Neighbors:
              precision    recall  f1-score   support

           0       0.80      0.91      0.85      4687
           1       0.36      0.18      0.24      1313

    accuracy                           0.75      6000
   macro avg       0.58      0.55      0.55      6000
weighted avg       0.70      0.75      0.72      6000
```

```
Classification Report for Random Forest:
              precision    recall  f1-score   support

           0       0.84      0.95      0.89      4687
           1       0.65      0.35      0.46      1313

    accuracy                           0.82      6000
   macro avg       0.75      0.65      0.67      6000
weighted avg       0.80      0.82      0.80      6000
```

```
Classification Report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.84      0.95      0.89      4687
           1       0.68      0.35      0.47      1313

    accuracy                           0.82      6000
   macro avg       0.76      0.65      0.68      6000
weighted avg       0.80      0.82      0.80      6000
```

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

           0       0.84      0.94      0.89      4687
           1       0.65      0.37      0.47      1313

    accuracy                           0.82      6000
   macro avg       0.74      0.65      0.68      6000
weighted avg       0.80      0.82      0.80      6000
```

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

           0       0.84      0.95      0.89      4687
           1       0.67      0.37      0.47      1313

    accuracy                           0.82      6000
   macro avg       0.76      0.66      0.68      6000
weighted avg       0.80      0.82      0.80      6000
```

Confusion Matrix

Decision Tree		Actual Values Yes	Actual Values No
Predicted Values	Yes	3844	843
	No	802	511

KNN		Actual Values Yes	Actual Values No
Predicted Values	Yes	4271	416
	No	1075	238

Random Forest		Actual Values Yes	Actual Values No
Predicted Values	Yes	4440	247
	No	850	463

Logistic Regression		Actual Values Yes	Actual Values No
Predicted Values	Yes	4461	226
	No	852	461

XGBoost		Actual Values Yes	Actual Values No
Predicted Values	Yes	4424	263
	No	833	480

LightGBM		Actual Values Yes	Actual Values No
Predicted Values	Yes	4448	239
	No	830	483

Overall Conclusion for all the Three Dataset A,B and C are as follows

Accuracy	Logistic Regession	Decision Tree	Random Forest	KNN	Light GBM	XGBOOST
Data-A	0.9006	0.9978	0.9973	0.9867	0.9821	0.9895
Data-B	0.9681	0.9979	0.9995	0.9773	0.997	0.999
Data-C	0.82	0.723	0.8161	0.7515	0.8218	0.8173

Precision	Logistic Regession	Decision Tree	Random Forest	KNN	Light GBM	XGBOOST
Data-A	0.0114	0.3587	0.3053	0.0737	0.064	0.102
Data-B	0.0434	0.4106	0.8138	0.0495	0.431	0.987
Data-C	0.6768	0.3719	0.6454	0.3639	0.6689	0.646

Recall	Logistic Regession	Decision Tree	Random Forest	KNN	Light GBM	XGBOOST
Data-A	0.8985	0.8303	0.8529	0.8135	0.957	0.9314
Data-B	0.9044	0.7941	0.8676	0.7279	0.642	0.806
Data-C	0.3541	0.3861	0.3549	0.1812	0.3678	0.3655

F1-Score	Logistic Regession	Decision Tree	Random Forest	KNN	Light GBM	XGBOOST
Data-A	0.0225	0.501	0.4496	0.1352	0.12	0.185
Data-B	0.0829	0.5414	0.0928	0.8399	0.516	0.887
Data-C	0.465	0.3789	0.4579	0.2419	0.4746	0.4669

ROC	Logistic Regession	Decision Tree	Random Forest	KNN	Light GBM	XGBOOST
Data-A	0.9	0.91	0.93	0.9	0.96	0.97
Data-B	0.94	0.89	0.93	0.85	0.89	0.91
Data-C	0.65	0.6	0.65	0.55	0.66	0.65

Kappa	Logistic Regession	Decision Tree	Random Forest	KNN	Light GBM	XGBOOST
Data-A	0.02	0.5001	0.4486	0.1332	0.118	0.183
Data-B	0.0801	0.5404	0.8396	0.0901	0.515	0.887
Data-C	0.3703	0.2007	0.3583	0.1129	0.3781	0.3667

Based on the results obtained from the machine learning algorithms applied to the three datasets (Data-A, Data-B, and Data-C) for fraud identification in the banking industry, we can draw the following conclusions:

CONCLUSION

Accuracy

- Random Forest and XGBoost algorithms consistently demonstrate the highest accuracy levels across all datasets. They achieve accuracy values above 99% for Data-B and competitive accuracy for Data-A.
- Logistic Regression, Light GBM, and Decision Tree algorithms also provide relatively high accuracy, exceeding 97% on most datasets.
- K-Nearest Neighbors (KNN) performs reasonably well but shows lower accuracy compared to the top-performing algorithms

Precision and Recall

- For Data-A and Data-B, Light GBM and XGBoost consistently achieve the highest precision and recall values, indicating their ability to correctly classify both fraud and non-fraud instances effectively.
- Decision Tree and Random Forest algorithms show competitive results on precision and recall for Data-A and Data-B, but they are outperformed by Light GBM and XGBoost.

F1-Score

- XGBoost and Decision Tree algorithms demonstrate the highest F1-Score on most datasets, with XGBoost performing the best on Data-B and Decision Tree on Data-C.
- Logistic Regression and Light GBM show moderate F1-Score values, while KNN performs relatively poorly on all datasets.

ROC Curve

- Light GBM and XGBoost consistently demonstrate higher ROC values across all datasets, indicating their superior ability to distinguish between fraud and non-fraud cases.
- Decision Tree and Random Forest algorithms also perform well in terms of ROC on most datasets.
- Logistic Regression and KNN exhibit lower ROC scores, suggesting their limited effectiveness in handling this specific problem.

In conclusion, for fraud identification in the banking industry, XGBoost and Light GBM are recommended as the top-performing algorithms, followed by Decision Tree and Random Forest. Logistic Regression and KNN may not be the best choices for this specific problem, given their relatively weaker performance in comparison. Researchers and practitioners should consider these findings when selecting an appropriate machine learning model for their fraud detection systems.

Random Forest → Decision Tree → LIGHT GBM → XGBOOST

Future Scope and Recommendations

There are several potential future scopes and directions for further work in the field of fraud identification in the banking industry.

- **Ensemble Methods and Model Stacking**: Since Random Forest and XGBoost have shown strong performance individually, consider exploring the possibility of combining their predictions using ensemble methods like Voting or Stacking. This could potentially lead to further improvements in accuracy and overall model performance.
- **Feature Engineering and Selection**: Investigate more advanced feature engineering techniques and feature selection methods to identify the most relevant predictors for fraud detection. Utilizing domain knowledge and incorporating new data sources could lead to enhanced model performance.
- **Handling Imbalanced Data**: Address the issue of imbalanced data in the datasets, especially evident in Data-C, by applying various techniques such as oversampling, under sampling, or using advanced algorithms like SMOTE (Synthetic Minority Over-sampling Technique) to balance the class distribution and improve the performance of the models.

References

- Online Payment Fraud Detection. (2022, October 26). Kaggle. https://www.kaggle.com/datasets/jainilcoder/online-payment-fraud-detection
- Default of credit card clients dataset. (2016, November 3). Kaggle. https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset
- Prediction of default payment of credit card clients using Data Mining Techniques. (2019, June 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/8950597
- Predictive analytics for default of credit card clients. (2021, January 21). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9378671
- Srohit. (n.d.). ML-Misc/FraudDetection/Credit Card Fraud Detection.ipynb at master · srohit0/ML-Misc. GitHub. https://github.com/srohit0/ML-Misc/blob/master/FraudDetection/Credit%20Card%20Fraud%20Detection.ipynb
- Baghel, V. S. (2021, December 9). Math behind GBM and XGBoost - Analytics Vidhya - Medium. Medium. https://medium.com/analytics-vidhya/math-behind-gbm-and-xgboost-d00e8536b7de
- UCI Machine Learning Repository. (n.d.). https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients