Credit Card Fraud Detection classifier using ML and Deep learning

A Project Report submitted to Manipal Academy of Higher Education in partial fulfilment ofthe requirements for the award of the degree of

of

BACHELOR OF TECHNOLOGY

in

Computer and communication engineering

Submitted by

Nishad Chaoji
170953046

Under the guidance of
Dr Smitha N Pai
H.O.D and Guide
ICT department,
MIT, Manipal

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

August 2022

DECLARATION

I hereby declare that this project work entitled **Credit Card Fraud Detection classifier using ML and Deep learning** is original and has been carried out by me in the Department of Information and Communication Technology of Manipal Institute of Technology, Manipal, under the guidance of Dr Smitha N Pai, HOD, Department of Information and Communication Technology, M. I. T., Manipal. No part of this work has been submitted for the award of a degree either to this University or to any other Universities.

Place: Manipal

Date :03-08-22

Nishad Chaoji
170953046

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

CERTIFICATE OF COMPLETION

This is Certificate is presented to Nishad Chaoji for Successfully Completing his Final year project under the guidance of Dr Smitha N Pai, HOD , Department of ICT , Manipal Institute of technology, for the duration of four months starting from 16th March to 31st of July 2022.

Dr Smitha N Pai
H.O.D and Guide
ICT Department
Manipal Institute of technology
Manipal

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

CERTIFICATE

This is to certify that the project titled **Credit Card Fraud Detection classifier using ML and Deep learning** is carried out by Nishad Chaoji **from** 16th March 2022 to 31st July 2022 and has satisfactorily completed the project as part of requirement of B.Tech. (Computer and communication Engineering) project work evaluation.

Dr Smitha N Pai
H.O.D and Guide
ICT Department
Manipal Institute of technology
Manipal

ACKNOWLEDGEMENTS

I would like to thank **Dr Smitha N Pai** for her guidance and governance.Due to which I was able to learn diminutive characteristics of project alongside her vast knowledge in the field, provided all the tools which helped us to bring this project to a successful finish.

I would also like to extend my gratitude to the Project Coordinator **Mr. Sanjay Singh** for his unbroken help and surveil during the project work. His efforts and guidance made our workflow coherent.

Furthermore, I am grateful to the management of **Manipal Institute of Technology** for granting this opportunity to learn from these exhilarating experiences and hence giving an exposure to practicality of subjects through hand on projects.

I also want to sincerely thank our Parents and my whole class who have impartedus with the strength to get over all the challenges and overcome all the hurdles in life while also supporting us mentally through the times of pandemic.

ABSTRACT

The rapid growth of the e-commerce industry has led to an exponential increase in credit card usage. With this increase in usage, there have been many cases of fraudulent use of credit cards. Machine learning plays a key role in detecting these fraudulent transactions. For forecasting, these banks use various machine learning and deep learning techniques, historical data is collected, and new features are used to improve their forecasting power.

The performance of credit card transaction fraud detection is highly influenced by the sampling approach to the dataset, the choice of variables, and the detection technique used. The Credit Card Transactions dataset was collected by Kaggle and contains a total of 2,84,808 credit card transactions from European bank records. We consider fraudulent transactions as a "positive class" and genuine transactions as a "negative class". The dataset is highly imbalanced, containing approximately 0.172% fraudulent transactions and the rest genuine transactions. A Python scripting language (version 3.8) for implementing these ML and deep learning techniques.

ACM CSS Taxonomy

[Computing Methodologies]: Machine learning - Learning paradigms -Supervised learning - Supervised learning by classification; Machine learning -Machine learning approaches -Neural networks; Machine learning- Machine learning algorithms -Ensemble methods -Boosting.

Contents

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

LR- Logistic Regression

NB - Naïve Bayes

RF – Random Forest

ML – Machine Learning

DL – Deep Learning

AUC – Area under the ROC Curve

ROC - Receiver operating characteristic

CHAPTER 1

INTRODUCTION

"Fraud" in credit card transactions means illegal and unwanted transactions made by a person other than the owner of the account. We are able to take the necessary precautions to deter this abuse and to investigate such fraudulent behavior so that we can minimize it and protect against similar occurrences in the future. In other words, credit card fraud can be defined as the use of someone else's credit card for personal reasons without the owner and card issuer knowing the card is being used.

Detection includes monitoring the activity of user populations to assess, detect, or prevent aggressive behavior such as fraud, intrusions, and late payments. This is a highly relevant issue and deserves attention from the community, such as machine learning, which can automate the resolution of this issue. Train and train a model to detect fraudulent transactions using machine learning algorithms. Fraud detection methods are constantly being improved to protect systems against various fraud strategies.

- Credit card frauds
- Card theft
- Account bankruptcy
- Device intrusion
- Application fraud
- Counterfeit card
- Telecommunication fraud

Commonly used Machine learning and deep learning techniques are Neural Networks, Fuzzy Logic, logistic regression, Decision trees, Naïve Bayes, etc.

1.1 Motivation

The Motivation behind doing this project is to provide the best ML/DL model that predicts

whether any credit card transaction done is fraud or legal accurately. This project will be an implementation of a mixture of classifiers , Boosting algorithms and a deep learning Model.

The Idea behind this project came from my personal usage of credit cards and the surge in the usage of credit cards off-late due to the various benefits provided by the banks and credit card companies on usage of credit cards

CHAPTER 2

LITERATURE REVIEW

2.1. Referred Literature work

[1] Dejan Varmedja: Proposed Various Machine learning algorithms for credit card fraud detection and analyzed them. The models used were Logistic Regression, Random Forest, and Multilayer perceptron which is an ANN algorithm.

[2] Mohammed Zamini: Proposed an unsupervised algorithm using autoencoders based clustering. The autoencoder is an anauto associater neural network that lowers the dimensionality and extracts the featues.

[3] Sai Kiran: proposed an improved algorithm for credit card fraud detection. That is named as Naïve Bayes improved K-nearest Neighbor method (NBKNN). They have used a dataset on which they had applied the algorithms to identify the fraudulent transaction in the taken dataset.

[4] Kuldeep Randhwa : Proposed multiple ML algorithms for credit card fraud detection ,In addition used the AdaBoost and Majority Voting Methods are applied for making the models Hybrid.

[5] Haibo He, Yang Bai, Edwardo A. Garcia: a novel adaptive synthetic (ADASYN) sampling approach for learning from imbalanced data sets. The essential idea of ADASYN is to use a weighted distribution for different minority class examples according to their level of difficulty in learning

[6] A Candel, V Parmar, E LeDell: Proposed a study on the H2O deep learning Model

2.2 Data analysis tools and libraries

All the graphs and matrices are made using the Matplotlib.pyplot library present in Python.

- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

2.3. Text Preprocessing and Hyper parameter tuning tools.

ADASYN

- It's an Oversampling algorithm that generates synthetic data of the Minority class to balance out the imbalanced data.
- It is a more generic framework, for each of the minority observations it first finds the impurity of the neighborhood, by taking the ratio of majority observations in the neighborhood and k.
- this impurity ratio is converted into a probability distribution by making the sum as 1.
- Then higher the ratio more synthetic points are generated for that point.[5]

.

Model Hyper Parameter tuning

- It is the process in which the parameters are tuned of the model present as a tuple while building the model
- The Aim is to find the parameters where the model performs the best and the error rate is least
- Two types of parameter tuning algorithms are used
    1. Grid Search cv
    2. Randomized Search cv

- In Randomized search cv the model randomly selects the best parameters, and they are put in the random grid and the model selects the best parameters out of it
- In Grid search CV the best parameters have to be generated by creating a params grid and then implement it into model training.
- Result wise both give very similar results

2.4. Machine Learning Models

Machine learning models are created by running the algorithms by feeding in the test data. There are hundreds of machine learning models that are used today for various applications that even include day to day activities. One of them is credit card fraud detection. [1]

Naïve Bayes

Naïve Bayes classifier is an algorithm that is based on the Bayes theorem in probability.

Bayes Theorem:

$P(A/B) = \{P(B/A)*P(A)\}/P(B)$

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.
P(B) is Marginal Probability: Probability of Evidence.

The advantages of using the naïve bayes algorithm are that the computational time is very low, and it can be used in binary as well as multiclass classifications, however it cannot study and

analyse the relation between features because it assumes that all features are independent and unrelated.

Cat Boost

Cat Boost or Categorical Boosting is an open-source boosting library that is used for ranking recommendation systems, weather forecasting , and personal assistants. Cat Boost builds a sequence of approximations in a greedy manner for a given loss function.
Cat boost provides one of the best-in-class accuracy.

Furthermore, Cat boost reduces the need of hyper parameter tuning and also the chances of overfitting are very low.

The attributes in Cat boost algorithm are –

- tree_count_: Returns the number of trees in the model.
- feature_imporatance_: Returns the no of feature importances present
- random_seed: The seed implanted at the start of training.
- learning_rate_: The rate of learning of the model.
- best_score_: returns the best possible result of each metric calculated on each validation dataset.
- best_iteration_: returns the identifier of the best iteration of the evaluation metric.

XGBoost

XG Boost stands for extreme gradient boosting. Gradient Boosting framework are those in which the predecessor data is corrected by the predictor. XG boost is an example of gradient boosting.

In XG boost the decision trees are aligned in a sequential manner and weights are assigned to

the variables which are then fed into the decision trees. The variables that are predicted wrong get an increase in weight and are then fed to the subsequent successor in the sequence.

XGBoost is a scalable and highly accurate implementation of gradient boosting that pushes the limits of computing power for boosted tree algorithms, being built largely for energizing machine learning model performance and computational speed. With XGBoost, trees are built in parallel, instead of sequentially like GBDT. It follows a level-wise strategy, scanning across gradient values and using these partial sums to evaluate the quality of splits at every possible split in the training set.

LightGBM

LightGBM, like XGBoost is also a gradient boosting framework that works on decision trees architecture . Unlike other boosting algorithms LGBM grows leafwise and not tree wise. LGBM uses two techniques i.e., gradient based on side sampling and exclusive feature bundling.

So, the first one only takes the large gradients for information gain and ignore the significant part of the dataset, and the second one i.e. exclusive feature bundling which bundles all these exclusive features into one bundle. This impacts the overall result for an effective feature elimination without compromising the accuracy of the split point. So these two together make a fast and an effective training algorithm.

H2O

H20 is a java-based software used for modelling and computing of data. The main goal of this framework is to give horizontal scaling to any given problem to provide the solution faster. H20's deep learning estimator is a feed forward artificial neural network that is trained with stochastic gradient descent using back propagation. It contains many layers of interconnected neurons starting with an input layer, a middle layer and an ending layer . The starting input contains feature space the middle layer contains nonlinearity and the end has classification or a regression layer for the output.

H2O's deep learning functionalities include supervised training protocol for regression and classification tasks, multi threaded and distributed parallel computing techniques, automatic, per-neuron, adaptive learning rate for fast convergence , optional specification of learning rate, annealing, and momentum options.

H2O Deep Learning is scalable and can leverage large clusters of compute nodes. There are 3 modes of operation. The default behavior is to allow each node to train on the entire (replicated) dataset, but automatically shuffle (and/or use a subset of) the training samples locally on each iteration.

CHAPTER 3

METHODOLOGY

3.1 Understanding the data:

The dataset contains information of European credit card holders' details and transactions taken place in two days. The source of the data used for this project is in the following link: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

The dataset contains:

1. V1 to V28 containing encrypted information of the holder's data.
2. Amount, Time and Class are the other 3 which are non-encrypted

The dataset is highly imbalanced and contains only 492 fraudulent transactions and 284315 non – fraudulent transactions.

3.2. Mapping the real-world problem to the machine learning models:

Performance matrix/ key performance indicator:

How to analyze if the model trained is performing well or not? Hence the need for performance matric. Once the training of the model is completed, the model will be tested using the test and different metrics will be found out which will be used to analyzed how good the model is . These metrics include Accuracy score, Precision score , Recall Score , AUC Score , Confusion Matrix and the ROC curve.

3.3. Exploratory data analysis:

3.3.1. Reading data:

So the data is loaded using the pandas library which is a python library used for data

manipulation and analysis. The data is loaded from a csv file.

3.3.2. Pre-processing performed on the dataset:

The biggest drawback with these kinds of datasets are lack of negative cases or in the case Class = 0.

Class	Count
0	284315
1	492

Table 3.1. Imbalanced dataset

It can be seen in the above table Table 3.1 the dataset is extremely imbalanced. So, to solve this problem the technique used is an oversampling technique called ADASYN . So, using this technique the dataset can be balanced.

Class	Count
0	284315
1	284314

Table 3.2. Balanced dataset

From the above table, Table 3.2 it can be seen that the dataset is now balanced and can be used for model creation. Next thing on the agenda is to check the correlation by drawing a heat map and check the feature importance to check if there is negative correlation with class and whether there are any outliers that need to be removed.

As you can see there is negative correlation between class and V14, V10 and V12 so the outliers will be removed accordingly. It is done by getting the outlier indexes and then removing them and then using that for model creation.

Outlier datapoints

[V10]

Int64Index ([4, 120, 146, 375, 381, 388, 397, 442,
461, 483,
...
568229, 568260, 568263, 568277, 568356, 568398, 568411, 568505,
568537, 568601],
dtype='int64', length=10056)

[V12]

Int64Index ([79, 96, 103, 122, 161, 175, 307, 346,
350, 490,
...
568400, 568420, 568445, 568470, 568486, 568495, 568554, 568588,
568590, 568607],
dtype='int64', length=14449)

[V14]

Int64Index ([182, 889, 1469, 3147, 3220, 3602, 5314, 5407,
5799, 6126,
...
563355, 563743, 563780, 563888, 564177, 565298, 565358, 565919,
567541, 568130],
dtype='int64', length=1288)

After all this preprocessing the data is ready to be split into training testing and validation data (hyper tuning the parameters). For this the library used is the Sk-learn library which is a robust library available for machine learning in Python.

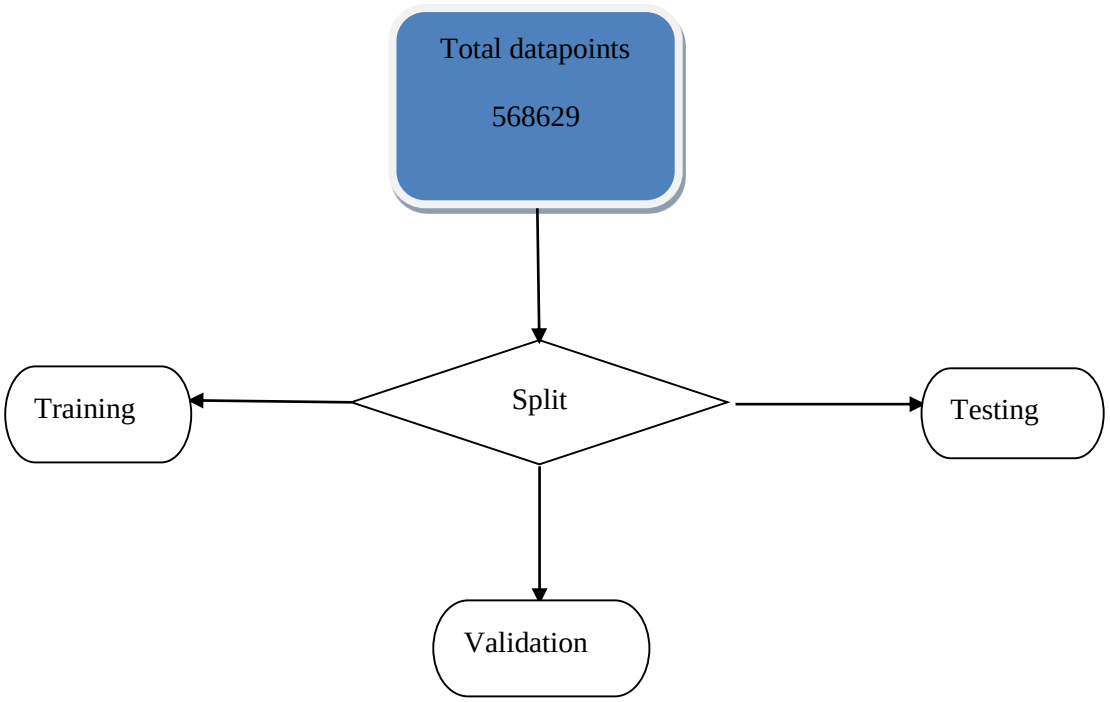

Fig.3.1. Flow chart of the train-test-split.

The split was done in the ratio 80:17:3 i.e. the training data had 341176 datapoints, test data had 159216 data points and the rest was the validation data. Here validation data is only used for getting the best params for hyper parameter tuning. Note that the total does not add up to the original figure of 568629 because of the removal of the outliers.

3.4 Machine learning

After Splitting the datapoints into the 3 datasets the next step is training the Machine learning models using the training data. The model self learns by analysing the training data and once

the training is completed the model will be tested using the testing data and then move to other steps like hyper parameter tuning and so on . The code looks like

Ml_model = Model ();
Ml_model.fit ();

Here **Ml_model** is the object created and **Model** is the model name that will be implemented. The models which are going to be implemented are :

1. Naïve Bayes Classifier
2. Cat Boost
3. LGBM
4. XG Boost
5. H20 estimator

An overview of how the process of machine learning works after the datapoints are split and are ready to train the model using the training dataset can be seen in Fig 3.2 below

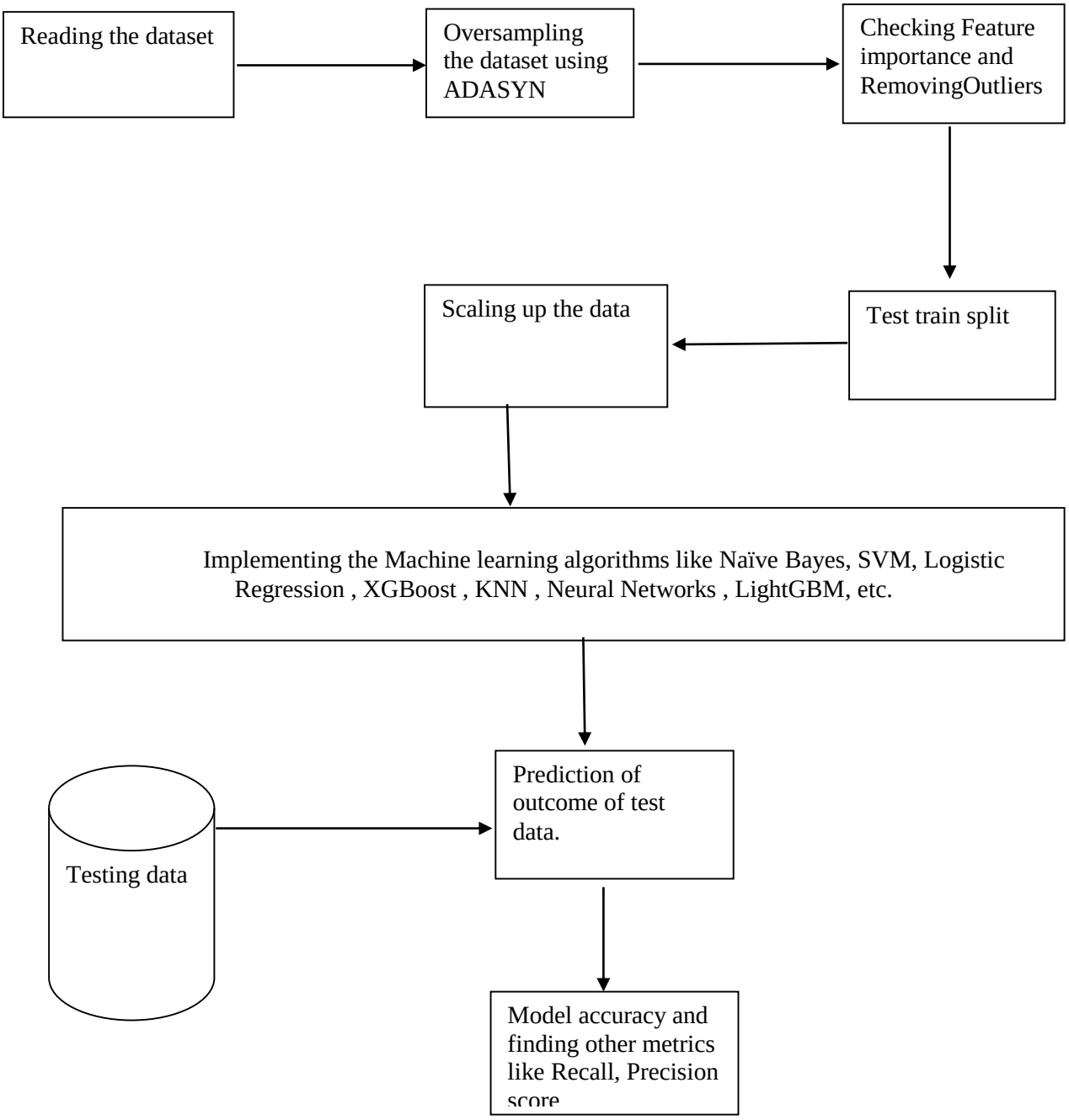

Fig.3.2. System Architecture

In the above figure Fig 3.2 it can be seen how the overall machine learning process works.

3.4.1. Naïve Bayes

Since there are multiple classes, Multinomial Naive Bayes algorithm is used. It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features.

The pseudo code looks like:

multinomialModel = MultinomialNB ()
multinomialModel
multinomialModel.fit ()

Now that the model training is done, the model metrics need to be checked before proceeding to the Hyperparameter tuning.

Accuracy: 0.7795685434556781
Precision: 0.7628594450925447
Recall: 0.7967917797206311
F1-Score: 0.7794564903506455
AUC score: 0.7745515526077781

After this the next step is to tune the model for which the hyperparameter tuning algorithm used is grid search cv. Below is the pseudocode of grid search cv.

param_grid = {}
grid = GridSearchCV (Model())
grid.fit ()

After hyperparameter tuning, the metrics are checked once again and see and improvement in the Accuracy , precision and the F1 score however there is a slight drop in recall score.

Accuracy: 0.790391669178977
Precision: 0.8544704601560943
Recall: 0.7000050246206412
F1-Score: 0.7695632660107027
AUC score: 0.790391669178977

3.4.2 Cat boost

Cat boost is a gradient boosting algorithm that is used for gradient boosting. It is commonly used in self driving cars and weather predictions. It provides great results even with default parameters.

Fig.3.3. Catboost Gradient Boosting

Here in the above image Fig 3.3 it can be seen that each successor tree corrects the error of its predecessor.

The pseudocode of cat boost classifier is :

from cat boost import catboostclassifier

cbc = catboostclassifier ()

cbc.fit(X_train, y_train)

Here the catboost classifier is imported from the from the catboost library which is present in the python 3.8 library ,First the algorithm will be implemented using the default params. The metrics are .

Accuracy: 0.9978708170033163

Precision: 0.9957596907951518

Recall: 1.0

F1-Score: 0.9978753407915766

AUC score: 0.9978708170033163

Now next step is hyper tuning the parameters. for hyper parameter tuning the algorithm used is randomized search cv. The parameters used are

depth = 5,

iterations = 200,

learning_rate = 0.1,

l2_leaf_reg = 0.1,

loss_function= 'Logloss',

eval_metric = 'AUC: hints=skip_train~false

Once the 200 iterations are finished, the metrics are checked once again

Accuracy Score: 0.9996671209732629

Recall Score: 1.0

AUC Score 0.9996665408330188

F1 Score 0.9996678094856062

The metrics have improved but not by much since the default metrics of catboost are very good itself.

3.4.3 LightGBM

LightGBM is another gradient boosting framework based on decision trees. It supports both parallel and GPU learning. LightGBM has over 100 different parameters the Light GBM architecture looks like:

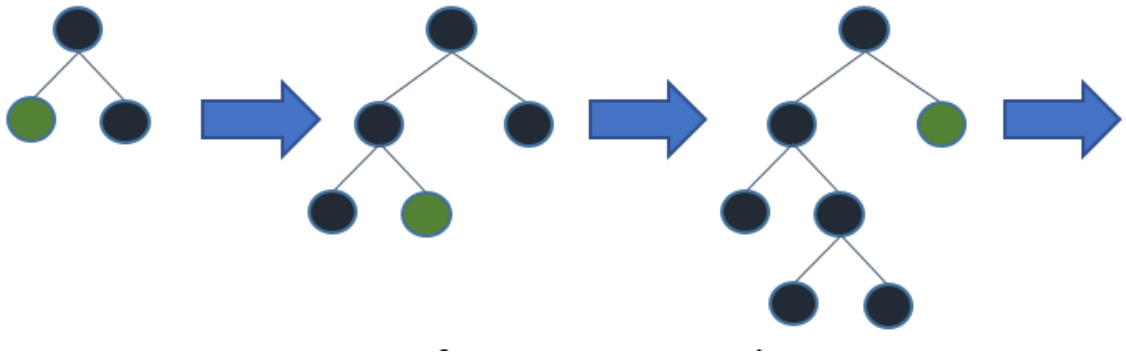

Fig.3.4. leafwise growth of the algorithm.

For this model as well the model will be run first with just the default parameters, the pseudo code looks like :

```
import lightgbm as lgb
lgbm= lgb. LGBMClassifier()
lgbm.fit (X_train, y_train)
```

Once the model is trained the metrics of the untuned model is checked. The metrics look like

Accuracy Score: 0.9994661374099499

Recall Score: 0.9997856052867214

AUC Score 0.9994673985094336

F1 Score 0.9994641855304878

Precision Score 0.9991429723734624

The metrics of this model are very good even without the hyper parameter tuning however there are still false negatives and false positives so there is always room for improvement hence, the hyper parameters are further tuned .

```
params = {
   'max_depth': [6,8,10,14],
   'learning_rate': [0.1,0.15,0.2],
   'num_leaves': [10,20,30]
}
```

The Metrics after hyper parameter tuning are:

----- LGBM -----

Accuracy Score: 0.9994661374099499

Recall Score: 0.9997856052867214

AUC Score 0.9994673985094336

F1 Score 0.9994641855304878

Precision Score 0.9991808340369759

There is significant increase in the metrics.

3.4.4 XG Boost

XG-Boost is a gradient boosting algorithm that uses Trees architecture. It's very good for structured or tabular datasets on classification and regression predictive modeling problem. It builds a sequence of models with each improving over the previous one.

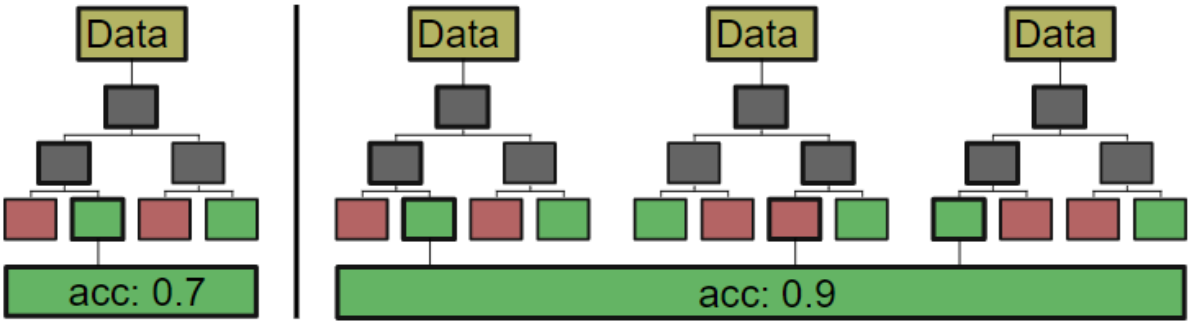

Fig.3.5. XgBoost architecture

Pseudocode:

from xgboost import XGBClassifier

model4 = XGBClassifier(random_state=2)

xgb = model4.fit (x_train, y_train)

First the model will train the model without hyper parameter tuning and test it to get the metrics.

Accuracy Score: 0.9869235069119504

Recall Score: 0.9797860734573087

AUC Score 0.9869359460025942

F1 Score 0.9868520763867837

After first round of model training, next the hyperparameter tuning will be performed to further enhance the model metrics . For this, grid search cv method is used .

Params grid = {'min_child_weight': [1, 5, 10],

'gamma': [0.5, 1, 1.5, 2, 5],

'subsample': [0.6, 0.8, 1.0],

'Colsample_bytree': [0.6, 0.8, 1.0],

'max_depth': [3, 4, 5]}

Using this params, the model will be trained and the new metrics are :

Accuracy Score: 0.986964851521627

Recall Score: 0.9799012857607031

AUC Score 0.9866444624332434

F1 Score 0.9862262814074265

3.5. Deep Learning Model – H2O . AI

H20 is an artificial Neural network The network can contain many hidden layers

consisting of neurons with tanh, rectifier, and maxout activation functions. Advanced features such as adaptive learning rate, rate annealing, momentum training, dropout, L1 or L2 regularization, checkpointing, and grid search enable high predictive accuracy.

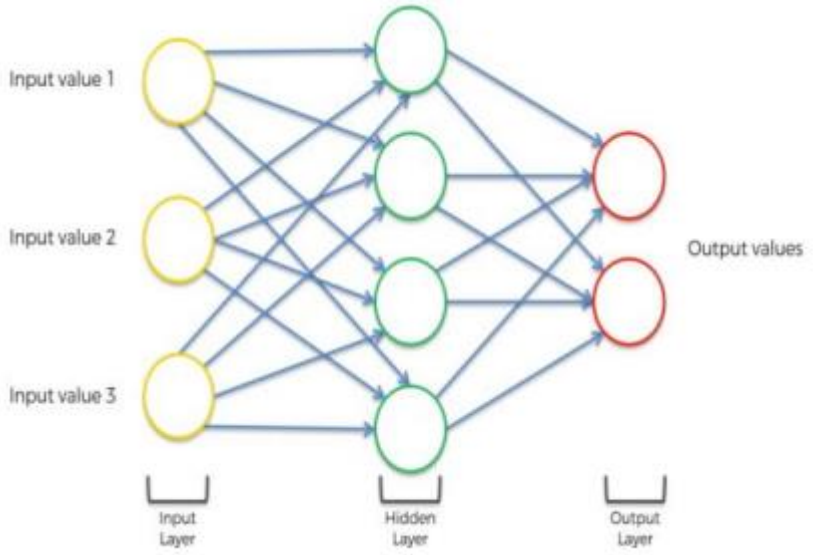

Fig.3.6.H2O Estimator

The first step is to connect the to the local host. The default setting is **http://localhost:54321**

Next step is to pick the factor column, in this case the 'Class' column and then begin building the deep learning model using the H20 Estimator . The params used were

epochs=2000,

distribution = 'bernoulli',

missing_values_handling = "MeanImputation",

variable_importances=True,

nfolds = 2,

fold_assignment = "Stratified",

keep_cross_validation_predictions = True,

balance_classes=False,

standardize = True,

activation = 'RectifierWithDropout',

hidden = [100,100],

stopping_metric = 'logloss',

loss = 'CrossEntropy'

The time taken for the model to train was 140814.80 seconds or roughly 39hrs and 12 minutes. Once the training part is done next the model will be cross validated using the cross validation data and the metrics for the training and cross validation will be calculated.

Model Details

==========

H2ODeepLearningEstimator: Deep Learning

Model Key: DeepLearning_model_python_1657893532580_449

Next step is testing the model, for this the first step is to convert the numpy array into H2o data frame

_test = h2o.H2OFrame(X_test)

X_test['C31'] = X_test['C31']. asfactor()

After converting the it into H2O dataframe the next step is testing the model using the test data

deeplearning prediction progress:

(done) 100%

(119407, 3)

Once the results are out the next step is to convert the H2o dataframe back to pandas dataframe and then check all the metrics of the created Model.

CHAPTER 4

Results and Discussion

4.1 Data Analysis

The dataset is extremely imbalanced i.e., the no of negative cases are much lower than the no of positive cases.

No of negative cases: 492

No of positive cases: 284315

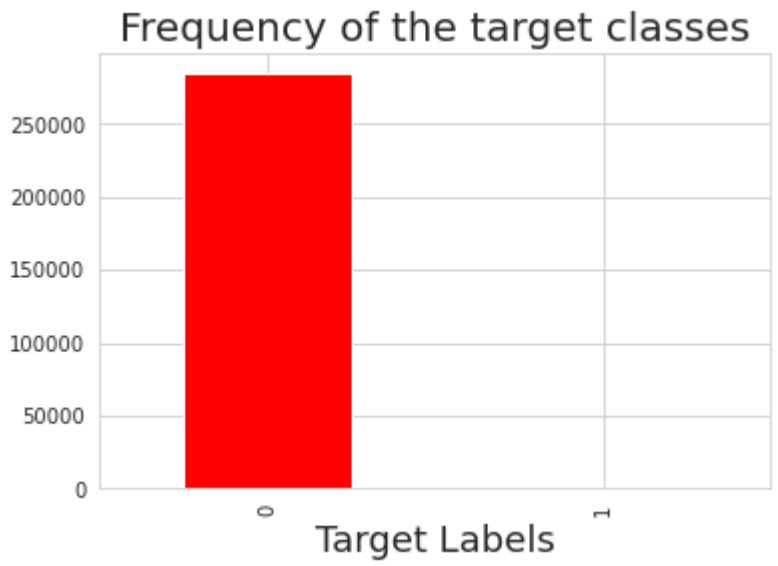

Fig.4.1. Frequency of target classes

The above plot (Fig.4.1) shows the frequency of the targets. After applying ADASYN the dataset has balanced the below figure (Fig.4.2) shows the count plot of the classes after updating the dataset using ADASYN .

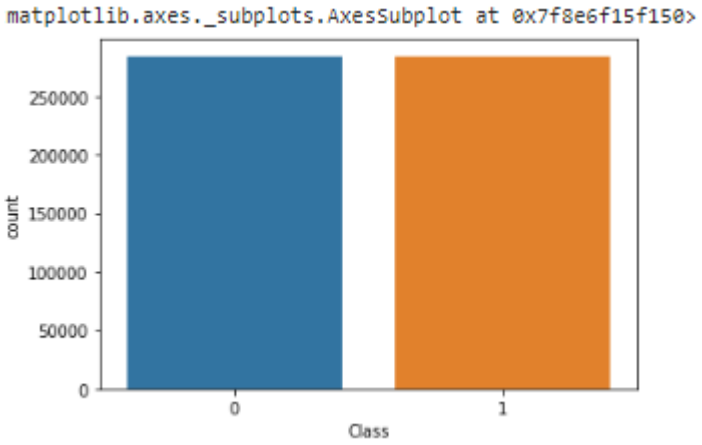

Fig.4.2. Count plot of the classes post ADASYN

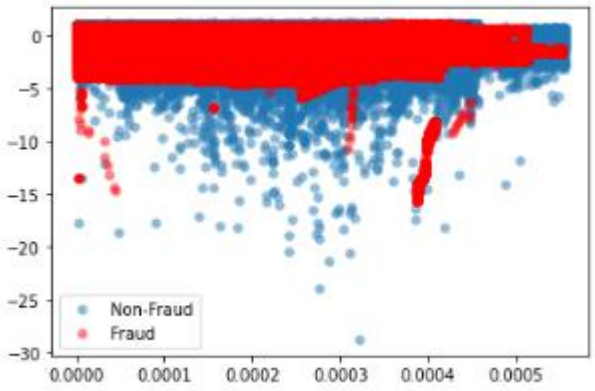

Fig.4.3. Scatter plot of the target labels

From the above Fig.4.3. it can be seen that now the dataset is balanced and the number of positive and the number of negative cases is same.

Next to remove the outliers feature importance and correlation matrix has to be plot to find out which columns have negative correlation with class , and then remove those outliers

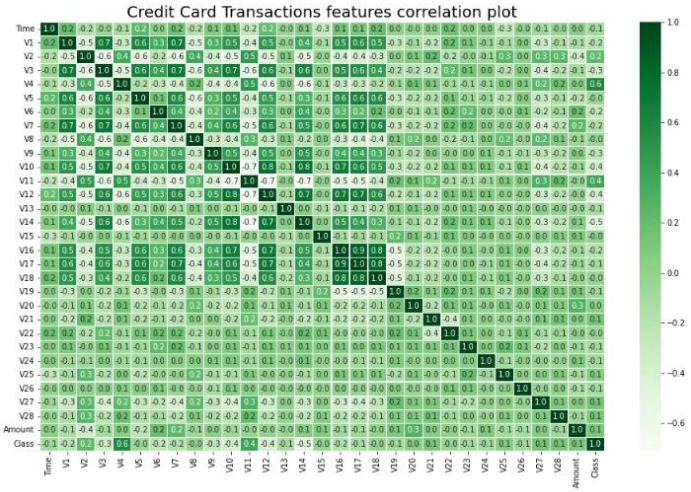

Fig.4.4. Feature Correlation plot of the dataset

From the above figure Fig.4.3 it is noticed that the columns that have negative correlation with class are **V10 , V12 , V14** . Outliers need to be removed from these three or else Overfitting will happen. This can be confirmed in the below figure Fig.4.5. which shows the importance of each of the features?

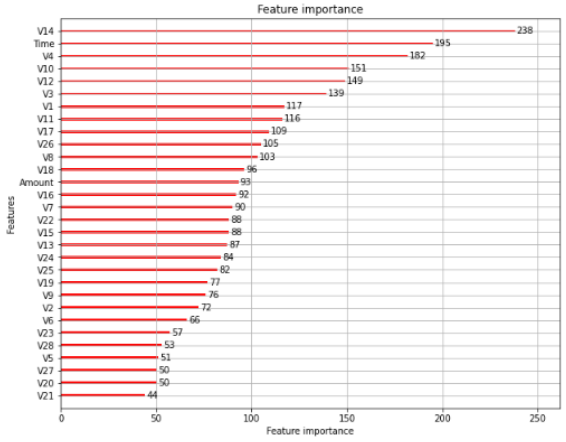

Fig.4.5. Feature Importance of each of the features

Next the outlier data is checked and removed . A total of 25,793 data points got removed, hence the no data points in the final train-test split is

Data	Count
Train	341176
Test	159216
Validation	42444

Table 4.1. Train test split

4.2 Machine Learning Models Results

4.2.1. Naïve Bayes Results and Graphs

First the Model is tested without Hyperparameter tuning. The metrics are

Class	Precision	Recall	F1	AUC	Accuracy
Non-Fraudulent	0.79	0.75	0.78	0.78	0.77
Fraudulent	0.76	0.80	0.77	0.78	0.77
Overall	0.762	0.796	0.78	0.78	0.77

Table 4.2. Metrics table for Naïve bayes

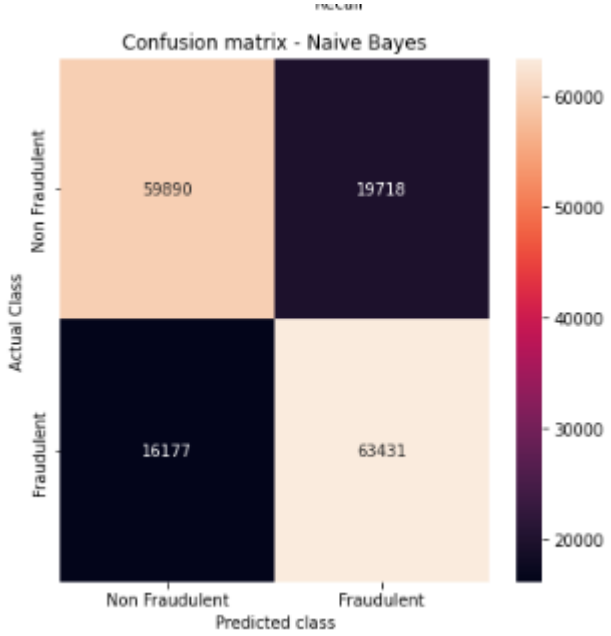

Fig.4.6. Confusion Matrix of Naïve bayes

The above figure Fig.4.6 shows the confusion matrix of the algorithm with a big number of false negatives and false positives. From the above table it can be concluded that the metrics of the trained model can be made better by hyper parameter tuning using the grid search cv.

Alpha	Verbose	Fit_prior
3	3	'false'

Table.4.3. Best Params Table

After hyperparameter tuning the model will be trained again and the new metrics are

Class	Precision	Recall	F1	AUC	Accuracy
Non Fraudulent	0.75	0.88	0.81	0.79	0.79
Fraudulent	0.85	0.70	0.77	0.81	0.80
overall	0.85	0.70	0.77	0.79	0.79

Table.4.4. Hyper parameter tuned metrics

From the above table some noticeable changes have occurred in all the metrics with an increase in four out of 5 metrics with a reduction in Recall score. To confirm whether the model is working better or not, the confusion matrix is checked

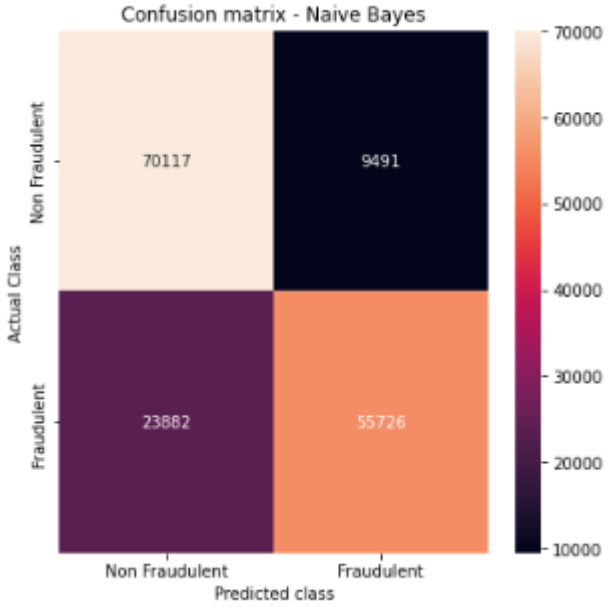

Fig.4.7. Confusion Matrix Hyperparameter tuned model

In the above confusion matrix Fig.4.7, it is seen that the sum of false positives and false negatives has reduced by over 3000 . Below is the precision recall curve and the ROC curve for the hyper tuned Naïve Bayes model.

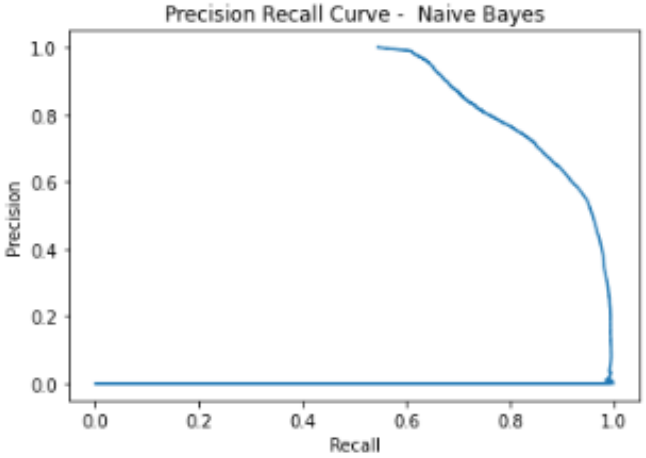

Fig.4.8. Precision recall curve

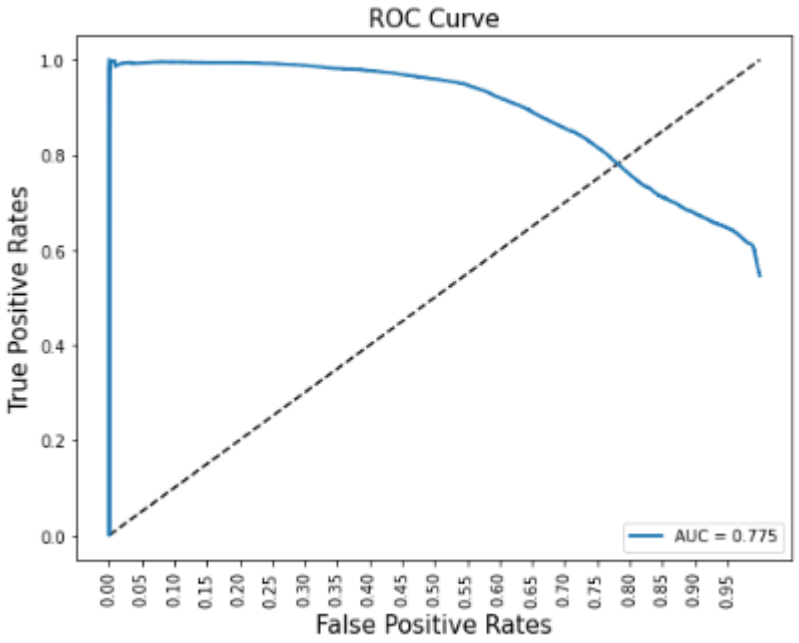

Fig.4.9.ROC Curve Naïve Bayes tuned.

4.2.2. CatBoost results and graphs

Similarly for this the first step is to train the model with the default parameters. then after the default parameters are analyzed the next step is hyper tuning. The below table shows the metrics after using default parameters

Class	Precision	Recall	F1	AUC	Accuracy
Non Fraudulent	0.995	1	0.979	0.997	0.997
Fraudulent	1	1	1	1	1
overall	0.995	1	0.979	0.997	0.997

Table.4.5. Metrics table catboost

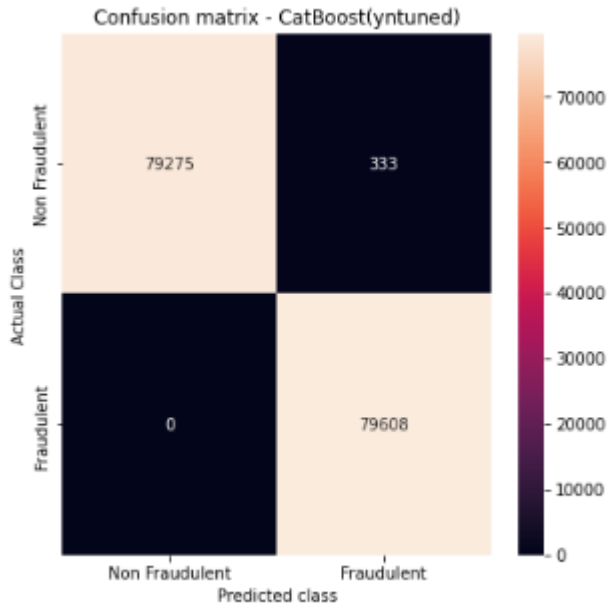

Fig.4.10. Confusion Matrix Catboost

In the confusion matrix in the above figure Fig.4.9 there are 333 false negative cases which can

From the above table the it can be analyzed that even with default parameters catboost gave near perfect metrics, but there is always room for improvement so , hyper tuning of the model is done as there is still room to reduce the false negatives that have occurred while testing . Best params used are shown in the below table

Parameters	Values
Iterations	‘12’
Eval_metric	‘AUC’
Random_seed	2022
Od_type	‘Iter’
Metric_period	1
Od_wait	100

Table.4.6. Params used for hypertuning

The Metrics after hyper parameter tuning are

Class	Precision	Recall	F1	AUC	Accuracy
Non Fraudulent	0.9966	1	0.999	0.999	0.999
Fraudulent	1	1	1	1	1
overall	1	1	0.999	0.999	0.999

Table.4.7. Metrics table catboost after tuning.

From the above Table.4.7, there is a small increase in all the 5 metrics and this can be confirmed by plotting the confusion matrix.

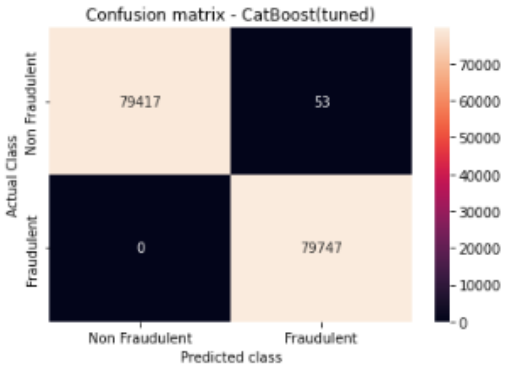

Fig.4.11. Confusion matrix catboost tuned .

In the above confusion Matrix in Fig.4.9 its observed that the no of false negatives has reduced to 53 from 333. Below is the precision recall curve and the AUC Curve of the tuned catboost model

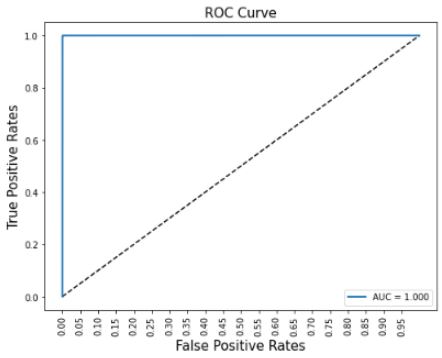

Fig.4.12.ROC Curve catboost tuned

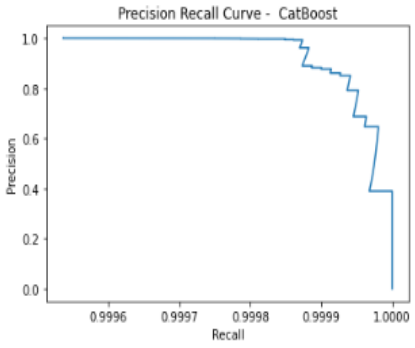

Fig.4.13. P-R curve catboost tuned

The 2 above images show the trends of the AUC Score (Fig.4.12) and the Comparison between precision and recall (Fig.4.13)

4.2.3. LGBM results and graphs

Model metrics before hyperparameter tuning are,

Class	Precision	Recall	F1	AUC	Accuracy
Non Fraudulent	0.99	0.99	0.99	0.99	0.99
Fraudulent	1	1	1	1	1
overall	0.9991	0.9997	0.9994	0.9995	0.9994

Table.4.8. Metrics table LGBM untuned

The Metrics for the model even with the default parameters are very good as seen in Table.4.8 which can be confirmed with the confusion matrix below.

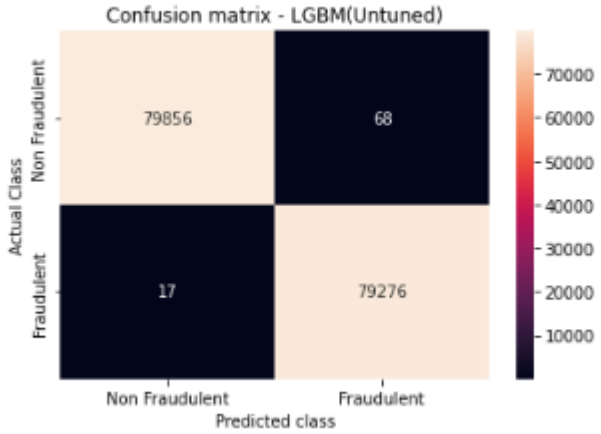

Fig.4.14. Confusion Matrix of lgbm

From the above Confusion Matrix, it can be seen that the number of false negatives and false positives are very low i.e. only 85 out of 159216 data points. However there is room to further enhance the metrics and reduce the false values present and further increase the metrics. The best params using grid search cv are

Max_death	Learning_rate	Num_leaves
10	0.1	20

Table.4.9. Best params

Metrics after retraining the model with the best parameters

Class	Precision	Recall	F1	AUC	Accuracy
Non Fraudulent	0.99	0.99	0.99	0.99	0.99
Fraudulent	1	1	1	1	1
overall	0.9991	0.9997	0.9994	0.9995	0.9994

Table.4.10. Model metrics LGBM tuned

There is a very small change in the metrics, which is not visible here, however it can be seen in the confusion matrix below

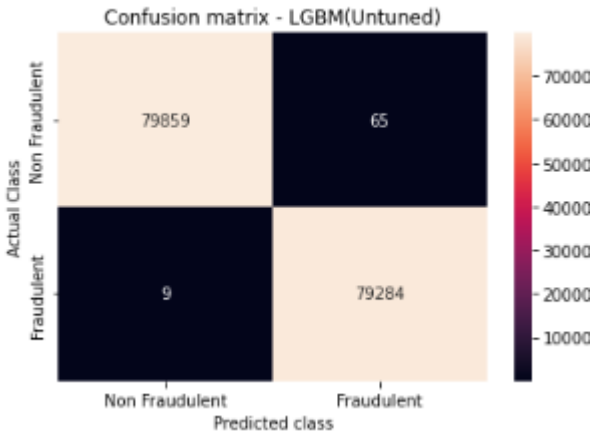

Fig.4.15. Confusion Matrix

From the confusion matrix it is seen that the number of false negatives and false positives are reduced from 85 to 74 which is not a significant change however it is still an improvement over the previous results. Below are the precision-recall curve and the ROC curve of the given Model

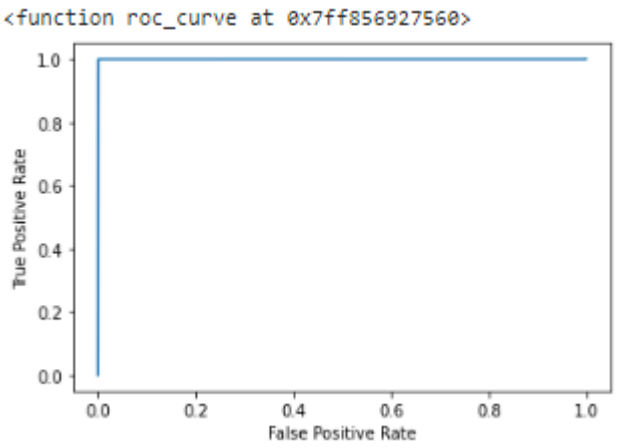

Fig.4.16.ROC Curve

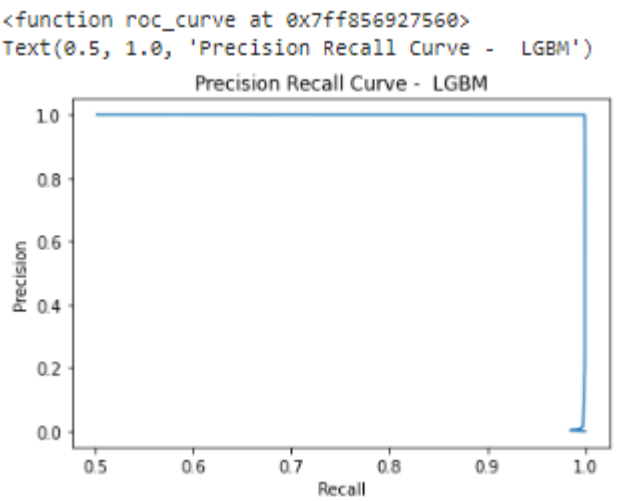

Fig.4.17. Precision-recall curve LGBM

4.3.4 XGBoost Results and Graphs

Model metrics after training the model with default parameters are

Class	Precision	Recall	F1	AUC	Accuracy
Non Fraudulent	0.98	0.989	0.99	0.98	0.99
Fraudulent	0.99	0.97	0.99	0.99	0.99
overall	0.985	0.979	0.9868	0.986	0.986

Table.4.11. Metrics XGBoost untuned

From the above table it is analyzed that the model metrics are very good however there is room for improvement which can be confirmed in the confusion matrix below.

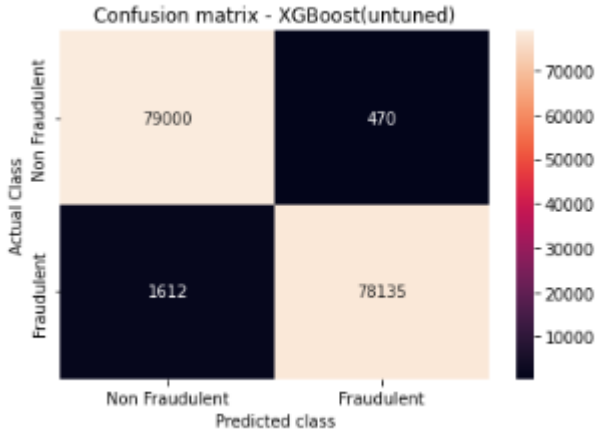

Fig.4.18. Confusion matrix XgBoost untuned

From the confusion matrix above Fig.4.18. it can be noted that there are a few false positive and false negative, which can be reduced by hyper parameter tuning . Best params are

Min_child_ weight	Max_ depth	gamma	Colsample_ bytree	Sub sample
5	5	5	0.6	1

Table.4.12. Best Params for xgboost

Once the model is retrained this time with best params, the new metrics are

Class	Precision	Recall	F1	AUC	Accuracy
Non Fraudulent	0.98	0.989	0.99	0.98	0.99
Fraudulent	0.99	0.97	0.99	0.99	0.99
overall	0.985	0.98	0.986	0.98	0.987

Table.4.13. Metrics XGboost

From the above metrics table, a slight improvement can be seen in the metrics of the model. This can be seen better in the confusion matrix by comparing the number of false

positives and false negatives of the tuned and untuned model.

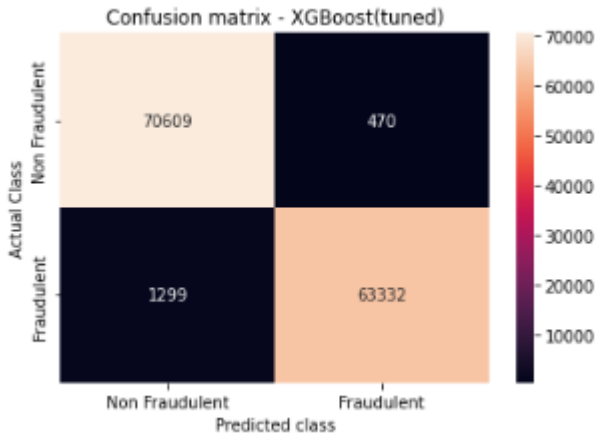

Fig.4.19. Confusion Matrix XGBoost tuned model

From the above figure Fig.4.19, the no of false positives and false negatives has reduced by 333 data points after hyper parameter tuning. Below are the Precision-Recall curve and the ROC Curve

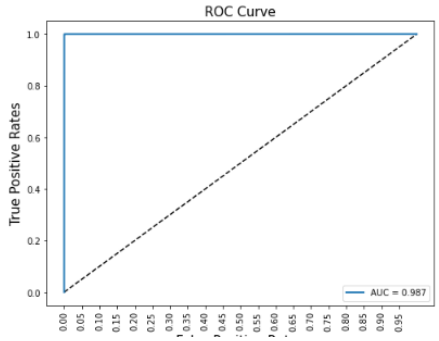

Fig.4.20.ROC Curve XGboost tuned

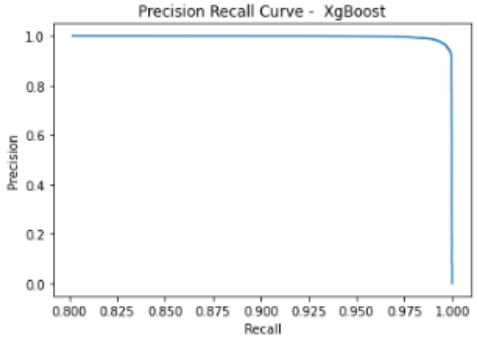

Fig.4.21. Precision-Recall Curve XG Boost tuned

The above figures Fig.4.20 and Fig.4.21 depict the ROC curve and precision recall curve respectively, and it can be said that the graphs are near perfect .

4.2.5 H2O deep learning Estimator results and graphs

The model is run for 1000 epochs and Bernoulli distribution was used. The training time of the model was 140814.80 seconds. The metrics on the training data are

Metrics	Values
Gini:	0.9995046118844686
MSE:	0.00036861534104013426
RMSE:	0.019199357828847668
LogLoss:	0.005753125992453461
Mean Per-Class Error:	0.00034167588267096285
AUC:	0.9997523059422343

Table 4.14 Metrics table for the training data for the H2o

The above table 4.14 shows all for the training datapoints, and as seen the metrics even without any hyper tuning are near perfect which can be confirmed with the confusion Matrix

	Non-Fraudulent (0)	Fraudulent (1)	Rate of false value
Non-Fraudulent (0)	170809	89	89/170898
Fraudulent (1)	7	170271	7/170278
Overall	170816	170360	96/341176

Table.4.15. Confusion matrix of the training dataset

From the above confusion matrix Fig 4.16 , it can be seen that there are only 96 total false values which is very low hence it can be said the training part went well. Next the model is getting cross validated before putting in the test data for prediction and finding the metrics. below table gives the cross-validation metrics

Metrics	Values
Gini:	0.9995046118844686
MSE:	00029071416905164087
RMSE:	0.017199357829747848
LogLoss:	0.0036774895622737895
Mean Per-Class Error:	0.00034167588267096285
AUC:	0.9998481069531275

Table.4.16. Metrics table cross validation

	Non-Fraudulent (0)	Fraudulent (1)	Rate of false value
Non-Fraudulent (0)	21195	12	12/21207
Fraudulent (1)	5	21232	5/21237
Overall	21200	21244	17/42444

Table.4.17. Confusion Matrix of the cross validation

The Model is cross validated and can be tested for prediction. And check the model metrics. the converted H2O dataframe looks like

Predict	P0	P1
1	1.549855e-081	1.000000e+00
1	6.680133e-07	9.999993e-01
1	4.824268e-09	1.000000e+00
0	1.000000e+00	2.911745e-183
1	8.283057e-08	9.999999e-01

Table.4.18.the converted H2O dataframe

The shape of the test dataframe is (159216, 2) . After testing the metrics and converting the H2O data frame obtained are

Precision	Recall	AUC
0.9998827529604878	0.9999162493090568	0.9999084659064857

Table.4.19. Metrics table for H2O estimator

From the above table it can be seen that the model metrics are very good and the model is working very well, which can be confirmed from the confusion matrix below .

	Non-Fraudulent	Fraudulent	Rate of false value prediction
Non-Fraudulent	79187	8	8/79198
Fraudulent	11	80010	11/80018

Overall	79198	80018	19/159216

Table.4.20. Confusion matrix

From the above confusion matrix Table. 4.20 it can be confirmed that the model is performing well giving only a total of 19 false values out of 159216 total datapoints.

4.3. Metrics Table

The below table shows the overall comparison between all the models.

MODEL	Precision	Recall	Accuracy	AUC	Training Time
NB	0.596357	0.992550	0.660373	0.660373	4.6s
NB + Parameter tuning	0.861625	0.700595	0.794041	0.794040	15.2s
Cat Boost	0.995759	1.000000	0.997870	0.997870	179.5s
Catboost + Parameter Tuning					1135.6s
LightGBM	0.9991	0.9997	0.9995	0.9994	78.4s
LightGBM + parameter tuning	0.9991	0.9997	0.9995	0.9994	800s
XG Boost	~0.99	0.97986	0.987692	0.986935	244s
XGBoost + parameter tuning	~0.99	0.9799	0.987696	0.986944	1375.3s
H20 (Training)	-	-	-	0.999848	140814.8s
H20 (Testing)	0.9999162	0.999428	0.99989	0.9999487	-

Table.4.21. Overall Metrics Table

CHAPTER 5
Conclusion and Future Work

From the above Metric table, the following conclusions can be made

- The 3 Boosting algorithms work better here than the Classification algorithm when purely comparing just machine learning.
- Naïve Bayes is the Fastest algorithm in terms of training and computation time but time but that does not mean faster is always better.
- Deep Learning model performs better than the Machine Learning model however the training and computation time is too much for it because of a complex architecture and not powerful enough GPU.

This research is being carried out to offer a multiclass classifier to detect the fraud credit card transactions from the genuine ones. It also aids in the development of customized Fraud detection Software's. For this project only have one classifying algorithm i.e., Naïve Bayes is used and three boosting algorithms i.e., LightGBM, XGBoost and Cat Boost. Along with these a deep learning model is also implemented, reason being to test out how the deep learning model works in comparison to these. The metrics on which the model performance will be evaluated were confusion matrices, AUC score, Precision, Recall,and Accuracy score of the above models

Practically the best model for our requirement after observing the above outputs and graphs would be Cat Boost in Machine learning Models, and the deep learning model gives better results than machine learning models however the computational time is too high in the case of deep learning

In the future for this application the model can be enhanced by using better machines or can be implemented on cloud and in the process multiple rounds of hyper tuning can be run ,more deep learning algorithms like MLP and LSTMs and also hyper tune the deep learning model

as well.

Also, there is a lot of encrypted data in the dataset so if the data can be decrypted then multiple classes can be created and ML and DL algorithms can be run on them to get an even more effective Model.

REFERENCES

[1] Pelin Yıldırım Taşer, Fatma Bozyiğit. "Chapter 7 Machine Learning Applications for Fraud Detection in Finance Sector", Springer Science and Business Media LLC, 2022

[2] Carcillo, F., Dal Pozzolo, A., Le Borgne, Y.-A., Caelen, O., Mazzer, Y., & Bontempi, G. (2018). SCARFF: A scalable framework for streaming credit card fraud detection with spark. Information Fusion, 41, 182–194.
doi: 10.1016/j.inffus.2017.09.005.

[3] Carcillo, F., Le Borgne, Y.-A., Caelen, O., Kessaci, Y., Oblé, F., & Bontempi, G. (2019). Combining Unsupervised and Supervised Learning in Credit Card Fraud Detection. Information Sciences.
doi: 10.1016/j.ins.2019.05.042.

[4] Cheng, D., Xiang, S., Shang, C., Zhang, Y., Yang, F., & Zhang, L. (2020). Spatio-Temporal Attention-Based Neural Network for Credit Card Fraud Detection. Proceedings of the AAAI Conference on Artificial Intelligence, 34(01), 362–369. doi:10.1609/aaai. v34i01.5371.

[5] De Sá, A. G., Pereira, A. C., & Pappa, G. L. (2018). A customized classification algorithm for credit card fraud detection. Engineering Applications of Artificial Intelligence, 72, 21-29.

[6] Shahnawaz, M. R. (2011). ANN and rule based model for English to Urdu-Hindi machine translation system. In Proceedings of National Conference on Artificial Intelligence and agents: Theory& Application, AIAIATA (pp. 115-121).

[7] Szmigiera, M. (2021). Value of fraudulent card transactions worldwide 2021-2027, Statista.com, accessed on 15-Jan-2022. https://www.statista.com/statistics/1264329/.

[8] Xuan, S., Liu, G., Li, Z., Zheng, L., Wang, S., & Jiang, C. (2018). Random forest for credit card fraud detection. 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC).

doi:10.1109/icnsc.2018.8361343.
[9] Zhang, X., Han, Y., Xu, W., & Wang, Q. (2019). HOBA: A Novel Feature Engineering Methodology for Credit Card Fraud Detection with a Deep Learning Architecture. Information Sciences.
doi: 10.1016/j.ins.2019.05.023
[10] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C. Prati, Bartosz Krawczyk, Francisco Herrera. "Learning from Imbalanced Data Sets", Springer Science and Business Media LLC, 2018

PROJECT DETAILS

Student Details

Student Name	Nishad Chaoji		
Registration Number	170953046	Section/Roll No.	B/72
Email Address	nishadchaoji@gmail.com	Phone No.(M)	9071194124

Project Details

Project Title	**Credit Card Fraud Detection classifier using ML and Deep learning**		
Project Duration	4-6 Months	Date of Reporting	16-03-2022

Internal Guide Details

Faculty Name	Dr Smitha N Pai
Full Contact Address with PIN Code	Department of Information and Communication Technology, Manipal Institute of Technology, Manipal-576104
Email Address	smitha.pai@manipal.edu

PLAGARISM REPORT

Credit Card Fraud Detection classifier using ML and Deep learning A

ORIGINALITY REPORT

11%	10%	6%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	2%
2	www.ijitee.org Internet Source	1%
3	cse.anits.edu.in Internet Source	1%
4	docplayer.net Internet Source	1%
5	deepnote.com Internet Source	1%
6	Haibo He, Yang Bai, Edwardo A. Garcia, Shutao Li. "ADASYN: Adaptive synthetic sampling approach for imbalanced learning", 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 2008 Publication	<1%
7	catboost.ai Internet Source	<1%

8	docs.h2o.ai Internet Source	<1 %
9	iwaponline.com Internet Source	<1 %
10	link.springer.com Internet Source	<1 %
11	github.com Internet Source	<1 %
12	www.ijera.com Internet Source	<1 %
13	keep.lib.asu.edu Internet Source	<1 %
14	gnu.inflibnet.ac.in Internet Source	<1 %
15	Pelin Yıldırım Taşer, Fatma Bozyiğit. "Chapter 7 Machine Learning Applications for Fraud Detection in Finance Sector", Springer Science and Business Media LLC, 2022 Publication	<1 %
16	ijarcce.com Internet Source	<1 %
17	impressions.manipal.edu Internet Source	<1 %
18	mafiadoc.com Internet Source	<1 %

19	repositorio.ufmg.br Internet Source	<1 %
20	Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C. Prati, Bartosz Krawczyk, Francisco Herrera. "Learning from Imbalanced Data Sets", Springer Science and Business Media LLC, 2018 Publication	<1 %
21	Eunji Kim, Jehyuk Lee, Hunsik Shin, Hoseong Yang, Sungzoon Cho, Seung-kwan Nam, Youngmi Song, Jeong-a Yoon, Jong-il Kim. "Champion-challenger analysis for credit card fraud detection: Hybrid ensemble and deep learning", Expert Systems with Applications, 2019 Publication	<1 %
22	Georgios Zioviris, Kostas Kolomvatsos, George Stamoulis. "Credit card fraud detection using a deep learning multistage model", The Journal of Supercomputing, 2022 Publication	<1 %
23	Pramod Gupta, Naresh K. Sehgal. "Introduction to Machine Learning in the Cloud with Python", Springer Science and Business Media LLC, 2021 Publication	<1 %

24	Shruti Deshpande, J. Gujarathi, P. Chandre, Pravin Nerkar. "Chapter 11 A Comparative Analysis of Machine Deep Learning Algorithms for Intrusion Detection in WSN", Springer Science and Business Media LLC, 2021 Publication	<1 %
25	T. Yamauchi, S. Kawabata, T. Mochizuki. "Pressure response inside double glazed windows", Journal of Wind Engineering and Industrial Aerodynamics, 1983 Publication	<1 %
26	Zhiyong Li, Junfeng Zhang, Xiao Yao, Gang Kou. "How to identify early defaults in online lending: A cost-sensitive multi-layer learning framework", Knowledge-Based Systems, 2021 Publication	<1 %
27	arrow.tudublin.ie Internet Source	<1 %
28	eprints.untirta.ac.id Internet Source	<1 %
29	pure.tue.nl Internet Source	<1 %
30	www.irjmets.com Internet Source	<1 %

31	Shahnawaz Khan, Abdullah Alourani, Bharavi Mishra, Ashraf Ali, Mustafa Kamal. "Developing a Credit Card Fraud Detection Model using Machine Learning Approaches", International Journal of Advanced Computer Science and Applications, 2022 Publication	<1 %
32	Anahita Farhang Ghahfarokhi, Taha Mansouri, Mohammad Reza Sadeghi Moghaddam, Nila Bahrambeik et al. "Credit card fraud detection using asexual reproduction optimization", Kybernetes, 2021 Publication	<1 %

Exclude quotes	On	Exclude matches	< 3 words
Exclude bibliography	On		