An empirical study on credit card fraud detection with modern performance comparisons

Ritu Ramakrishnan
School of Computer Science and Engineering
University of the Pacific
Stockton,United States of America
r_ramakrishnan@u.pacific.edu

***Abstract*—The paper covers the significant increase in credit card fraud that has occurred as a result of the rise in online payments. It emphasizes the importance of a fraud detection system for credit card transactions capable of detecting fraudulent transactions. The paper analyzes fraud using machine learning approaches and dealing with credit card transaction dataset imbalances. On a credit card transaction dataset, it compares the performance of various models. In this paper comparison of the models is based on the accuracy, precision, recall, and F1-score of Ada Boost, Cat Boost, XGBoost, and Light GBM models in detecting fraud. The algorithm with the highest F1 score is considered the most effective at detecting fraud. In summary, the efficiency of detecting fraud in the credit card transaction is compared on the machine learning algorithms. The study's findings may help develop a better model for detecting fraud in the credit card transactions.**

Keywords—Credit Card, Machine Learning models, Ada Boost, Cat Boost, Boost and Light GBM model.

I. INTRODUCTION

The digital payments has increased in the recent days which has led to an increase in fraudulent activities. This is one of the methods in credit card fraud transactions. According to recent reports, credit card fraud is expected to cause global losses of around $43 billion within the next five years, and as many as 80% of US credit cards have been compromised. This poses a significant challenge to e-commerce merchants, who are at high risk of facing the costs of these type of transactions. However, implementing such a system is highly challenging due to the issues associated with the class imbalance and the changing statistical properties of transaction behaviour over time.

This issue can be addressed by developing a model for detecting the fraud in the credit card transactions. This system involves observing the different transactional behaviour of card holders to verify and avoid suspicious behaviour, which may be fraudulent transactions.

The machine learning approaches can be promising and automated. The fraud detection system can be embedded with supervised machine learning algorithms that learn the pattern of transaction behaviour to detect fraudulent transactions. These models are trained with past historical transaction data of the credit card that contains true and fake transactions. If suspect behavior is discovered, it is investigated further by specialists, and frequent input is sent to the system to continuously train the algorithm. The growing concerns about credit card fraud and associated financial losses is the motivation for developing such a system. Data sources for developing this system include historical credit card transaction data, which is used to train the machine learning algorithms. The primary challenges in developing such a system include dealing with the class imbalance and the changing statistical properties of transaction behaviour over time. By addressing these challenges, a Credit Card Fraud Detection System can safeguard against fraudulent transactions, helping merchants minimize financial damage.

Let us discuss in more detail the fraudulent transactions in the dataset, recent research in this field and efficiency of the models in the upcoming headings.

II. RELATED WORK

The paper examines several studies on Credit Card Fraud Detection Using Machine Learning Approaches. Several researchers have proposed various techniques for dealing with imbalanced datasets and lowering the false positive rate, which is a critical challenge in detecting Credit Card fraud. Random Forest, Decision Trees, Nave Bayes, Support Vector Machines (SVM), and Artificial Neural Networks (ANN) are popular methods used in literature. The authors assessed each algorithm's performance using accuracy, precision, recall, and F1-score. The random forest algorithm outperformed the other three algorithms in the study, with an F1-score of 0.87. The study gives valuable insights into the effectiveness of several Machine Learning algorithms for identifying Credit Card fraud, which can assist academics and practitioners in developing more efficient and accurate fraud detection systems [1].

Several machine techniques have been used to identify fraudulent credit cards transactions, such as decision trees, support vector machines, neural networks, and ensemble approaches. In addition, the study examines the difficulties in identifying credit card fraud, such as unbalanced datasets, idea drift, and feature selection. It also investigates feature engineering strategies for improving machine learning model performance in identifying fraud. The study's writers reviewed several research findings and emphasized the advantages and disadvantages of various techniques. They also covered the real-world uses of machine learning algorithms in detecting credit card fraud, including case studies and commercial implementations [2].

The use of a Convolutional Neural Network (CNN) model to detect credit card fraud. The authors highlight the significance of credit card fraud and its increasing prevalence in the digital age. They mention the use of traditional methods like rule-based approaches and anomaly detection, but also point out the limitations of these methods in detecting complex and evolving fraud patterns. The authors then introduce their proposed CNN model, which utilizes both the spatial and temporal features of credit card transactions to detect fraud. They explain the data pre-processing steps, including oversampling of the minority class and normalization of the data. They also describe the architecture of the CNN model and its various layers, including convolutional, pooling, and fully connected layers. A CNN model is used to identify credit card fraud. The writers stress the importance of credit card fraud and its growing prevalence in the digital era. They cite established methods such as rule-based approaches and anomaly detection but highlight their limits in identifying complex and developing fraud patterns. The authors then present their suggested CNN model, which detects fraud by utilizing both the geographical and temporal aspects of credit card transactions. They describe the data pre-processing processes, such as oversampling the minority class and data normalization. They also detail the CNN model's architecture and its many levels, which include convolutional, pooling, and fully connected layers. The authors compare the performance of their CNN model on a publicly available dataset to that of other cutting-edge machine learning algorithms. They claim that their CNN model surpasses other models in terms of accuracy, precision, recall, and F1 score. They also test how different parameters affect the model's performance.[3]

Detecting fraudulent credit card transactions with the use of several machine learning techniques. This paper's linked work contains many earlier works that studied similar methodologies. Among these research are supervised machine learning techniques for credit card fraud detection, such as logistic regression, decision trees, support vector machines, and neural networks. Other research has looked into employing unsupervised machine learning methods like clustering and anomaly detection to detect fraud. Furthermore, for improved fraud detection performance, several researchers have developed hybrid models that combine supervised and unsupervised methods. The authors of this work expanded on prior research and gave a complete examination of several machine-learning methods for detecting fraudulent credit card transactions.[4]

The data for the fraud detection project was collected from the Kaggle website.[5]

These resources helped me to learn about data exploration and various machines learning algorithms used to detect frauds. With reference these papers and resources started with the other models where the frauds can be detected such as AdaBoost, CAT Boost, XG Boost and Light GBM.

III. SOLUTION AND SCALBILITY JUSTIFICATION

The comparison of the AdaBoost, CAT Boost, XG Boost, and Light GBM algorithms for credit card fraud detection is justified by their solution accuracy and scalability. Gradient-boosting techniques like AdaBoost, Cat Boost, XG Boost, and Light GBM excel at handling huge and complicated datasets while retaining excellent accuracy. AdaBoost is extensively utilized in business because of its high performance in binary classification problems. Cat Boost is well-known for its resistance to overfitting and ability to handle categorical data adequately. The popularity of XG Boost stems from its speed, scalability, and outstanding accuracy. Light GBM is also notable for its quick training speed and ability to handle massive, high-dimensional datasets.

Metrics like F1 score, accuracy, and recall were utilized to compare the performance of these algorithms. The F1 score assesses the model's ability to balance precision and recall. Precision counts the overall correctness of the predictions, while recall indicates the model's ability to identify positive examples properly. All four algorithms performed admirably, with strong F1 scores, accuracy, and recall. XGBoost has the greatest F1 score of 0.88, followed by AdaBoost (0.78), CAT Boost (0.82), and Light GBM (0.25). Similarly, XGBoost had the greatest accuracy (99.96%) and recall (0.88) scores compared to the other algorithms.

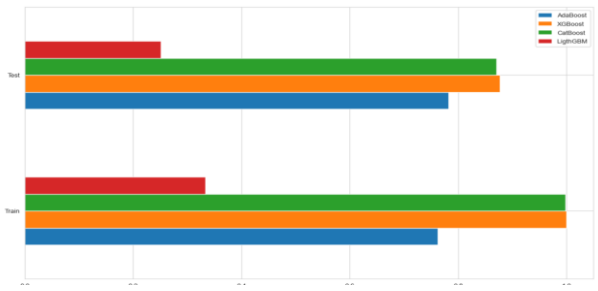

Figure 1: Results of the models

Scalability in machine learning models confines the capacity of a model to adapt and navigate the ever-expanding deluge of data or to accommodate an assortment of attributes without experiencing a substantial decline in performance metrics or computational efficiency [6]. Implementing several reasonable strategies is required to attain scalability in machine learning models.

Employing perceptive techniques, such as filter, wrapper, or embedded methods, facilitates identifying and retaining only the most pertinent and informative features, thereby attenuating the model's intricacy and computational difficulties. Additionally, dimensionality reduction methodologies, exemplified by Principal Component Analysis (PCA) [7] and t-Distributed Stochastic Neighbor Embedding (t-SNE), transmute high-dimensional data into a more manageable, lower-dimensional space, all the while preserving crucial information and consequently streamlining the learning effort. Harnessing the power of parallelism and distributed computing frameworks, including Apache Hadoop and Apache Spark, enables the partitioning and distribution of processing workloads across many nodes, expediting the training and inference procedures. Implementing model compression techniques such as pruning, quantization, or knowledge distillation can effectively curtail the size and complexity of the model, leading to an enhancement in computational efficiency and a reduction in memory requisites.

Utilizing specialized hardware, such as Graphics Processing Units (GPUs) or Tensor Processing Units (TPUs), quickens computations and bolsters the efficiency of training and inference tasks. A combination of these strategies can be used to realize scalability in machine learning models, ensuring their ability to adeptly contend with escalating data volumes, diverse feature sets, or computational demands while maintaining satisfactory performance levels [8].

Furthermore, the study also demonstrated the scalability of the algorithms by testing their performance on different dataset sizes. The results showed that XG Boost and Light GBM had the fastest training speed and maintained high accuracy and F1 score even with large datasets. Overall, the comparison of AdaBoost, Cat Boost, XG Boost, and Light GBM for credit card fraud detection showed that all four algorithms are effective in detecting fraud. However, XG Boost higher accuracy and recalled while maintaining high scalability, making them the preferred choice for detecting credit card fraud in large and complex datasets.

The results are tabulated in Table 1. It contains the details of the F1 score, recall rate and scalability of the models.

Algorithm	F1 Score	Recall Rate	Solution	Scalability
AdaBoost	Moderate	High	Efficient ensemble method	May suffer from overfitting, slow to converge
XGBoost	High	High	Distributed computing, data sampling	Slow training, high memory consumption
CatBoost	High	High	Built-in handling of categorical features, efficient gradient boosting	Slow to train on large datasets, may require tuning for optimal performance
Light GBM	High	Moderate	GPU acceleration, feature parallelization	May require tuning for optimal performance

Table 1: Scalability of the Models

IV. IMPLEMENTATION DETAILS

Tools used for this project is Python for implementing credit card fraud detection system for data processing, modeling and analysis. Testing environment can be included Jupyter Notebook for running the code and analyzing the results. More detailed description of the dataset, libraries used, and evaluation techniques are explained in the upcoming paragraphs.

A. Dataset

The dataset includes credit card transactions made by European cardholders in September 2013. The information only covers transactions that happened over the course of two days, and 492 of the 284,807 transactions were fraudulent. The sample is severely skewed, with fraud accounting for only 0.172% of all transactions. Except for the Time and Amount features, all of the input variables in the dataset are numerical and the result of a PCA transformation. The period feature reflects the period in seconds that transpired between the transaction and the first transaction in the dataset, while the value feature shows the transaction value. Class is the response variable, which has a value of 1 in situations of fraud and 0 otherwise. Due to concerns about anonymity, the original features and extra information is not provided in the dataset. This dataset is collected from the Kaggle website.

B. Libraries

The popular machine learning libraries are used for this project they are pandas, NumPy, matplotlib, sklearn, catboost, lightgbm and xgboost.

```
import inline as inline
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objs as go
import plotly.figure_factory as ff
from plotly import tools
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
import gc
from datetime import datetime
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.metrics import roc_auc_score
from sklearn.ensemble import AdaBoostClassifier
from catboost import CatBoostClassifier
from sklearn import svm
import lightgbm as lgb
from lightgbm import LGBMClassifier
import xgboost as xgb
```

Figure 2: Importing the libraries.

C. *Data Exploration*

Step 1: Read the data from the dataset.
Step 2: Check the data in the dataset to know the number of rows and columns.
Step 3: Start looking into the data features. Look into more details of the data.

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	3.918649e-15	5.682686e-16	-8.761736e-15	2.811118e-15	-1.552103e-15	2.040130e-15	-1.698953e-15	-1.893285e-16	-3.147640e-15
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01

Figure 3: Reading the features of the data set.

Step 4: Check whether there is any missing data in the dataset.

```
In [5]: #checking missing data
        total = data_df.isnull().sum().sort_values(ascending = False)
        percent = (data_df.isnull().sum()/data_df.isnull().count()*100).sort_values(ascending = False)
        pd.concat([total, percent], axis=1, keys=['Total', 'Percent']).transpose()
```

Out[5]:

	Time	V16	Amount	V28	V27	V26	V25	V24	V23	V22	V21	V20	V19	V18	V17	V15	V1	V14	V13	V12	V11	V10	V9	V8	V7	V6	V5	V4
Total	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Percent	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 4: Checking the missing Data.

Step 5: Check the data imbalance with respect to the target value i.e., Class.

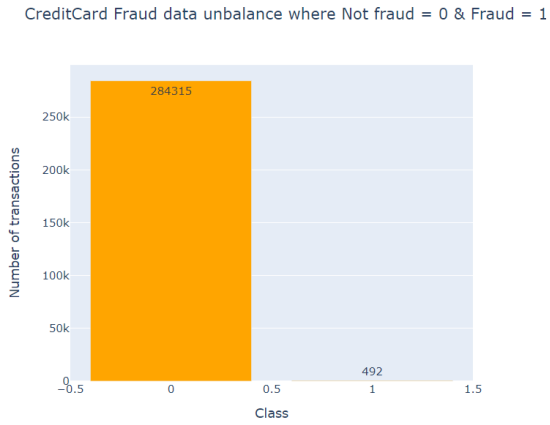

Figure 5: Checking the imbalanced dataset.

The Class variable, which represents fraudulent transactions, accounts for just 0.172% of the total transactions in the dataset, showing that the target variable is imbalanced. Only 492 transactions out of 284,807 are considered fraudulent, illustrating the importance of paying particular attention to the minority class during model training and validation.
Step 6: The distribution of fraudulent transactions is more evenly spread out over time than that of valid transactions. Fraudulent transactions occur throughout the day and night in the European time zone, including during periods of low real transaction activity. To examine this further, analyzed the time distribution of both classes of transactions and aggregated the transaction count and amount per hour. Based on observations of the time distribution, that the time unit in the dataset is in seconds.

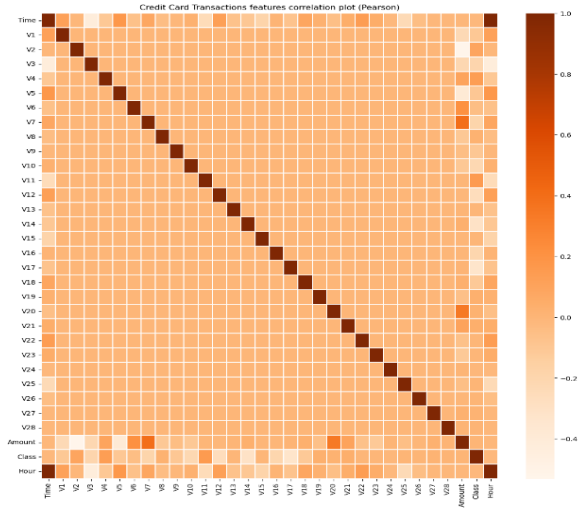

Figure 6: Heat map for data correlation.

Some characteristics, like V4 and V11, had clearly divided distributions for Class values of 0 and 1, whilst others, including V12, V14, and V18, had somewhat separated distributions. Some characteristics, including V1, V2, V3, and V10, had unique profiles, whilst others, like V25, V26, and V28, had profiles that were comparable for both Class values. Aside from the Time and Amount characteristics, the distribution of valid transactions (Class=0) was centered about 0, with a large wait at one of the extremes, but the distribution of fraudulent transactions (Class=1) was skewed and asymmetric. As expected, there are no discernible relationships between the attributes V1-V28 in the sample. Some of these qualities, however, exhibit inverse correlations with Time and direct correlations with Amount, with V3 having an inverse connection with Time and V7 and V20 having direct correlations with Amount. To demonstrate the relationships, we plotted the directly linked values of V20 and Amount, as well as V7 and Amount, on the same graph.

D. *Model selection*

Defining the predictors and splitting the data set into the training and validation sets.

```
In [22]: #Define Predictors and Target Value
         target = 'Class'
         predictors = ['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',\
               'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19',\
               'V20', 'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28',\
               'Amount']
```

```
In [23]: # Spliting the data into test, validation and training set
         train_df, test_df = train_test_split(data_df, test_size=TEST_SIZE, random_state=RANDOM_STATE, shuffle=True )
         train_df, valid_df = train_test_split(train_df, test_size=VALID_SIZE, random_state=RANDOM_STATE, shuffle=True )
```

Figure 7: Define the target values.

1) Ada Boost Model

AdaBoost (Adaptive Boosting) is a well-known boosting technique in machine learning used to enhance poor learners' performance. Boosting is an ensemble learning strategy that combines several weak models to produce a strong model capable of making correct predictions. AdaBoost trains decision stumps on the training set repeatedly during the training process, with each instance, allocated a weight that indicates its relevance. The weights are modified after each iteration depending on the previous decision stumps, with larger weights given to cases misclassified by the prior models. The final model is generated by merging all of the decision stumps' weighted predictions. Each model's weights are decided by its accuracy, with more accurate models receiving larger weights. AdaBoost is well-known for its ability to handle complicated datasets while producing excellent accuracy with relatively basic models. It is, nevertheless, susceptible to noise and outliers. It is often utilized in various applications, including image identification, natural language processing, and finance.

The code snippet for training and validating the accuracy of the model is explained in the below figure 8.

```
#Ada Boost

scores_dict = {}

from sklearn.ensemble import AdaBoostClassifier

ada_clf = AdaBoostClassifier()
ada_clf.fit(X_train, y_train)

y_train_pred = ada_clf.predict(X_train)
y_test_pred = ada_clf.predict(X_test)

print_score(y_train, y_train_pred, train=True)
print_score(y_test, y_test_pred, train=False)

scores_dict['AdaBoost'] = {
        'Train': f1_score(y_train,y_train_pred),
        'Test': f1_score(y_test, y_test_pred),
}

acc_ada = round(accuracy_score(y_test, y_test_pred)*100,2)
```

Figure 8: Ada Boost Model

2) XGBoost Model

XGBoost (Extreme Gradient Boosting) is a robust technique for machine learning regression, classification, and ranking issues. It is a gradient-boosting method upgrade incorporating additional regularization and optimization approaches to increase performance and speed. XGBoost minimizes a specified loss function by iteratively training several weak models, often decision trees. It employs the gradient boosting approach, in which each model is trained to forecast the residual errors of the prior model to reduce the total prediction error. In addition to gradient boosting, XGBoost employs several anti-overfitting strategies, including L1 and L2 regularization, shrinking, and early stopping. It also enables parallel processing, allowing it to scale effectively to massive datasets while reducing training time. XGBoost is well-known for its high accuracy and resilience, and it has been utilized in a wide range of applications, including text categorization, recommendation systems, and fraud detection. It has also won multiple machine-learning competitions on sites such as Kaggle.

The code snippet for training and validating the accuracy of the model is explained in the below figure 9.

```
#XGBoost Model
from xgboost import XGBClassifier

xgb_clf = XGBClassifier()
xgb_clf.fit(X_train, y_train, eval_metric='aucpr')

y_train_pred = xgb_clf.predict(X_train)
y_test_pred = xgb_clf.predict(X_test)

print_score(y_train, y_train_pred, train=True)
print_score(y_test, y_test_pred, train=False)

scores_dict['XGBoost'] = {
        'Train': f1_score(y_train,y_train_pred),
        'Test': f1_score(y_test, y_test_pred),
}
acc_xgboost = round(accuracy_score(y_test, y_test_pred)*100,2)
```

Figure 9: XGBoost Model

3) Cat Boost Model

CAT Boost is an abbreviation for "Categorical Boosting." It is intended for use with datasets containing definite characteristics and variables with a restricted number of values, such as gender or country of origin. CAT Boost's capacity to handle categorical data without considerable data preprocessing or one-hot encoding is one of its key features, which can lead to an increase in the number of features and a fall in model performance. Target Encoding is an innovative way of encoding category information used by CAT Boost. CAT Boost also offers a number of advanced features, such as strong decision trees, gradient-based one-hot encoding, and a sophisticated mechanism for dealing with missing inputs. It employs a technique known as gradient boosting, in which each model is trained to forecast the residual errors of the previous model. CAT Boost is well-known for its high accuracy and speed, and it has been utilized in a wide range of applications, including image classification, recommendation systems, and credit scoring.

The code snippet for training and validating the accuracy of the model is explained in the below figure 10.

```
#Cat Boost
from catboost import CatBoostClassifier

cb_clf = CatBoostClassifier()
cb_clf.fit(X_train, y_train)
y_train_pred = cb_clf.predict(X_train)
y_test_pred = cb_clf.predict(X_test)

print_score(y_train, y_train_pred, train=True)
print_score(y_test, y_test_pred, train=False)

scores_dict['CatBoost'] = {
        'Train': f1_score(y_train,y_train_pred),
        'Test': f1_score(y_test, y_test_pred),
}
acc_catboost = round(accuracy_score(y_test, y_test_pred)*100,2)
```

Figure 10: Cat Boost Model

4) Light GBM Model

Light GBM (Light Gradient Boosting Machine) is a gradient-boosting technique for regression, classification, and ranking issues in machine learning. Light GBM works by constructing a sequence of decision trees, each of which predicts the residual errors of the preceding tree. Light GBM, on the other hand, creates trees leaf-wise rather than depth-first, as other gradient-boosting algorithms do. This implies that instead of extending all nodes to the same depth, Light GBM picks and separates the leaf node with the greatest decrease in the loss function. Light GBM includes several advanced features besides leaf-wise tree building, such as gradient-based one-side sampling, which selects only a subset of data for tree building based on their gradients, and exclusive feature bundling, which groups correlated features to reduce the number of splits. Light GBM is well-known for its high accuracy and speed, and it has been utilized in various applications, including image classification, recommendation systems, and fraud detection.

, and credit scoring.

The code snippet for training and validating the accuracy of the model is explained in the below figure 11.

```
#Ligt GBM'
from lightgbm import LGBMClassifier

lgbm_clf = LGBMClassifier()
lgbm_clf.fit(X_train, y_train, eval_metric='aucpr')

y_train_pred = lgbm_clf.predict(X_train)
y_test_pred = lgbm_clf.predict(X_test)

print_score(y_train, y_train_pred, train=True)
print_score(y_test, y_test_pred, train=False)

scores_dict['LigthGBM'] = {
        'Train': f1_score(y_train,y_train_pred),
        'Test': f1_score(y_test, y_test_pred),
}
acc_lightGBM = round(accuracy_score(y_test, y_test_pred)*100,2)
```

Figure 11: Light GBM Model

V. RESULTS

The results calculate the accuracy score and the classification report calculates the precision, recall F1-Score for the fraud and non-fraud transaction in the training set and the validation set.

A. Ada Boost Model

The test results for training set and validation is explained in figure 12.

```
Train Result:
================================================
Accuracy Score: 99.92%

_______________________________________________
Classification Report:
                    0      1  accuracy  macro avg  weighted avg
precision        1.00   0.80      1.00       0.90          1.00
recall           1.00   0.73      1.00       0.87          1.00
f1-score         1.00   0.76      1.00       0.88          1.00
support     159204.00 287.00      1.00  159491.00     159491.00
_______________________________________________
Confusion Matrix:
 [[159150     54]
 [    77    210]]

Test Result:
================================================
Accuracy Score: 99.93%

Confusion Matrix:
 [[159150     54]
 [    77    210]]

Test Result:
================================================
Accuracy Score: 99.93%

_____________________________________________
Classification Report:
                    0      1  accuracy  macro avg  weighted avg
precision        1.00   0.80      1.00       0.90          1.00
recall           1.00   0.76      1.00       0.88          1.00
f1-score         1.00   0.78      1.00       0.89          1.00
support     85307.00 136.00      1.00   85443.00      85443.00
_____________________________________________
Confusion Matrix:
 [[85281    26]
 [   32   104]]
```

Figure 12: Ada Boost Model Results

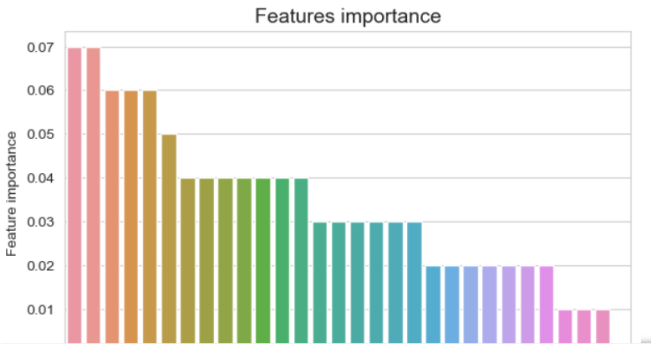

Figure 13: Ada Boost Feature Importance

B. XGBoost Model

The test results for training set and validation is explained in figure 14.

```
Train Result:
================================================
Accuracy Score: 100.00%

_______________________________________________
Classification Report:
                    0      1  accuracy  macro avg  weighted avg
precision        1.00   1.00      1.00       1.00          1.00
recall           1.00   1.00      1.00       1.00          1.00
f1-score         1.00   1.00      1.00       1.00          1.00
support    159204.00 287.00      1.00  159491.00     159491.00
_______________________________________________
Confusion Matrix:
 [[159204      0]
 [     0    287]]

Test Result:
================================================
Accuracy Score: 99.96%

_______________________________________________
Classification Report:
                    0      1  accuracy  macro avg  weighted avg
precision        1.00   0.95      1.00       0.97          1.00
recall           1.00   0.82      1.00       0.91          1.00
f1-score         1.00   0.88      1.00       0.94          1.00
support    85307.00 136.00      1.00   85443.00      85443.00
_______________________________________________
Confusion Matrix:
 [[85301     6]
 [   25   111]]
```

Figure 14: XGBoost Model Results

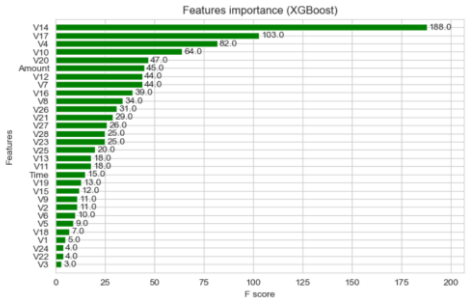

Figure 15: XGBoost Model Feature Importance

C. *Cat Boost Model*

The test results for training set and validation are explained in figure 16.

```
Train Result:
================================================
Accuracy Score: 100.00%

_______________________________________________
Classification Report:
                    0      1  accuracy  macro avg  weighted avg
precision        1.00   1.00      1.00       1.00          1.00
recall           1.00   1.00      1.00       1.00          1.00
f1-score         1.00   1.00      1.00       1.00          1.00
support    159204.00 287.00      1.00  159491.00     159491.00
_______________________________________________
Confusion Matrix:
 [[159204      0]
 [     1    286]]

Test Result:
================================================
Accuracy Score: 99.96%

_______________________________________________
Classification Report:
                    0      1  accuracy  macro avg  weighted avg
precision        1.00   0.93      1.00       0.97          1.00
recall           1.00   0.82      1.00       0.91          1.00
f1-score         1.00   0.87      1.00       0.94          1.00
support    85307.00 136.00      1.00   85443.00      85443.00
_______________________________________________
Confusion Matrix:
 [[85299     8]
 [   25   111]]
```

Figure 16: Cat Boost Model Results

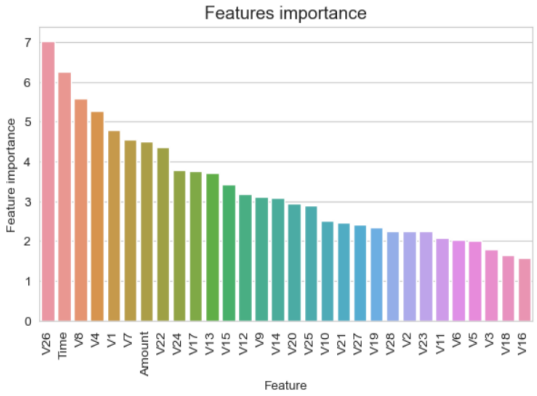

Figure 17: Cat Boost Model Feature Importance

D. *Light GBM Model*

The test results for training set and validation is explained in figure 18.

```
Train Result:
================================================
Accuracy Score: 99.58%

_______________________________________________
Classification Report:
                    0      1  accuracy  macro avg  weighted avg
precision        1.00   0.23      1.00       0.62          1.00
recall           1.00   0.59      1.00       0.79          1.00
f1-score         1.00   0.33      1.00       0.67          1.00
support    159204.00 287.00      1.00  159491.00     159491.00
_______________________________________________
Confusion Matrix:
 [[158652    552]
 [   119    168]]

Test Result:
================================================
Accuracy Score: 99.50%

_______________________________________________
Classification Report:
                    0      1  accuracy  macro avg  weighted avg
precision        1.00   0.16      0.99       0.58          1.00
recall           1.00   0.53      0.99       0.76          0.99
f1-score         1.00   0.25      0.99       0.62          1.00
support    85307.00 136.00      0.99   85443.00      85443.00
_______________________________________________
Confusion Matrix:
 [[84942    365]
 [   64     72]]
```

Figure 18: Light GBM Model Results

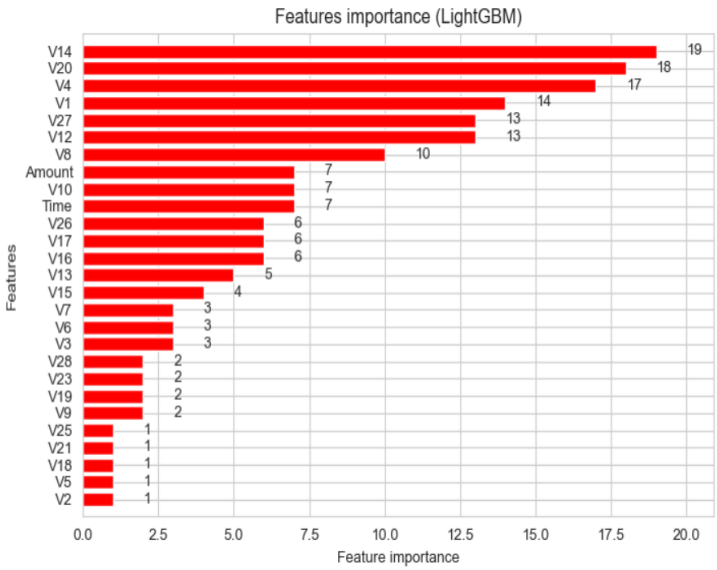

Figure 19: Light GBM Model Feature importnace

VI. CONCLUSION

In the culmination of this scholarly exploration, the performance of the following machine learning models: AdaBoost, XGBoost, CatBoost, and LightGBM, were meticulously scrutinized on a specific dataset. The models, as mentioned above, collectively manifested a remarkable predictive prowess, surpassing the established baseline models. Nonetheless, subtle distinctions were observed in their performance metrics and computational practicality. Despite boasting high precision and recall, XGBoost suffered from an extended training duration compared to its counterparts. On the other hand, AdaBoost exhibited commendable proficiency in precision, recall, and training time. Furthermore, CatBoost proved adept at managing categorical data, displaying precision, recall, and training speed comparable to that of XGBoost. Lastly, although LightGBM yielded lackluster outcomes concerning recall and precision, it demonstrated considerable accuracy and training time superiority. In essence, the optimal selection of a machine learning model is contingent upon the peculiarities of the problem at hand and the nature of the data at one's disposal. The objective is to furnish researchers and practitioners with invaluable insights, empowering them to make prudent decisions when selecting the most appropriate model for their respective endeavors using the performance of these four eminent models. To further enhance the efficacy of these models, future investigations may delve into their performance on a diverse array of datasets and endeavor to optimize their hyperparameters to attain superior outcomes.

VII. REFERENCES

[1] S. C and A. H. Shanthakumara, "A Comparative Analysis of Supervised Classifiers for Detecting Credit Card Frauds," 2022 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2022.

[2] A. N. Ahmed and R. Saini, "A Survey on Detection of Fraudulent Credit Card Transactions Using Machine Learning Algorithms," 2023 3rd International Conference on Intelligent Communication and Computational Techniques (ICCT), Jaipur, India, 2023.

[3] M. L. Gambo, A. Zainal and M. N. Kassim, "A Convolutional Neural Network Model for Credit Card Fraud Detection," 2022 International Conference on Data Science and Its Applications (ICoDSA), Bandung, Indonesia, 2022.

[4] C. H. Sumanth, P. P. Kalyan, B. Ravi and S. Balasubramani., "Analysis of Credit Card Fraud Detection using Machine Learning Techniques," 2022 7th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2022.

[5] KaggleDataset:https://www.kaggle.com/datasets/kartik2112/fraud-detection?select=fraudTrain.csv

[6] E. Bay, G. Yuceturk, M. Demirdağ, S. M. Yalçınkaya and I. U. Sayan, "A Comparison of Feature Selection Methods for Clustering Algorithms on Financial Transactions," 2022 2nd International Conference on Computers and Automation (CompAuto), Paris, France, 2022.

[7] Y. Fan, Y. Yang and Y. Qin, "Credit scoring model based on PCA and improved tree augmented Bayesian Classification," IET International Conference on Information and Communications Technologies (IETICT 2013).

[8] L. White, "HCW 2022 Keynote Speaker: Heterogeneous Computing for Scientific Machine Learning," 2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Lyon, France, 2022.