[1]Sahithi Arjun

[2]Nageswara Rao Moparthi

Fraud Detection in Banking Data by Machine Learning Techniques

Abstract: - An important part of the study is looking at how to use machine learning to find scams in bank data. This is a very big problem in the banking world, where finding and stopping scams is very important. The study adds class weight-tuning hyper parameters to make scam detection better. These factors make it easier for the model to tell the difference between real and fake transactions, which makes the scam detection system more accurate. Cat Boost, Light GBM, and XG Boost are three well-known machine learning methods that are used in a smart way in this work. Each program is good at something different, and using them together should make the scam detection method work better overall.To fine-tune the hyper parameters, deep learning methods are used in the work. This combination improves the fraud detection system's speed and ability to change, making it better at finding new fraud schemes. Real-world data are used in the project to do thorough reviews. These tests show that using Light GBM and XG Boost together works better than other ways when looking at different factors. By this measure, the suggested method is better at finding fake actions than other approaches. It has a Stacking Classifier that combines results from the Random Forest and Light GBM classifiers with certain settings. Using a Gradient Boosting Classifier as the final predictor, this ensemble method improves the accuracy of predictions by using the best features of different models.

Keywords: Bayesian optimization, data Mining, deep learning, ensemble learning, hyper parameter, unbalanced data, machine learning..

I. INTRODUCTION

As financial institutions have grown and web-based e-commerce has become more popular, there have been a lot more financial activities in the past few years. Fraudulent activities are becoming more common in online banking, and it has always been hard to spot fraud [1, 2]. Since credit cards have changed over time, so has the way credit card theft is done. Credit card fraud has always been updated, and criminals try to seem authentic. Fraudsters work hard to seem legitimate. Instead of learning how fraud detection systems function, they add to them, making fraud detection tougher. Because of this, experts are always looking for new ways to do things or ways to make the ones they already have work better [3].

Fraudsters often get what they want by taking advantage of flaws in security, control, and tracking in business software. Truth be told, technology can help fight scams [4]. To stop theft from happening again, it's important to find it right away [5]. Criminal or unethical lying with the goal of making money or getting something nice for yourself is what fraud is. Credit card theft occurs when someone uses a stolen credit card to make transactions in person or online. Cardholders frequently give up their card information, expiry date, and verification number over the phone or online [6], thus scams might happen.

To escape loses due to fraud, there are two tools that can be used: fraud protection and fraud identification. The goal of fraud protection is to keep theft from happening in the first place. On the other hand, scam detection is needed when someone tries to make a fake exchange. [7]. Fraud detection in banks is thought of as a binary classification problem, where material is either real or fake [8]. Because there is a lot of financial data and files hold a lot of transaction data, it is either not possible or takes a long time to look through it all by hand and find trends for illegal transactions. Because of this, methods built on machine learning are very important for finding and predicting scams [9].

[1,2] Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Andhra Pradesh, Vaddeswaram, 522302, India

1 sahithiarjun9@gmail.com

[2*] mnrphd@gmail.com

Corresponding author Name and email ID: Nageswarara Rao Moparthi , mnrphd@gmail.com

Large files and finding scams can be done more quickly and accurately with machine learning techniques and a lot of computer power. [15] Deep learning and machine learning algorithms can also solve problems quickly and correctly in real time [10]. We describe an effective credit card fraud detection method in this study. It uses improved algorithms LightGBM, XGBoost, CatBoost, logistic regression, majority voting, deep learning, and hyperparameter settings and was tested on public datasets. A perfect fraud detection system would catch more fraud and be accurate. All findings should be appropriately recognized to increase client confidence and prevent the bank from losing money due to misidentification.

II. LITERATURE SURVEY

E-commerce fraud is difficult to prevent since fraud patterns vary. [1] This study introduces fraud islands

(link analysis) and multi-layer machine learning models [10, 15, 20] to identify scam tendencies. Link analysis examines the networks of fraudulent organizations to identify hidden, complex fraud patterns. This produces fraud islands. Scam trends vary, hence a multi-layer model is utilized. Current fraud labels are determined by banks' refusal judgments, human review agents' rejection decisions, fraud warnings, and refund requests. Fraud risk prevention groups like banks, human review teams, and fraud machine learning models should be able to identify distinct fraud kinds. Combining many machine learning models trained with distinct fraud labels improved fraud choice accuracy [10].

There are a lot more government and private health programs than there used to be, and the number of bogus billing cases is also going up. [9] Fraudulent deals in healthcare systems are hard to spot because of the complicated connections between changing factors like customers, doctors, and services. To make health support programs more open, we need to create smart fraud detection models that can find holes in the way things are done now. This way, cases of fake medical bills can be found and stopped. It is also important to find the best balance between the client's medical perks and the service provider's costs. [2] Using sequence mining ideas, this study describes a new process-based fraud detection method for finding scams involving insurance claims in the healthcare system. Instead of employing sequence generation of services within each area to uncover fraud, much current research has focused on amount-based analysis or medication vs illness sequencing analysis. With the suggested method, common sets with different pattern lengths are made. For each series, the confidence level and confidence values are found. The sequence rule engine comes up with common sequences and confidence levels for each hospital's specialty. It then checks these against the real patient numbers [2, 7, 9]. This finds strange things because neither order would match the rule engine's patterns. The process-based fraud detection method has been tested using transactional data from a local hospital over the last five years. This data shows that many cases of fraud have been recorded.

Since the economy has been doing well, the number of credit card transactions has been going through the roof. A lot of scam businesses are also growing quickly. In this situation, finding theft is becoming a more important issue. But the percentage of scams is much lower than the percentage of smart transactions. This makes the problem much harder to solve because of the imbalance dataset. This study [3] is mostly about how to use boosting methods to solve the problem of finding credit card scams. It also includes a short comparison of these boosting methods [29, 30].

Credit card theft has become a very important problem around the world because of the huge growth of e-commerce and the number of ways to pay online. Recently, there has been a lot of interest in using machine learning algorithms as a data mining method to find credit card scams. But there are a lot of problems, like the fact that there aren't many freely available data sets, class sizes that aren't fair at all, different kinds of scams, and so on. [5] This paper looks at how well three machine learning algorithms—Random Forest, Support Vector Machine, and Logistic Regression—find fraud in real-life credit card transaction data [20]. We use the SMOTE selection method to fix the problem of unequal class sizes. The issue of scam trends that change all the time is looked at by using studies that use gradual learning of certain machine learning methods. Precision and memory are two well-known metrics that are used to judge how well the methods work.

Credit card theft is a big issue in the business world. Fraud with credit cards costs a lot of money every year. Because of concerns about privacy, there aren't many studies that look at real credit card data. There are machine learning methods [10, 15, 20] used in this study to find credit card scams. At first, standard versions are used. Following that, mixed methods are used that combine Ada Boost and majority vote. A credit card data set that is open to everyone is used to test how well the model works. [6]This is followed by an analysis of real credit card data from a bank. Noise is also added to the data sets to test how well the methods work in general. The test results show that the majority vote method is a good way to find credit card scams.

In the United States, healthcare theft is a costly white-collar crime that does not not hurt anyone. The costs of theft are passed on to the public through higher fees or major harm to recipients [2, 7]. To fight this threat to society, digital systems that find healthcare scams need to change quickly. It's hard to use digital advances in healthcare in the US because of the country's complicated, different data methods and different health models. Healthcare fraud detection provides agents with suggestions they may investigate for recoupments, refunds, or referrals to the relevant authorities or organizations. This essay [7] summarizes previous healthcare fraud detection technologies. It lists peer-reviewed studies in this field, along with their main aims, conclusions, and data types. We'll discuss issues with using these technologies with real-world healthcare data. The authors propose some research subjects for other specialists to address these gaps.

III. METHODOLOGY

Proposed Work

Using machine learning, the project creates a sophisticated method for finding scams in banking data. It works better by using Bayesian optimization and class weight-tuning, and using algorithms like [29, 30, 31, 32]CatBoost, LightGBM, and XGBoost. The system is made even better by deep learning, and it has been tested thoroughly using real-world data and key measures to make sure it works well at finding and stopping theft. It has a Stacking Classifier that combines results from the RandomForest and LightGBM [17, 28] classifiers with certain settings. Using a GradientBoostingClassifier as the final predictor, this ensemble method improves the accuracy of predictions by using the best features of different models. A simple Flask framework combined with SQLite has also been created, complete with signup and signin features for effective user testing and to make the system more accessible and useful in real-life scam detection scenarios.

System Architecture

The system starts with basic data that includes information about credit card transactions, such as marks and features that show whether the transaction is real or fake. To get the data ready for machine learning, it goes through preparation, which includes selecting and extracting features. The dataset is split into two parts: a training set for building models and a test set for checking how well the models work. Bayesian optimization is used to make the hyperparameters of machine learning methods work better. Fivefold cross-validation is used with machine learning methods like CatBoost, [17] LightGBM, and XGBoost on the training data to make sure the model is stable. As an addition to the project, we have also looked into stacked algorithm. Several review measures are used to see how well the models find credit card scams while reducing the number of fake hits.

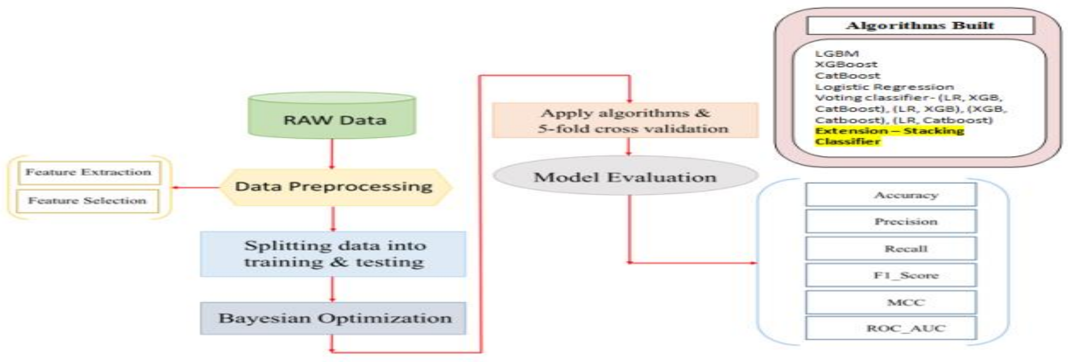

RAW Data
Feature Extraction
Feature Selection
Data Preprocessing
Splitting data into training & testing
Bayesian Optimization
Apply algorithms & 5-fold cross validation
Model Evaluation
Algorithms Built
LGBM
XGBoost
CatBoost
Logistic Regression
Voting classifier - (LR, XGB, CatBoost), (LR, XGB), (XGB, Catboost), (LR, Catboost)
Extension – Stacking Classifier
Accuracy
Precision
Recall
F1_Score
MCC
ROC_AUC

Fig. 1: Proposed Architecture

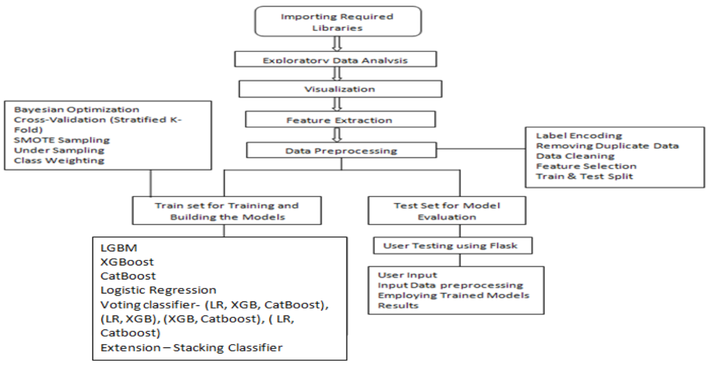
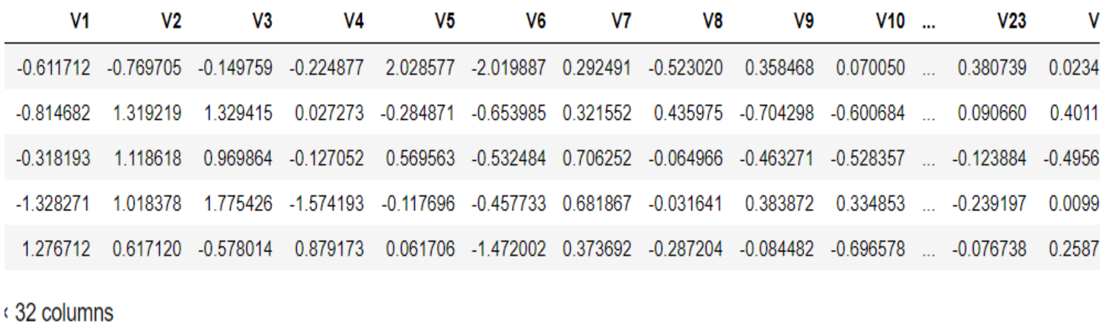


Fig. 2 : Flow Chart

Dataset collection:

We trained machine learning models with the Credit Card Fraud Detection dataset from Kaggle. The original dataset had different traits linked to transactions, such as "Amount," "Time," and "V1" through "V28."Information that was important was kept secret, including details about the original features.



V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V23	V
-0.611712	-0.769705	-0.149759	-0.224877	2.028577	-2.019887	0.292491	-0.523020	0.358468	0.070050	...	0.380739	0.0234
-0.814682	1.319219	1.329415	0.027273	-0.284871	-0.653985	0.321552	0.435975	-0.704298	-0.600684	...	0.090660	0.4011
-0.318193	1.118618	0.969864	-0.127052	0.569563	-0.532484	0.706252	-0.064966	-0.463271	-0.528357	...	-0.123884	-0.4956
-1.328271	1.018378	1.775426	-1.574193	-0.117696	-0.457733	0.681867	-0.031641	0.383872	0.334853	...	-0.239197	0.0099
1.276712	0.617120	-0.578014	0.879173	0.061706	-1.472002	0.373692	-0.287204	-0.084482	-0.696578	...	-0.076738	0.2587

32 columns

Fig. 3 : Dataset

Dataset User Link:

https://www.kaggle.com/datasets/arockiaselciaa/creditcardcsv

Data Processing:

Data handling is the process of turning unstructured data into knowledge that businesses can use. In general, data scientists handle data, which means they gather it, organize it, clean it, check it, analyze it, and turn it into forms that can be read, like graphs or papers. There are three ways to handle data: by hand, mechanically, or electronically. The goal is to make knowledge more useful and decision-making easier. This helps companies run better and make smart strategy decisions more quickly. This is made possible in large part by automated data handling tools, like computer programs. It can help turn big data and other types of data into useful information for decision-making and quality control.

Feature Selection:

Feature selection is the process of picking out the most reliable, useful, and non-redundant traits to use in building a model. As the number and types of records increase, it is important to reduce their sizes in a planned way. One of the main goals of feature selection is to make a prediction model work better and use less computing power.

One of the most important parts of feature engineering is feature selection, which is the process of choosing the most important features to feed into algorithms for machine learning. Feature selection methods get rid of unnecessary or useless features and only keep the ones that are most important to the machine learning model.

This lowers the number of input factors. If you choose which traits are most important ahead of time instead of letting the machine learning model do it, here are the major benefits.

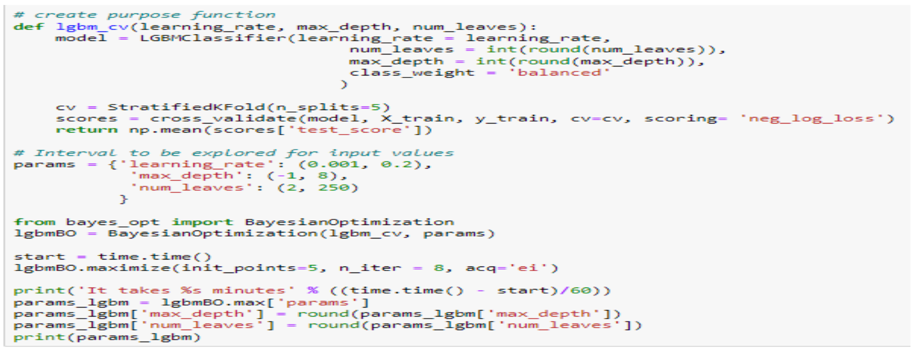
Algorithms:

LGBM(Light Gradient Boosting Machine)

LGBM (Light Gradient Boosting Machine): LGBM is a gradient boosting system that works very well with big datasets and is very fast. It is known for being fast and accurate, which makes it good for jobs like finding scams. LGBM puts together a group of decision trees and improves the boosting process so that the results come together faster [28].

$$V_{j|O}(d) = \frac{1}{n_o}\left(\frac{\left(\sum_{\{x_i\in O: x_{ij}\le d\}} g_i\right)^2}{n^j_{l|O}(d)} + \frac{\left(\sum_{\{x_i\in O: x_{ij}> d\}} g_i\right)^2}{n^j_{r|O}(d)}\right)$$

where $n_O = \sum I[xi \in O]$, $n^j_{l|O}(d) = \sum I[xi \in O : xij \le d]$ and $n^j_{r|O}(d) = \sum I[xi \in O : xij > d]$.



```
# create purpose function
def lgbm_cv(learning_rate, max_depth, num_leaves):
    model = LGBMClassifier(learning_rate = learning_rate,
                                          num_leaves = int(round(num_leaves)),
                                          max_depth = int(round(max_depth)),
                                          class_weight = 'balanced'
                                         )

    cv = StratifiedKFold(n_splits=5)
    scores = cross_validate(model, X_train, y_train, cv=cv, scoring= 'neg_log_loss')
    return np.mean(scores['test_score'])

# Interval to be explored for input values
params = {'learning_rate': (0.001, 0.2),
           'max_depth': (-1, 8),
           'num_leaves': (2, 250)
         }

from bayes_opt import BayesianOptimization
lgbmBO = BayesianOptimization(lgbm_cv, params)

start = time.time()
lgbmBO.maximize(init_points=5, n_iter = 8, acq='ei')

print('It takes %s minutes' % ((time.time() - start)/60))
params_lgbm = lgbmBO.max['params']
params_lgbm['max_depth'] = round(params_lgbm['max_depth'])
params_lgbm['num_leaves'] = round(params_lgbm['num_leaves'])
print(params_lgbm)
```

Fig. 4 : LGBM

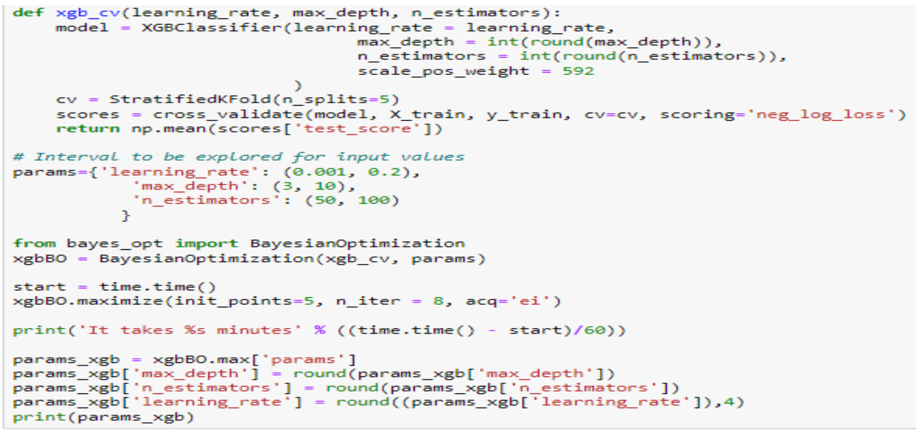
XGBoost(Extreme Gradient Boosting)

Another gradient boosting method that is often used for machine learning jobs is XGBoost. It's known for being strong and working well. XGBoost works well with uneven datasets because it uses a regularized gradient boosting scheme. This is very important for finding scams.

Real value (label) known from the training data-set

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$

Can be seen as f(x + Δx) where x = $\hat{y_i}^{(t-1)}$



```
def xgb_cv(learning_rate, max_depth, n_estimators):
    model = XGBClassifier(learning_rate = learning_rate,
                                       max_depth = int(round(max_depth)),
                                       n_estimators = int(round(n_estimators)),
                                       scale_pos_weight = 592
                                  )
    cv = StratifiedKFold(n_splits=5)
    scores = cross_validate(model, X_train, y_train, cv=cv, scoring='neg_log_loss')
    return np.mean(scores['test_score'])

# Interval to be explored for input values
params={'learning_rate': (0.001, 0.2),
         'max_depth': (3, 10),
         'n_estimators': (50, 100)
        }

from bayes_opt import BayesianOptimization
xgbBO = BayesianOptimization(xgb_cv, params)

start = time.time()
xgbBO.maximize(init_points=5, n_iter = 8, acq='ei')

print('It takes %s minutes' % ((time.time() - start)/60))

params_xgb = xgbBO.max['params']
params_xgb['max_depth'] = round(params_xgb['max_depth'])
params_xgb['n_estimators'] = round(params_xgb['n_estimators'])
params_xgb['learning_rate'] = round((params_xgb['learning_rate']),4)
print(params_xgb)
```

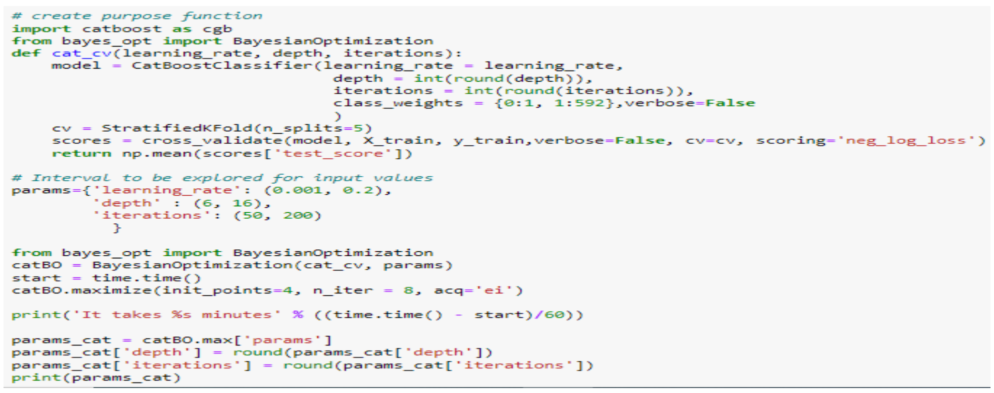
Fig. 5 : XGBoost

CatBoost(Categorical Boosting)

CatBoost is a library for gradient boosting that is meant to work well with category features. It takes care of category data automatically, which makes it easy to work with. Overfitting doesn't affect it too much, and it can be useful when working with real-world banking data [29, 30, 31, 32].

$$\frac{\sum_{i=1}^{N} w_i \log \left(\frac{e^{a_{it_i}}}{\sum_{j=0}^{M-1} e^{a_{ij}}} \right)}{\sum_{i=1}^{N} w_i},$$

$$t \in \{0, \ldots, M-1\}$$



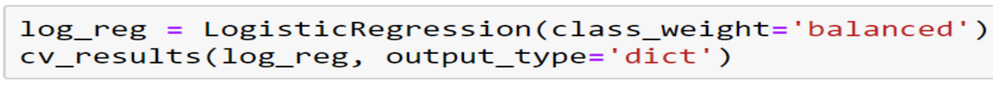
```
# create purpose function
import catboost as cgb
from bayes_opt import BayesianOptimization
def cat_cv(learning_rate, depth, iterations):
    model = CatBoostClassifier(learning_rate = learning_rate,
                               depth = int(round(depth)),
                               iterations = int(round(iterations)),
                               class_weights = {0:1, 1:592},verbose=False
                               )
    cv = StratifiedKFold(n_splits=5)
    scores = cross_validate(model, X_train, y_train,verbose=False, cv=cv, scoring='neg_log_loss')
    return np.mean(scores['test_score'])

# Interval to be explored for input values
params={'learning_rate': (0.001, 0.2),
        'depth' : (6, 16),
        'iterations': (50, 200)
         }

from bayes_opt import BayesianOptimization
catBO = BayesianOptimization(cat_cv, params)
start = time.time()
catBO.maximize(init_points=4, n_iter = 8, acq='ei')

print('It takes %s minutes' % ((time.time() - start)/60))

params_cat = catBO.max['params']
params_cat['depth'] = round(params_cat['depth'])
params_cat['iterations'] = round(params_cat['iterations'])
print(params_cat)
```

Fig. 6 : CatBoost

Logistic Regression

Logistic Regression is one of the most basic binomial classification algorithms. It's not as complicated as group methods like boosting, but it can be used as a starting point for finding fraud. It's easy to understand and can help you figure out how important a feature is.

$$P = \frac{e^{a+bX}}{1 + e^{a+bX}}$$

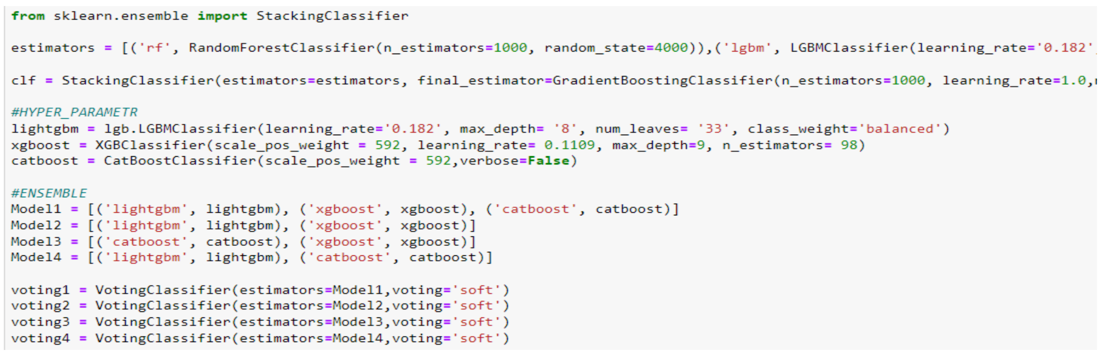


```
log_reg = LogisticRegression(class_weight='balanced')
cv_results(log_reg, output_type='dict')
```

Fig. 7 : Logistic Regression

Voting Classifier

The Voting Classifier takes the results from several machine learning models, like Logistic Regression, XGBoost, and CatBoost, and puts them all together to make a single forecast. This ensemble method uses the combined knowledge of several models, which usually leads to better accuracy and stability. Some of the vote models we've made use different mixes of algorithms [19, 24].

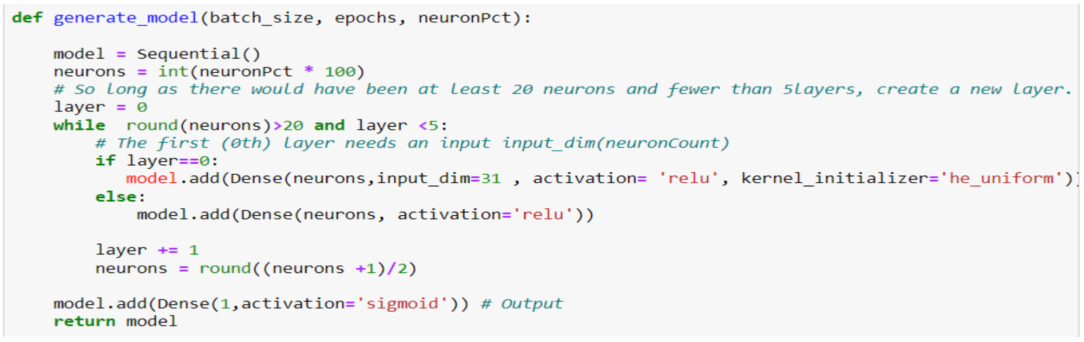


```
from sklearn.ensemble import StackingClassifier

estimators = [('rf', RandomForestClassifier(n_estimators=1000, random_state=4000)),('lgbm', LGBMClassifier(learning_rate='0.182'

clf = StackingClassifier(estimators=estimators, final_estimator=GradientBoostingClassifier(n_estimators=1000, learning_rate=1.0,

#HYPER_PARAMETR
lightgbm = lgb.LGBMClassifier(learning_rate='0.182', max_depth= '8', num_leaves= '33', class_weight='balanced')
xgboost = XGBClassifier(scale_pos_weight = 592, learning_rate= 0.1109, max_depth=9, n_estimators= 98)
catboost = CatBoostClassifier(scale_pos_weight = 592,verbose=False)

#ENSEMBLE
Model1 = [('lightgbm', lightgbm), ('xgboost', xgboost), ('catboost', catboost)]
Model2 = [('lightgbm', lightgbm), ('xgboost', xgboost)]
Model3 = [('catboost', catboost), ('xgboost', xgboost)]
Model4 = [('lightgbm', lightgbm), ('catboost', catboost)]

voting1 = VotingClassifier(estimators=Model1,voting='soft')
voting2 = VotingClassifier(estimators=Model2,voting='soft')
voting3 = VotingClassifier(estimators=Model3,voting='soft')
voting4 = VotingClassifier(estimators=Model4,voting='soft')
```

Fig. 8 : Voting Classifier

Neural Network

A Neural Network is a model for deep learning that is based on the way the brain works. In this case, it can find complicated patterns and connections in the data. One reason neural networks are used is that they can learn complicated scam patterns, especially in big datasets.



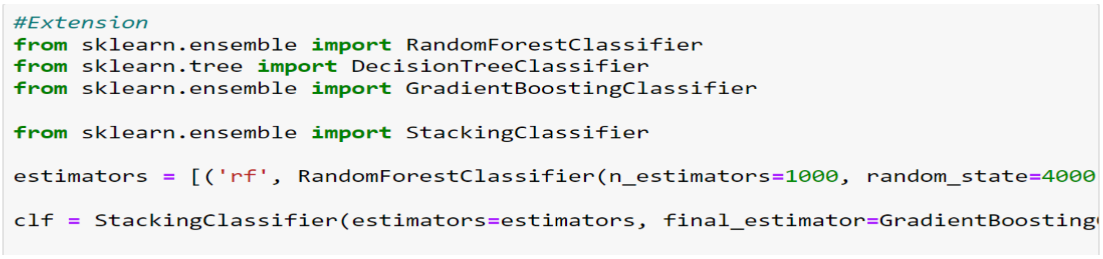
```
def generate_model(batch_size, epochs, neuronPct):

    model = Sequential()
    neurons = int(neuronPct * 100)
    # So long as there would have been at least 20 neurons and fewer than 5layers, create a new layer.
    layer = 0
    while  round(neurons)>20 and layer <5:
        # The first (0th) layer needs an input input_dim(neuronCount)
        if layer==0:
           model.add(Dense(neurons,input_dim=31 , activation= 'relu', kernel_initializer='he_uniform'))
        else:
            model.add(Dense(neurons, activation='relu'))

        layer += 1
        neurons = round((neurons +1)/2)

    model.add(Dense(1,activation='sigmoid')) # Output
    return model
```

Fig. 9 : Neural Network

Stacking Classifier

as an extension we have built a stacking classifier.

An ensemble method called the Stacking Classifier takes results from two base classifiers (RandomForest and LightGBM) and combines them with certain settings. It uses a GradientBoostingClassifier as the final predictor to improve the accuracy of predictions by combining the best features of different models in ensemble learning.



```
#Extension
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier

from sklearn.ensemble import StackingClassifier

estimators = [('rf', RandomForestClassifier(n_estimators=1000, random_state=4000

clf = StackingClassifier(estimators=estimators, final_estimator=GradientBoosting
```

Fig. 10 : Stacking Classifier

IV. EXPERIMENTAL RESULTS

Precision

Precision is the percentage of correctly classified cases or samples compared to those that were correctly classified as hits. So, here is the method to figure out the precision

Precision = True positives/ (True positives + False positives) = TP/(TP + FP)

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Recall

In machine learning, recall is a parameter that shows how well a model can find all the important cases of a certain class. It indicates how effectively a model captures class cases. Divide the number of accurately anticipated positive observations by the total genuine positives.

$$Recall = \frac{TP}{TP + FN}$$

Accuracy

Accuracy is the percentage of right guesses in a classification job. It shows how accurate a model's forecasts are generally.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

F1 Score

The harmonic mean of accuracy and recall is F1. This fair measure accounts for erroneous positives and negatives, therefore it may be used with unbalanced datasets.

$$\text{F1 Score} = 2 * \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} * 100$$

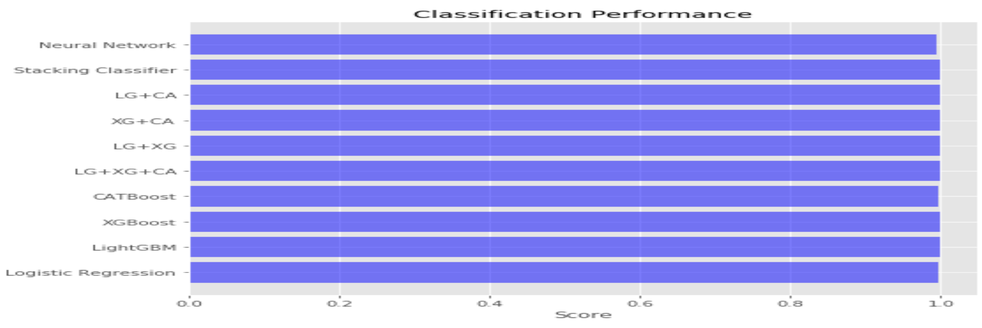
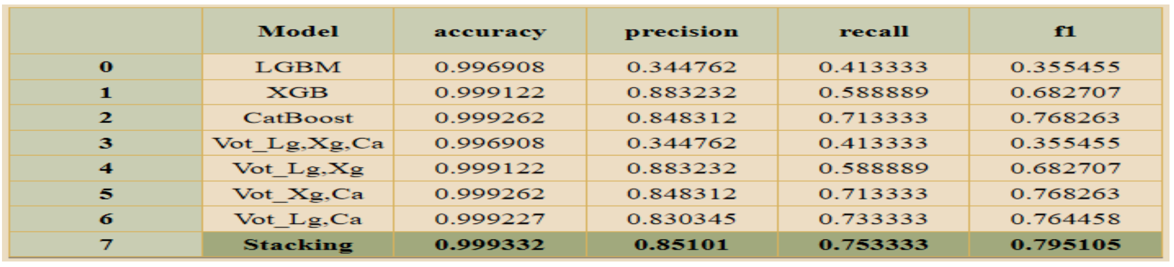


Fig. 11 : Performance Evaluation



	Model	accuracy	precision	recall	f1
0	LGBM	0.996908	0.344762	0.413333	0.355455
1	XGB	0.999122	0.883232	0.588889	0.682707
2	CatBoost	0.999262	0.848312	0.713333	0.768263
3	Vot_Lg,Xg,Ca	0.996908	0.344762	0.413333	0.355455
4	Vot_Lg,Xg	0.999122	0.883232	0.588889	0.682707
5	Vot_Xg,Ca	0.999262	0.848312	0.713333	0.768263
6	Vot_Lg,Ca	0.999227	0.830345	0.733333	0.764458
7	**Stacking**	**0.999332**	**0.85101**	**0.753333**	**0.795105**

Fig. 12 : Performance Evaluation Table

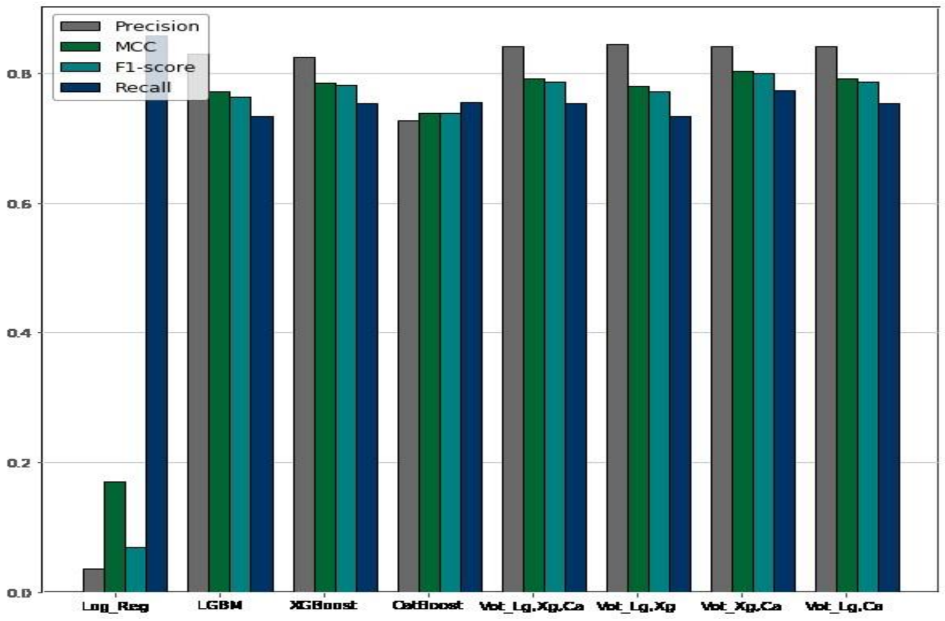


Fig. 13 : Comparison bar graph between previous and obtained value

Fig. 13 provides the differentiation between algorithms using the metrics precision , MCC , F-1 Score , Recall from the obtained output .

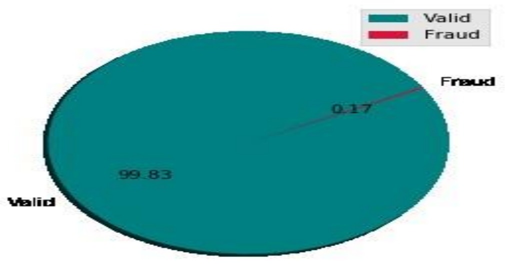


Fig. 14 : Pie Chart for number and percentage of fraudulent vs valid transactions

FIG. 14 represents the facile representation of valid and fraudulent transactions.

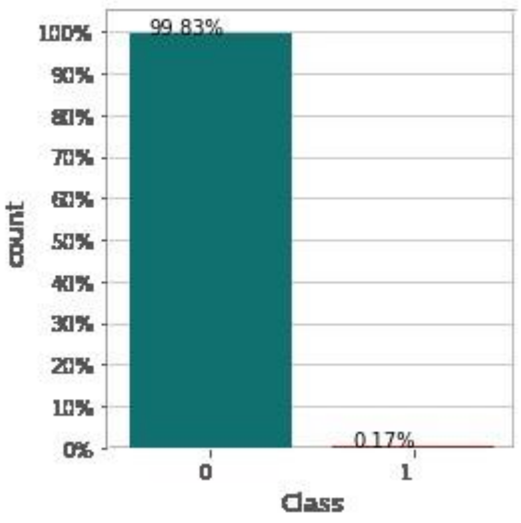


Fig. 15 : Plot for number and percentage of fraudulent vs valid transactions

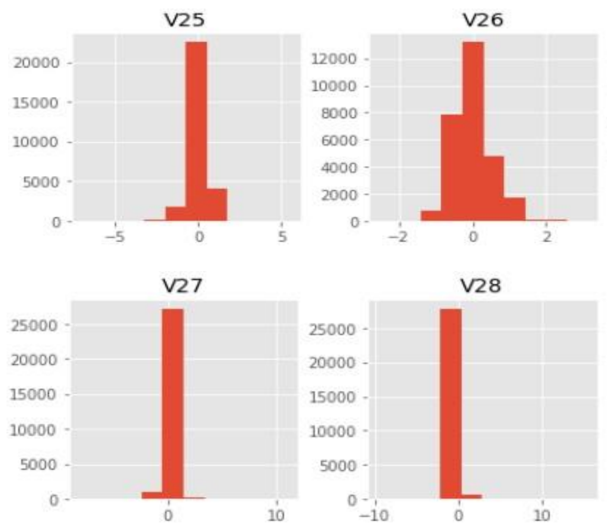


Fig. 16 : Histograms of induvidual parameter

Fig. 16 show the count , mean , std , minimum , maximum for each parameter as above

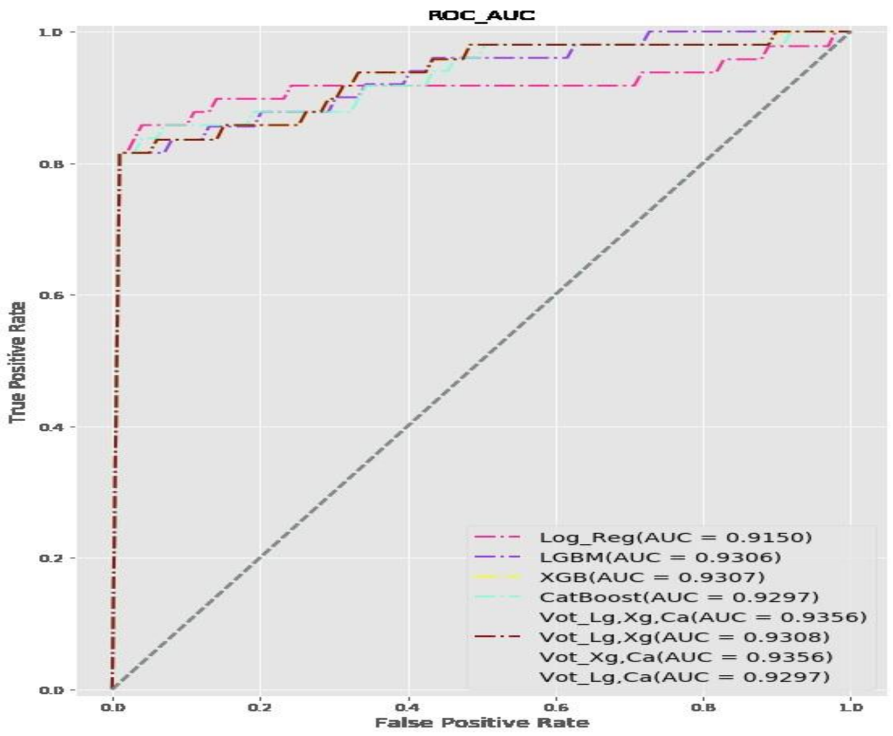


Fig. 17 : Plot for the ROC(Receiver Operating Characteristic Curve) and AUC(Area Under the Curve)

The graph shows the curve which provides the distinguish between TP rate and Fp rate for each model

V. Conclusion and future scope

The Stacking Classifier stood out because it had the best accuracy of all the models, showing how well it could find fraud. The project showed strong results in many machine learning models, such as LightGBM, XGBoost, CatBoost [29, 30, 31, 32], vote classifiers, and neural networks, showing how flexible it is. Using a variety of sampling and scaling methods greatly improved the accuracy of scam detection, highlighting how important these methods are. Using the Stacking Classifier ensemble method greatly improved the efficiency of scam identification,highlighting how well it works. Making a Flask front-end that is easy for people to use speeds up user testing and identification, making sure that the system is accessible and useful. Testing the system in Flask, where feedback was given, confirms that it works and is easy to use. [1, 2, 3] Findings from the project show that advanced machine learning methods can help find scams in the banking industry. This opens the door for more uses in the future. The project's results open up chances to keep getting better by looking into more group methods and optimization strategies. In the end, the project's results help the banking industry by making it easier to spot scams, cutting down on financial losses, and making sure that deals are safe, which boosts trust and security overall.

In the future, researchers will look into how to improve the accuracy and reliability of scam identification by adding more mixed models with CatBoost [29]. CatBoost's hyperparameters will be fine-tuned in the future, with a focus on finding the best number of trees to make the model work better [33]. Researchers will be looking into ways to make the model work even when fraud trends change, so it can keep finding new scam activities. Real-time data will make systems more sensitive and flexible, which will allow faster reactions to new threats. This is the goal of ongoing study. In the future, work will be done to make the model's decision-making process easier to understand. This will give us a better idea of how it builds trust and improves scam detection methods.

REFERENCES

[1] J. Nanduri, Y.-W. Liu, K. Yang, and Y. Jia, ‘‘Ecommerce fraud detection through fraud islands and multi-layer machine learning model,’’ in Proc. Future Inf. Commun. Conf., in Advances in Information and Communication. San Francisco, CA, USA: Springer, 2020, pp. 556–570.

[2] I. Matloob, S. A. Khan, R. Rukaiya, M. A. K. Khattak, and A. Munir, ‘‘A sequence mining-based novel architecture for detecting fraudulent transactions in healthcare systems,’’ IEEE Access, vol. 10, pp. 48447–48463, 2022.

[3] H. Feng, ‘‘Ensemble learning in credit card fraud detection using boosting methods,’’ in Proc. 2nd Int. Conf. Comput. Data Sci. (CDS), Jan. 2021, pp. 7–11.

[4] M. S. Delgosha, N. Hajiheydari, and S. M. Fahimi, ‘‘Elucidation of big data analytics in banking: A four-stage delphi study,’’ J. Enterprise Inf. Manage., vol. 34, no. 6, pp. 1577–1596, Nov. 2021.

[5] M. Puh and L. Brki¢, ‘‘Detecting credit card fraud using selected machine learning algorithms,’’ in Proc. 42nd Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO), May 2019, pp. 1250–1255.

[6] K. Randhawa, C. K. Loo, M. Seera, C. P. Lim, and A. K. Nandi, ‘‘Credit card fraud detection using AdaBoost and majority voting,’’ IEEE Access, vol. 6, pp. 14277–14284, 2018.

[7] N. Kumaraswamy, M. K. Markey, T. Ekin, J. C. Barner, and K. Rascati, ‘‘Healthcare fraud data mining methods: A look back and look ahead,’’ Perspectives Health Inf. Manag., vol. 19, no. 1, p. 1, 2022.

[8] E. F. Malik, K. W. Khaw, B. Belaton, W. P. Wong, and X. Chew, ‘‘Credit card fraud detection using a new hybrid machine learning architecture,’’ Mathematics, vol. 10, no. 9, p. 1480, Apr. 2022.

[9] K. Gupta, K. Singh, G. V. Singh, M. Hassan, G. Himani, and U. Sharma, ‘‘Machine learning based credit card fraud detection—A review,’’ in Proc. Int. Conf. Appl. Artif. Intell. Comput. (ICAAIC), 2022, pp. 362–368.

[10] R. Almutairi, A. Godavarthi, A. R. Kotha, and E. Ceesay, ‘‘Analyzing credit card fraud detection based on machine learning models,’’ in Proc. IEEE Int. IoT, Electron. Mechatronics Conf. (IEMTRONICS), Jun. 2022, pp. 1–8.

[11] N. S. Halvaiee and M. K. Akbari, ‘‘A novel model for credit card fraud detection using artificial immune systems,’’ Appl. Soft Comput., vol. 24, pp. 40–49, Nov. 2014.

[12] A. C. Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten, ‘‘Feature engineering strategies for credit card fraud detection,’’ Expert Syst. Appl., vol. 51, pp. 134–142, Jun. 2016.

[13] U. Porwal and S. Mukund, ‘‘Credit card fraud detection in e-commerce: An outlier detection approach,’’ 2018, arXiv:1811.02196.

[14] H. Wang, P. Zhu, X. Zou, and S. Qin, ‘‘An ensemble learning framework for credit card fraud detection based on training set partitioning and clustering,’’ in Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Oct. 2018, pp. 94–98.

[15] F. Itoo, M. Meenakshi, and S. Singh, ‘‘Comparison and analysis of logistic regression, Naïve Bayes and knn machine learning algorithms for credit card fraud detection,’’ Int. J. Inf. Technol., vol. 13, no. 4, pp. 1503–1511, 2021.

[16] T. A. Olowookere and O. S. Adewale, ‘‘A framework for detecting credit card fraud with cost-sensitive meta-learning ensemble approach,’’ Sci. Afr., vol. 8, Jul. 2020, Art. no. e00464.

[17] A. A. Taha and S. J. Malebary, ‘‘An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine,’’ IEEE Access, vol. 8, pp. 25579–25587, 2020.

[18] X. Kewei, B. Peng, Y. Jiang, and T. Lu, ‘‘A hybrid deep learning model for online fraud detection,’’ in Proc. IEEE Int. Conf. Consum. Electron. Comput. Eng. (ICCECE), Jan. 2021, pp. 431–434.

[19] T. Vairam, S. Sarathambekai, S. Bhavadharani, A. K. Dharshini, N. N. Sri, and T. Sen, ‘‘Evaluation of Naïve Bayes and voting classifier algorithm for credit card fraud detection,’’ in Proc. 8th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS), Mar. 2022, pp. 602–608.

[20] P. Verma and P. Tyagi, ‘‘Analysis of supervised machine learning algorithms in the context of fraud detection,’’ ECS Trans., vol. 107, no. 1, p. 7189, 2022.

[21] J. Zou, J. Zhang, and P. Jiang, ‘‘Credit card fraud detection using autoencoder neural network,’’ 2019, arXiv:1908.11553.

[22] D. Almhaithawi, A. Jafar, and M. Aljnidi, ‘‘Example-dependent costsensitive credit cards fraud detection using SMOTE and Bayes minimum risk,’’ Social Netw. Appl. Sci., vol. 2, no. 9, pp. 1–12, Sep. 2020.

[23] J. Cui, C. Yan, and C. Wang, ‘‘Learning transaction cohesiveness for online payment fraud detection,’’ in Proc. 2nd Int. Conf. Comput. Data Sci., Jan. 2021, pp. 1–5.

[24] M. Rakhshaninejad, M. Fathian, B. Amiri, and N. Yazdanjue, ‘‘An ensemble-based credit card fraud detection algorithm using an efficient voting strategy,’’ Comput. J., vol. 65, no. 8, pp. 1998–2015, Aug. 2022.

[25] A. H. Victoria and G. Maragatham, ‘‘Automatic tuning of hyperparameters using Bayesian optimization,’’ Evolving Syst., vol. 12, no. 1, pp. 217–223, Mar. 2021.

[26] H. Cho, Y. Kim, E. Lee, D. Choi, Y. Lee, and W. Rhee, ‘‘Basic enhancement strategies when using Bayesian optimization for hyperparameter tuning of deep neural networks,’’ IEEE Access, vol. 8, pp. 52588–52608, 2020.

[27] F. N. Khan, A. H. Khan, and L. Israt, ‘‘Credit card fraud prediction and classification using deep neural network and ensemble learning,’’ in Proc. IEEE Region 10 Symp. (TENSYMP), Jun. 2020, pp. 114–119.

[28] W. Liang, S. Luo, G. Zhao, and H. Wu, ‘‘Predicting hard rock pillar stability using GBDT, XGBoost, and LightGBM algorithms,’’ Mathematics, vol. 8, no. 5, p. 765, May 2020.

[29] S. B. Jabeur, C. Gharib, S. Mefteh-Wali, and W. B. Arfi, ‘‘CatBoost model and artificial intelligence techniques for corporate failure prediction,’’ Technol. Forecasting Social Change, vol. 166, May 2021, Art. no. 120658.

[30] J. Hancock and T. M. Khoshgoftaar, ‘‘Medicare fraud detection using CatBoost,’’ in Proc. IEEE 21st Int. Conf. Inf. Reuse Integr. Data Sci. (IRI), Aug. 2020, pp. 97–103.B.

[31] Dhananjay and J. Sivaraman, ‘‘Analysis and classification of heart rate using CatBoost feature ranking model,’’ Biomed. Signal Process. Control, vol. 68, Jul. 2021, Art. no. 102610.

[32] Y. Chen and X. Han, ‘‘CatBoost for fraud detection in financial transactions,’’ in Proc. IEEE Int. Conf. Consum. Electron. Comput. Eng. (ICCECE), Jan. 2021, pp. 176–179.

[33] A. Goyal and J. Khiari, ‘‘Diversity-aware weighted majority vote classifier for imbalanced data,’’ in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2020, pp. 1–8.

[34] A. Roy, J. Sun, R. Mahoney, L. Alonzi, S. Adams, and P. Beling, ‘‘Deep learning detecting fraud in credit card transactions,’’ in Proc. Syst. Inf. Eng. Design Symp. (SIEDS), Apr. 2018, pp. 129–134.