IDENTIFICATION OF FRAUDULENT TRANSACTIONS ON ETHEREUM BLOCKCHAIN

Muthyala Sai Teja[1], Dasari Aditya Vardhan[2]

Under the Guidance of

Ms.Nemmani Swapna

Associate Professor, ECM

Sreenidhi Institute of Science and Technology (SNIST)

***

Abstract: Decentralization, transparency, and immutability are the three pillars on which the blockchain's digital system is built. Two well-known Blockchain platforms, Bitcoin and Ethereum, allow users to deal anonymously online while being globally connected peer-to-peer. The enormous volume of decentralized transactions and the participants' apparent anonymity make it possible for scams, cyber frauds, hacks, money laundering, and fraudulent transactions. Since standard auditing approaches require more processing power, time, and memory for complicated queries to join combinations of data, it is difficult to identify such fraudulent acts using them. Utilizing machine learning algorithms is one strategy to get rid of fraud. Machine learning can be either supervised or unsupervised in nature. Several machine learning strategies have been presented in earlier works to address this issue, and while some of the results seem extremely promising, there is no one approach that stands out as being clearly superior. In those studies, the effectiveness of several supervised machine learning models, including SVM, Decision Tree, Naive Bayes, Logistic Regression, and a few deep learning models, was compared in order to identify fraudulent transactions in a blockchain network.We notice that these models have limitations such as scalability and accuracy.

We propose a new Machine Learning method i.e, XGB classifier to handle the above problem in addition to existing methods. Where we discuss the limitations like scalability, accuracy and handling of missing values. We conclude by highlighting the advantages of our proposed system in comparison to that of previous existing methods.

Index Terms: Blockchain, Machine Learning, Fraudulent Transactions, Random Forest, Support Vector Machine (SVM) , Gradient Boosting, XGBoost.

1.INTRODUCTION

Blockchain technology is a distributed ledger that is immutable and shared by hundreds of computer systems [1]. After Satoshi Nakamoto's [2] white paper was published in 2008, blockchain technology gained unprecedented attention. In his white paper, Satoshi proposed a decentralized P2P network to address the double-spending issue for digital currency [3]. Dubai appointed an artificial intelligence minister with the goal of becoming the world's first blockchain-powered government[4]. One of the Middle Eastern nations, Saudi Arabia, recently declared that it will employ blockchain technology [5] [17] to credit some of the liquidity that would be injected into the banking system. A trustworthy third party is being replaced by distributed consensus building in decentralized P2P networks [6].

1.1 Background

The most popular Turing Complete [5] [17] blockchain platform, Ethereum, enables programmers to create smart contracts with entirely customized ownership, transactional, and state transition rules. By operating an Ethereum node on their computer, anyone can take part in the Ethereum blockchain network [6] [7]. The layered design of the Ethereum blockchain is depicted in Figure 1.

Ethereum is the most well-known platform for peer-to-peer programming and a well-known cryptocurrency exchange platform.A lot of attention is now being paid to blockchain security and oversight [8] [9].Implementing smart contracts is possible thanks to Ethereum, an open-source blockchain technology. Its development addresses the issue of the Bitcoin protocol's constrained scalability. Because of the versatility of the Solidity language, developers can build general-purpose smart contracts [10] [11]. Among the numerous smart contracts, there can be a few that steal ether from network users. Ethereum has expanded quickly due to its flexibility and simplicity. Additionally, Ethereum has overtaken Bitcoin to become the second-most valuable cryptocurrency. One such cryptocurrency is Ethereum, which was developed by Vitalik Buterin.For those who are new, Ethereum is a cryptocurrency distribution system that allows you to send cryptocurrency to any person for a small price.

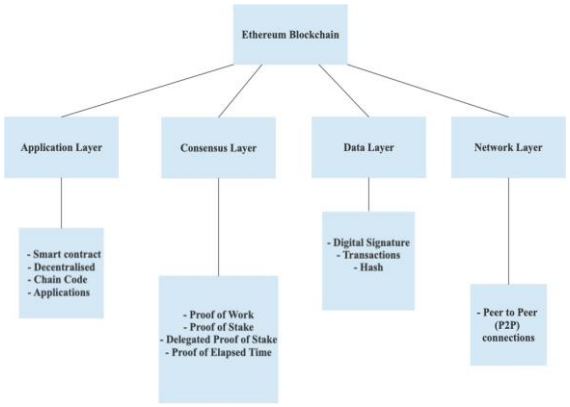

Figure 1.1 : Layered Structure of Ethereum Blockchain

1.2 Problem Definition

Cryptocurrency platforms have grown in popularity due to their ability to conduct secure and rapid banking transactions. Transactions on the blockchain network are not restricted to simply sending or receiving payments; smart contracts and cryptocurrency trading are also very frequent. With over one million transactions per day, Ethereum is the second-largest cryptocurrency network. However, Ethereum, like other cryptocurrency platforms, has its fair share of cyber fraud.

The transaction mechanism in Blockchain is extremely strong and secure. A consensus algorithm is run by all of the entities in the Blockchain network whenever the latest transaction is submitted by one of them. When the consensus algorithm determines that the transaction is valid, a block is added to the distributed ledger in encrypted form. Despite this internal security, Ethereum is used for a significant amount of fraud, money laundering, Ponzi schemes, bribery, and phishing. Given how regularly scammers target clients, this is a big problem. Certainly, a technique is required to screen out unauthorized accounts using Ethereum. However, aside from centralized databases like Cryptoscamdb that record fraudulent accounts, Ethereum does not provide a concrete solution to keep these activities in check. On Cryptoscamdb, there are currently over 6000 illicit accounts. This wealth of account data can be used to train supervised machine learning models. The model thus created can be used to detect suspicious internet transactions in the background. Therefore, it is clear that a system to identify illegal activity on Ethereum is needed.

1.3 Motivation

The following is a summary of the motivation for this work:

- Ethereum is a popular cryptocurrency, and the Blockchain platform can handle millions of transactions every second. The built-in security mechanism ensures transaction security and privacy. However, Many attacks and illegal activities are still reported on the Ethereum Blockchain. If there is a mechanism in place to identify fake transactions online, it can be used to identify the accounts.
- The Cryptoscamdb platform offers access to the transactional history of fraudulent transactions. This can be used to generate a dataset for training machine learning models to detect fake transactions.

2. LITERATURE SURVEY

Using a variety of classifiers and methodologies, several studies are being conducted to identify Ethereum scams. Here are a few of these in more detail:

Ajay & Kumar(2020) et al. [12] used machine learning methods to detect anomalies in Ethereum networks and determine whether users are suspicious. They recommended implementing secure smart contracts created using the ERC20 interface on a blockchain network equipped with the required features and protocols, in order to provide a thorough framework for protecting manufacturing processes based on the Cloud. Finding characteristics that enable the exact detection of anomalous contracts is highly challenging, and statistical analysis based on these characteristics is similarly ineffective. In addition, they fail to take into account the internal consistency and instability of smart contract accounts, which significantly increases model obversion.

To find the odd Ethereum smart contracts, Teng(2020) et al. [13] employed a pattern-based methodology. In order to train the model, they used both data slicing and LSTM, or long-term short-term memory.The results showed that contracts could be identified with great accuracy.

A machine learning model for phishing scam detection on the Ethereum market was developed by Yuan et al. [14] in 2020. They first created the transaction network model to foresee the fraudulent transactions using the transaction data. The SVM classifier and the node2vec embedding approach were used to train the next model. According to the findings, non-embedding methods are poor at extracting characteristics. The network embedding technique is therefore essential for obtaining transaction network characteristics. According to reports, Ethereum scams earn significant profits and pose a serious danger to the financial security of the Ethereum network.

In 2021, With the aid of supervised learning as well as ensemble learning techniques like random forests and decision trees, Rahmeh et al. [15] proposed a methodology to detect illicit transactions. Building a dataset with six features

using the correlation coefficient allowed them to train the model and forecast its accuracy. When these methods are used, the results show a significant improvement in time measurements, as well as an enhancement in the F measure when employing the random forest approach. To guarantee a secure investing environment, they came to the conclusion that an efficient fraud detection system is vital.

In 2022, An in-depth examination of Ethereum smart contract security flaws, including well-known security attacks and associated protective measures, was provided by Satpal et al. [17]. They meticulously covered the well-known hits and their recognition and evaluation tools before discussing the vulnerabilities in terms of their underlying causes and secondary causes. Tools for analyzing Ethereum smart contracts are highlighted based on five criteria, including the tool's nature, input, form of analysis, execution language, and accessibility. It has been discovered that Ethereum smart contracts are subject to adversarial attack due to security weaknesses. Because smart contracts deal with digital assets, a single mistake or security issue might result in a loss of millions of dollars.

Dataset

This dataset [18] comprises rows that contain recognised fraudulent and valid Ethereum transactions. Following Bitcoin, Ethereum (ETH) is the second most popular cryptocurrency. Ethereum, founded in 2015 by Vitalik Buterin and Gavin Wood, now accounts for more than 17% of the $1.2 trillion global crypto market.

There are some notable distinctions between Ethereum and the original cryptocurrency. Unlike Bitcoin (BTC), Ethereum is intended to be a whole lot more than merely a means of exchange or the storage of funds. Instead, Ethereum is a blockchain-based decentralized computing network. This dataset contains 9841 rows of recorded transactions categorized based on attributes in 48 columns.

Following is a list some dataset's columns:

- Index: the row's index number
- FLAG: determines if the transaction was fraudulent
- Avg min between sent tnx: Minutes between each transaction on average for the account
- Address: the ethereum account's address
- Received_tnx: Sum of the number of all received regular transactions
- Time Diff between first and_last (Mins): Separation in time between the initial and final transaction
- Avg min between received tnx: In minutes, the average amount of time between transactions received for the account
- Sent_tnx: Sum of all transactions that were sent normally
- NumberofCreated_Contracts: Total number of newly generated transactions
- Unique SentTo_Addresses20: Sum of the number of all unique addresses from which transactions were sent by each account
- MinValueReceived: lowest ever received value in ether

Is ERC20 the same as ETH?

What exactly is the distinction between ETH and ERC20?

ETH, or "Ether," is the native cryptocurrency of the Ethereum network, and it is employed for Ethereum blockchain transactions.

The Ethereum Request for Comment 20 (ERC-20) standard is the implemented standard for fungible tokens issued on the Ethereum blockchain. ERC-20 facilitates the creation of new Ethereum blockchain tokens that are interchangeable with other smart contract tokens. The ERC20 protocol is the industry standard for developing Ethereum-based tokens that may be used and deployed on the Ethereum network.

3. EXISTING SYSTEM

According to the project's foundational publication, "Comparative Study of Machine Learning Algorithms for Fraud Detection in Blockchain" [16], the workflow of the system is as follows:

Once the Blockchain network has authorized a transaction after completing basic checks, the system takes over and performs extra tests to identify if the transaction in question is potentially fraudulent. With this approach, extra checks on transactions are no longer necessary as the Blockchain platform itself can invalidate them. The block diagram of the existing system is shown in Figure 2.

There are three stages of the work that can be distinguished:

Phase 1: Preprocessing

Phase 2 : Creating as well as refining a variety of models, such as Logistic Regression, SVM, Decision Trees, and Näive Bayes.

Phase 3 : A performance assessment of each model.

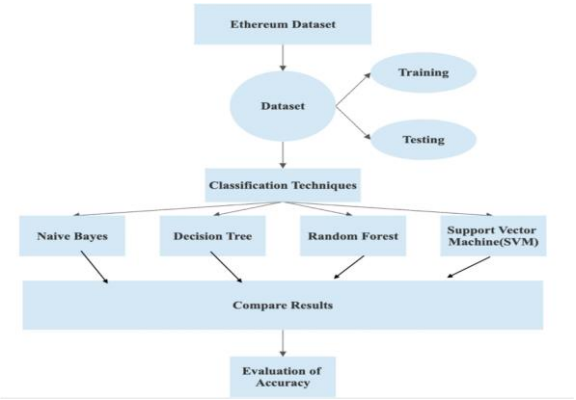

Figure 3.1 : Block Diagram of Existing System

3.1 Algorithms Used

1. Naïve Bayes

Using the "naive" premise that every pair of features is conditionally independent given the value of the class variable, a collection of supervised learning techniques known as naive Bayes methods are based on Bayes' theorem. The following relationship is stated by Bayes' theorem, given class variable y and dependent feature vectors x1 through xn, :

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}$$

In comparison to more complex techniques, naive Bayes classifiers and learners can operate at lightning speeds. Because the class conditional feature distributions are decoupled, each distribution can be approximated as a one-dimensional distribution. This then helps to solve problems brought on by the dimensionality curse [19].

However, while naive Bayes is a decent classifier, it is a poor estimator, therefore the likelihood outputs from predict_proba shouldn't be taken too seriously.

2. Support Vector Machine (SVM)

Support vector machines i.e. SVM are robust and adaptable supervised machine learning techniques which are used for outlier detection, classification and regression. SVMs are frequently employed in classification issues and are extremely effective in large dimensional spaces. Because the decision function only uses a small chunk of the training points in the decision function, SVMs are well-liked and memory-efficient algorithms [20].

The fundamental purpose of SVM is to partition datasets into different classes with the goal to discover a maximum marginal hyperplane (MMH), which can be accomplished in these two steps:

- Support Vector Machines will initially construct iterative hyperplanes that best distinguish the classes.
- The hyperplane that best separates the classes will then be chosen.

The following are some key SVM concepts: −

- Support Vectors − The data points that are closest to the hyperplane are known as support vectors. Support vectors aid in establishing the dividing line.
- Margin − Margin refers to the space between two lines on a graph of data points from various classes.
- Hyperplane − The division between a group of objects with various class identities in a decision plane or space.

The graph diagram below will help you understand these SVM concepts −

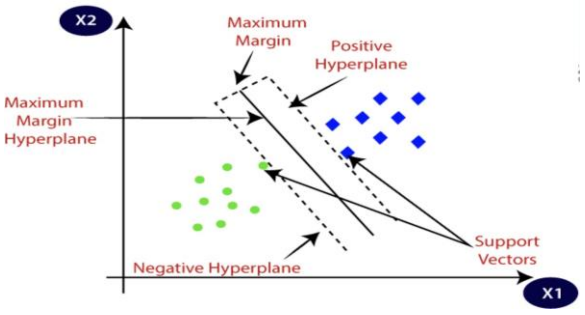

Figure 3.2 : SVM Classifier Graph

3. Decision Tree

A decision tree is a hierarchical decision support model that represents options and potential outcomes in the form of trees, including utility, resource costs, and chance event outcomes. It is one method of displaying an algorithm that only involves conditional control statements.

Decision trees are often used in operations research, notably decision analysis [21], to assist determine the method most likely to achieve a goal, but they are also a prominent technique in machine learning.A decision tree and the closely related impact diagram are used as a visual and analytical

decision support tool in decision analysis to calculate the expected values (or expected utility) of competing alternatives.

Three different node types make up a decision tree:[22]

1. Decision nodes – Squares are commonly used to symbolize decision nodes.
2. Chance nodes – usually depicted as circles
3. End nodes – End nodes are often symbolized as triangles.

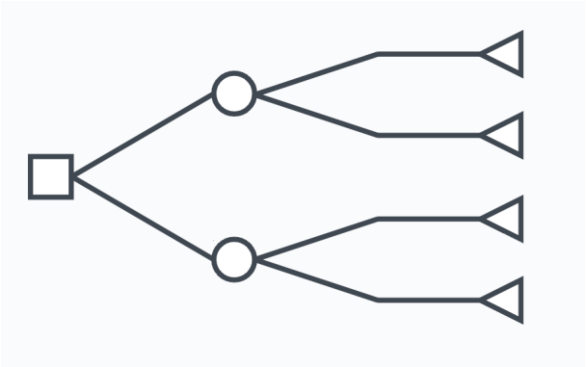

Figure 3.3 : Decision Tree Flowchart

Decision trees have the following advantages:

- Easy to comprehend and interpret. One can visualize trees.
- Little data preparation is necessary. Other procedures frequently need data normalization, the creation of dummy variables, and the removal of blank values. However, keep in mind that this module does not handle missing values.
- The cost of employing the tree (i.e., making predictions about the data) increases logarithmically with the quantity of data points needed to train the tree.
- Capable of handling numerical as well as categorical data. However, categorical variables are currently not supported by the scikit-learn implementation.
- It is possible to use statistical tests to verify a model. This enables the model's dependability to be taken into account.
- Performs well even if the underlying model from which the data were created slightly violates some of its basic assumptions.

The following are some downsides of decision trees:

- Decision-tree learners can produce overly complicated trees that do not generalize the data properly. This is referred to as overfitting. To prevent this problem, mechanisms like pruning, establishing the minimum amount of samples required at a leaf node, and setting the maximum depth of the tree are required.
- Decision trees are susceptible to instability since even minor changes in the data could produce an entirely different tree. An ensemble of decision trees is used to solve this problem.
- Decision tree predictions are not smooth or continuous, but rather piecewise constant approximations, as shown in the picture above. As a result, they are not adept at extrapolation.
- Certain concepts, like XOR, parity, or multiplexer problems, are challenging to understand because decision trees do not easily express them.
- If some classes dominate, the choices made by decision tree learners will yield biased trees. As a result, before fitting the dataset to the decision tree, it is recommended that it be balanced.

4. Logistic Regression

The logistic model, also referred to as the logit model, is a model of statistics that determines the likelihood that an event will occur by converting the event's log-odds into a linear combination of any number of independent variables. In regression analysis, logistic regression (also known as logit regression)[23] calculates a logistic model's parameters (the coefficients in the linear combination). In the case of binary logistic regression, there is only one binary dependent variable, which is coded by an indicator variable and has two values labeled "0" and "1," whereas the independent variables can be either binary variables (which have two classes, each coded by an indicator variable), or continuous variables (which can have any real value).

The value labeled "1" has a probability that can range from 0 (definitely the value "0") to 1 (definitely the value "1"), therefore the labels;[24] the logistic function is the function that transforms logarithmic-odds to probability, hence the name. The alternate names are derived from the logit, or logistic unit, which is the unit of measurement for the log-odds scale.

$$\text{Logit}(pi) = 1/(1+ \exp(-pi))$$

$$\ln(pi/(1-pi)) = \text{Beta\_0} + \text{Beta\_1}*X\_1 + \ldots + B\_k*K\_k$$

There are three different kinds of categorical response-based logistic regression models.

- *Binary logistic regression*: Here the response or dependent variable is dichotomous, meaning that there are only two possible outcomes (for example, 0 or 1). It is frequently used to determine if an email is spam or not, as well as whether a tumor is malignant or not.
- *Multinomial logistic regression*: A kind of logistic regression model where the dependent variable contains three or more potential values, but there is no predetermined order in which they should be ranked. For instance, in order to more successfully sell their films, movie companies aim to forecast the type of film a viewer will likely watch.
- *Binary logistic regression*: In a case where response variable includes three or more possible outcomes, however in this situation, these values do have a fixed order, the ordinal logistic regression model is used. Grading scales from A to F or rating scales from 1 to 5 are two examples of ordinal replies.

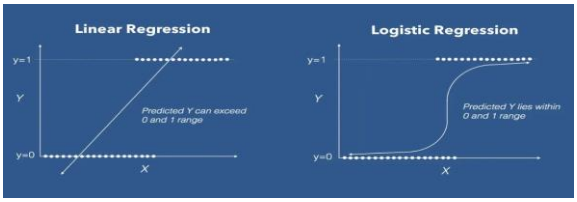

Figure 3.4 : Linear Regression Vs Logistic Regression

5. Random Forest Classifier

The widely used machine learning technique known as random forest, which mixes the output of several decision trees to obtain a single outcome, was developed by Leo Breiman and Adele Cutler. Its adaptability and usefulness, which it uses to address classification and regression problems, contribute to its widespread use.

As an ensemble learning technique for classification, regression, and other tasks, random forests—also referred to as random choice forests—work by building a lot of decision trees during the training phase. The output of a random forest is the categorization that the majority of trees choose for classification problems. Regression tasks return the mean or average forecast of each individual tree [25] [26]. The tendency of decision trees to overfit their training set is corrected by random decision forests.Although they frequently outperform decision trees, gradient boosted trees are more accurate than random forests.Their effectiveness, however, can be impacted by data characteristics [27].

N decision trees are combined to construct the random forest in the first phase of the Random Forest's operation, and then predictions are made for each tree that was produced in the first phase.

The steps and diagram below can be used to explain the working process:

1: Pick K data points randomly from the training set.

2: Create the decision trees linked to the subsets of data that have been chosen.

3: Select N for the size of the decision trees that you want to construct.

4 :Repeat steps 1 and 2.

5: By searching up each decision tree's predictions for the new data points, assign new data points to the category with the most votes.

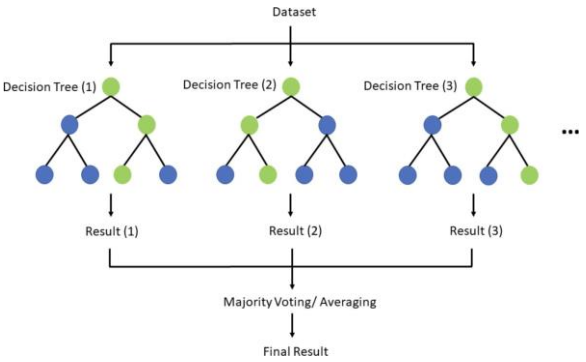

Figure 3.5 : Random Forest

4. PROPOSED SYSTEM

The majority of prior work for detecting fraudulent transactions, according to the literature review, involved machine learning techniques and optimization techniques, such as Support Vector Machine (SVM), Logistic Regression, Decision Tree, etc. The gradient boosting approach, which has the potential to produce results very quickly and accurately, has, however, received very little research.

As a result, the proposed study adopts an improved machine learning technique that employs superior gradient enhancement/boosting algorithms that detect fraudulent transactions with a high degree of accuracy while avoiding overfitting.

4.1 Algorithms Used

1.Gradient boosting:

Gradient boosting is a machine learning approach that is utilized in regression and classification applications, among other things. It offers a prediction model in the form of a collection of decision trees or other weak prediction models.[28][29] The resulting algorithm, known as gradient-boosted trees, typically outperforms random forest when a decision tree is the weak learner. [30] The development of a gradient-boosted trees model follows the same stage-wise process as earlier boosting techniques, but it generalizes those techniques by enabling the optimization of any differentiable loss function.

Before delving into the specifics of this algorithm, it is important to understand the AdaBoost Algorithm, another boosting technique. This algorithm begins by creating a decision stump, after which all the data points are given equal weights. The weights for all the incorrectly classified points are then increased, while those that are simple to classify or are correctly classified have their weights decreased. p. The primary distinction between these two techniques is that Gradient boosting uses Decision Trees as its fixed base estimator, whereas AdaBoost allows us to customize the base estimator to meet our specific needs.

Boosting is a subset of gradient boosting. The underlying assumption is that the best subsequent model, when combined with previous models, significantly reduces overall prediction error. To prevent errors, the main idea is to specify the outcomes you want from this subsequent model.

1. XGBoost Classifier

XGBoost is an optimized Gradient Boosting Machine Learning package. It was originally created in C++, but it includes APIs in various other languages. The fundamental XGBoost algorithm can be parallelized, which means it can undertake parallelization within a single tree.

The following are some advantages of using XGBoost:

- It is one of the most powerful algorithms in terms of speed and performance.
- It can take advantage of all of the computing power available in modern multicore systems.
- It is possible to train on big datasets.
- Consistently outperforms all approaches using a single algorithm.

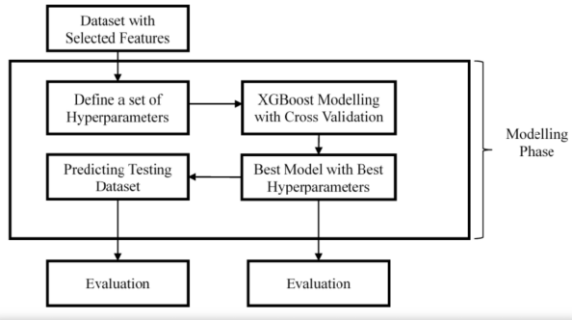

Figure 4.1 : Block Diagram of Proposed system

XGBoost is typically used in conjunction with a tree as the base learner; this decision tree is made up of a series of binary questions, and the leaf is where the final predictions are made. As a standalone ensemble method, XGBoost. Iteratively building the trees until a halting requirement is satisfied[31].

XGBoost is a tree ensemble-based boosting algorithm created in 2016 by the University of Washington. XGBoost considers an ensemble of classifiers in the form of decision trees. In the instance of XGBoost, boosting uses gradient-based optimisation to minimize classifier mistakes. Only classifiers that make mistakes and are weak are optimized. After the weak classifiers have been transformed into strong classifiers through optimization, bagging is performed, and the classification label is generated by taking a vote from all of the classifiers.

In addition to increasing the model's accuracy, XGBoost also makes the process quick by building tree ensembles in parallel. The algorithm works well for optimizing hardware. This algorithm also reduces computations and enhances performance by pruning the decision trees. Additionally, it supports cross-validation and regularization, two advanced features for algorithms [32].

CART (Classification and Regression Trees) Decision Trees are used by XGBoost. CART trees have real-valued scores in each leaf, regardless of if they are being utilized for classification or regression. Real-valued ratings can then be translated to categories for classification if necessary.

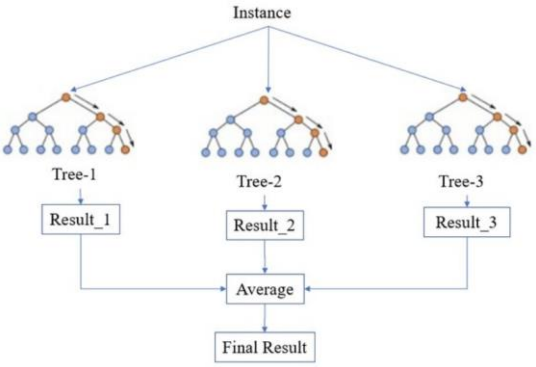

Figure 4.2 : Simplified Structure of XGBoost

The following are the steps involved in developing the proposed system:

1. Data collection
2. Data exploration
3. Data cleaning and preprocessing
4. Selecting machine learning models
5. Training the model using data
6. Data Evaluation/Testing
7. Hyperparameter Tuning
8. Display results.

Here we define some of the steps in detail:

In the first step, data is collected from the cryptoscamdb and the kaggle dataset.

In the second step, we start the process by importing the required packages and loading the dataset. At first let's use automl solutions to see what is going on. We chose Pycaret and the compare_models( ) function to automatically compare the various machine learning models.We will try to reach those evaluation metrics by doing some EDA (Exploratory data Analysis). After EDA it is found that we are dealing with a highly imbalanced data of 22% fraudulent instances.

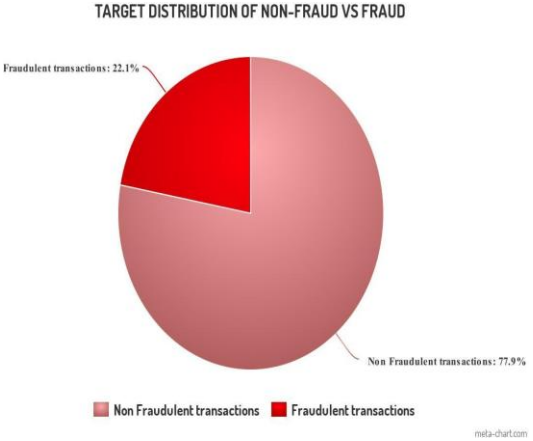

Figure 4.3 : Target Distribution of Non-Fraud Vs Fraud transactions

In the third step , we start data cleaning by dropping unnecessary variables and features with zero variance, since these features will not help in the performance of the model.

Now we display these features using a heatmap that masks zero variance features.

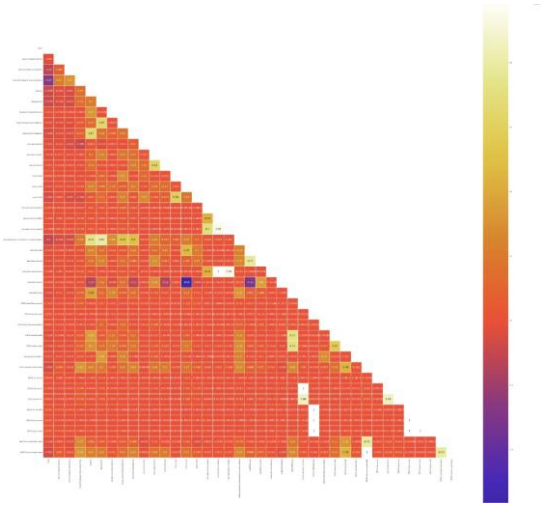

Figure 4.4 : Heatmap masking features with zero variance

Now we sort those features using the sorted_corr method and drop some of the highly correlated features for a clear understanding of data. We recheck the correlation matrix and draw a new heatmap

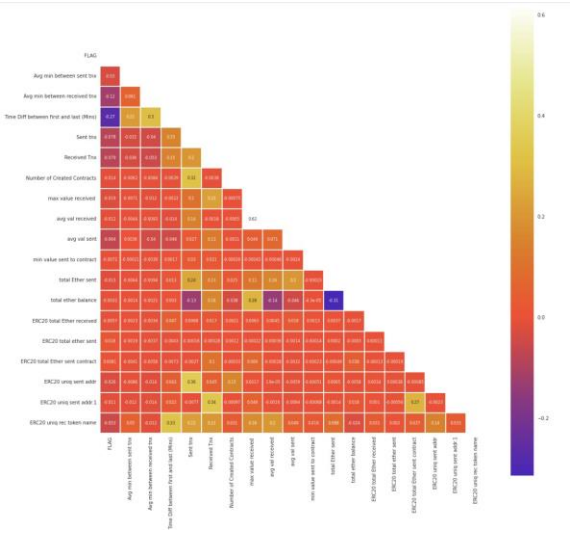

Figure 4.5 : New heatmap after dropping highly correlated features

Now drop the features that are mostly zeroes.

The fourth step includes selecting a machine learning model which will be done later in the process.

In the fifth step, let's focus on preparing data for training. We begin by splitting the data into training data (80%) and testing data (20%).

Use StandardScaler( ) function to transform X_train data.

Let's use SMOTE for OverSampling.

SMOTE stands for Synthetic Minority Oversampling Technique. This approach was first proposed in the Journal of Artificial Intelligence Research in 2002. SMOTE is a better approach for resolving classification issues involving unbalanced data.

Outputs after oversampling the training data:

Shape of the training before SMOTE: ((7872, 16), (7872,))
Shape of the training after SMOTE: ((12230, 16), (12230,))
BEFORE OVERSAMPLING
Non-frauds: 6115
Frauds: 1757
AFTER OVERSAMPLING
Non-frauds: 6115
Frauds: 6116

Now it's time to model our data using below code:

- Logistic Regression:

```
LR = LogisticRegression(random_state=42)
LR.fit(x_tr_resample, y_tr_resample)
```

- Random Forest Classifier

```
RF = RandomForestClassifier(random_state=42)
RF.fit(x_tr_resample, y_tr_resample)
```

- XGB Classifier

```
xgb_c = xgb.XGBClassifier(random_state=42)
xgb_c.fit(x_tr_resample, y_tr_resample)
```

5. SYSTEM DESIGN

5.1 System Architecture

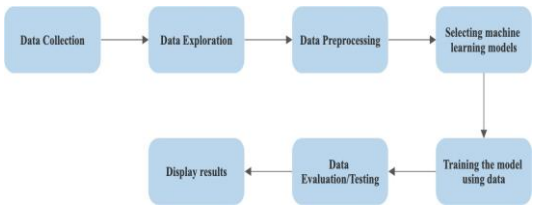

Figure 5.1 : System architecture

5.2 Data Flow Diagram

A data flow diagram (DFD) depicts the flow of data through a system or process. It uses preset symbols like rectangles, circles, and arrows as well as short text labels to indicate data inputs, outputs, storage places, and paths between each destination. Data flow diagrams can range from simple, one-level DFDs that only briefly summarize the data handling process to more intricate, multi-level DFDs that go in-depth at each level. They can be applied to analyze a current system or model a new one.

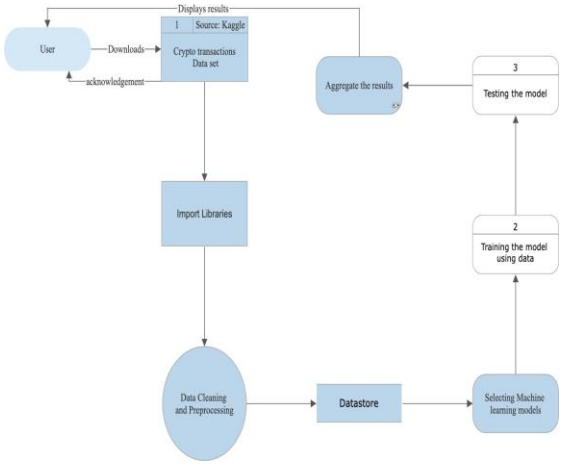

Figure 5.2 : Data Flow Diagram

- We can see how the model created to identify fraudulent transactions on the Ethereum blockchain works in the above figure.
- The process starts with giving the input which helps in the retrieval of the transactions from the dataset.
- The dataset's transactions are first processed before being classified based on features.
- A training set makes up 80% of the dataset, while the testing set makes up 20%.
- The data is sent for evaluation, where we assess the transactions using a variety of machine learning models, including naive bayes, random forest, SVM, logistic regression, decision trees, and a number of gradient boosting algorithms, including AdaBoost and XGBoost.
- Following testing, it is possible to compare the outcomes of the different machine learning models and select the most effective model.

5.3 UML Diagrams

The acronym UML stands for Unified Modelling Language. UML is a general-purpose modeling language that is standardized in the field of object-oriented software engineering. The Object Management Group is responsible for managing and developing the standard.

UML's objective is to become a standard language for creating object-oriented computer software models. UML consists of two key components in its current form: a

meta-model and a notation. Future developments in UML-related methods and processes are possible.

Business modeling, non-software systems, and software system artifacts can all be described, visualized, built, and documented using the Unified Modelling Language (UML), which is a standard language.

The UML is a collection of best engineering practices that have been effective in simulating huge, complicated systems.

The software development process and the creation of object-oriented software both heavily rely on the UML. The UML primarily uses graphical notations to describe how software projects are designed.

Use Case Diagram

Inside the Unified Modelling Language (UML), a use case diagram is a type of behavioral diagram that is defined by and derivable from a use-case study. Its objective is to present a graphical overview of a system's functionality in terms of actors, their objectives (expressed as use cases), and any interdependencies among those use cases. The main purpose of a use case diagram is to show which actor performs which system functions. The roles played by the actors in the system can be seen.

Figure 5.3 : Use-Case Diagram

Class Diagram

A class diagram is a sort of static structure diagram used in software engineering to represent the structure of a system by displaying the classes, properties, operations (or methods), and interactions between classes. It explains in which class information is included.

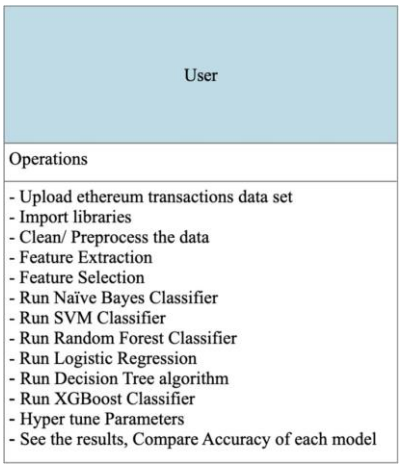

Figure 5.4 : Class diagram

Sequence Diagram

In the Unified Modelling Language (UML), a sequence diagram is a form of interaction diagram that depicts how and in what order processes interact with one another. It belongs to a message sequence chart.. Sequence diagrams are often referred to as event diagrams, event situations, and timing diagrams.

Figure 5.5 : Sequence Diagram

Collaboration Diagram

Activity diagrams are visual depictions of workflows with choice, iteration, and concurrency supported by activities and actions. Activity diagrams are used in the Unified Modelling Language to depict the business and operational step-by-step processes of system components. The whole flow of control is depicted by an activity diagram.

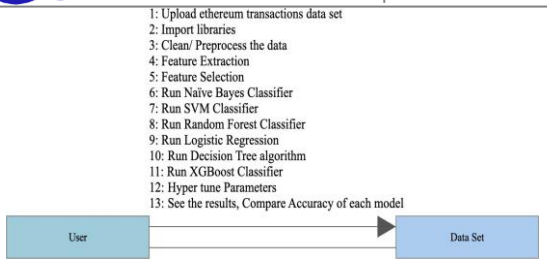

Figure 5.6 : Collaboration Diagram

6. IMPLEMENTATION

Modules to be implemented in this project are:

- Upload ethereum transactions data set
- Import libraries
- Clean/ Preprocess the data
- Feature Extraction
- Feature Selection
- Run Naïve Bayes Classifier
- Run SVM Classifier
- Run Random Forest Classifier
- Run Logistic Regression
- Run Decision Tree algorithm
- Run XGBoost Classifier
- Hypertuning Parameters
- See the results
- Compare Accuracy of each model.

Libraries Used:

1.Numpy
2.Pandas
3.Matplotlib
4.Seaborn
5.Scikit-learn
6.Pycaret
7.Plotly

7. OUTPUTS AND RESULTS

Modeling data using LR, RF and XGBoost:

1. **Logistic Regression(LR):**

[In]:

```
LR = LogisticRegression(random_state=42)

LR.fit(x_tr_resample, y_tr_resample)

# Transform test features
sc_test = sc.transform(X_test)
preds = LR.predict(sc_test)
print(y_test.shape)
y_test.value_counts()
```

[Out]:

```
(1969,)
0    1547
1     422
Name: FLAG, dtype: int64
```

[In]:

```
print(classification_report(y_test, preds))
print(confusion_matrix(y_test, preds))
```

[Out]:

```
              precision    recall  f1-score   support

           0       0.94      0.54      0.69      1547
           1       0.34      0.87      0.49       422

    accuracy                           0.61      1969
   macro avg       0.64      0.71      0.59      1969
weighted avg       0.81      0.61      0.64      1969

[[837 710]
 [ 55 367]]
```

1. **Random Forest Classifier (RF):**

[In]:

```
RF = RandomForestClassifier(random_state=42)
RF.fit(x_tr_resample, y_tr_resample)
preds_RF = RF.predict(sc_test)
print(classification_report(y_test, preds_RF))
print(confusion_matrix(y_test, preds_RF))
```

[Out]:

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      1547
           1       0.93      0.95      0.94       422

    accuracy                           0.97      1969
   macro avg       0.96      0.96      0.96      1969
weighted avg       0.97      0.97      0.97      1969
```

2. **XGBoost Classifier (XGB):**

[In]:

```
xgb_c = xgb.XGBClassifier(random_state=42)
xgb_c.fit(x_tr_resample, y_tr_resample)
preds_xgb = xgb_c.predict(sc_test)

print(classification_report(y_test, preds_xgb))
print(confusion_matrix(y_test, preds_xgb))
```

[Out]:

```
              precision    recall  f1-score   support
```

```
   0          0.99          0.98          0.99
1547
   1          0.94          0.96          0.95
422

accuracy
                         0.98        1969
macro avg
0.96       0.97          0.97        1969
weighted avg
0.98       0.98          0.98        1969

[[1519   28]
[  18  404]]
```

Hyperparameters Tuning for XGB Classifier

[In]:

```
params_grid = {'learning_rate':[0.01, 0.1, 0.5],
               'n_estimators':[100,200],
               'subsample':[0.5, 0.9],
               'max_depth':[3,4],
               'Colsample_bytree':[0.3,0.7]}

grid = GridSearchCV(estimator=xgb_c, param_grid=params_grid, scoring='recall', cv = 10, verbose = 0)

grid.fit(x_tr_resample, y_tr_resample)

print(f ' Best params found for XGBoost are: {grid.best_params_}')

print(f ' Best recall obtained by the best params: {grid.best_score_}')
```

[Out]:

```
Best params found for XGBoost are:
{'colsample_bytree': 0.3, 'learning_rate':
0.5, 'max_depth': 4, 'n_estimators': 200,
'subsample': 0.5}
Best recall obtained by the best params:
0.9844551950622037
```

[In]:

```
preds_best_xgb = grid.best_estimator_.predict(sc_test)

print(classification_report(y_test, preds_best_xgb))

print(confusion_matrix(y_test, preds_best_xgb))
```

[Out]:

```
              precision    recall  f1-score
support
   0          0.98          0.98          0.98
1547
   1          0.93          0.94          0.94
422

accuracy
                         0.97        1969
macro avg
0.96       0.96          0.96          1969
weighted avg
0.97       0.97          0.97          1969

[[1518   29]
[  25  397]]
```

[In]:

```
# Plotting AUC for tuned XGB Classifier

probs = xgb_c.predict_proba(sc_test)

pred = probs[:,1]

fpr, tpr, threshold = roc_curve(y_test, pred)

roc_auc = auc(fpr, tpr)


plt.figure(figsize=(12,8))

plt.title('ROC for tuned XGB Classifier')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)

plt.legend(loc = 'lower right')

plt.plot([0,1], [0,1], 'r--')

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()
```

[Out]:

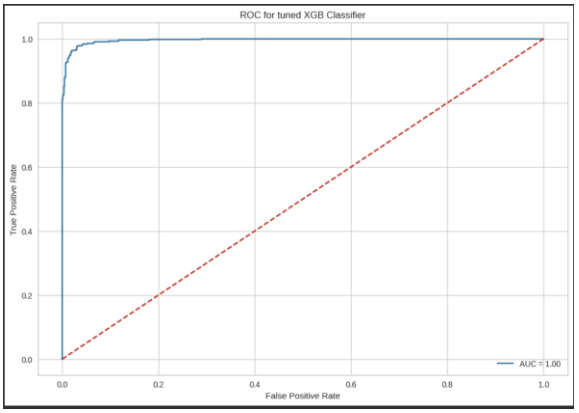

Figure 7.1 : AUC graph for tuned XGB Classifier

8. CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

"Identification of fraudulent transactions on the Ethereum blockchain" falls within the category of CART (Classification and Regression Trees) problems. This project sets out to address the fundamental issue of identifying and thoroughly verifying the transactions on the Ethereum blockchain.

We suggested a technique based on knowledge-based patterns, tactics, and machine learning approaches. These approaches are suggested in order to improve the accuracy of transactions on Ethereum.It focuses on analyzing daily cryptocurrency transactions from different accounts, providing the data to a machine learning model to train it, and then checking its correctness so that we can utilize this model in the future based on the results. It entails steps like data gathering, data preprocessing, feature extraction, feature selection, data classification using various ML models, model training, and model testing.

This study area has advanced over the last five years, with models reaching efficiencies of about 85%-92%. However, it still lacks accuracy in some ml models. Additionally, with increased adoption of crypto transactions, it has several kinds of application issues. As a result, we must upgrade our smart contracts regularly to protect the security of all user funds on multiple crypto exchanges.

We attempted to construct a model using the csv dataset containing ethereum transactions as input, analyzing the data features and determining whether these features might be used to develop a more accurate model, and we were successful in developing a highly precise prediction model with an accuracy of 99%.

Thus, we can draw the conclusion that the detection of fraudulent transactions on the Ethereum blockchain has a very promising future.

8.2 Future Scope

The analysts predict that the total market capitalization of cryptocurrencies would more than triple, reaching roughly $5 billion by 2030. The key drivers will be enhanced global payment system transparency and rising demand for international remittances.

Future research in this area should consider the price fluctuations of cryptocurrencies as well as the claim that taking into account multiple features improves the performance of different machine learning algorithms.

We might even develop brand-new algorithms to solve this issue.

The findings of this project reveal that investigations into cryptocurrency scams are rapidly increasing in both volume and breadth, even though we are still in the early phases of thinking about potential challenges and situations using cryptocurrencies.

The future scope of this project would emphasize the significance of enhanced industry collaboration and agreement on definitions of ethereum fraud, as well as to develop new and improved machine learning algorithms to identify and prevent these frauds well in advance, in order to address the concerns raised about the viability of cryptocurrencies as a means of exchange.

The application of active learning and deep learning techniques to identify fraudulent transactions and boost user and governmental confidence may be another possible topic of research.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our guide Ms.Nemmani Swapna for guiding and helping us in successfully completing our project.

REFERENCES

[1] A. Averin and O. Averina, ‘‘Review of blockchain technology vulnerabilities and blockchain-system attacks,’’ in *Proc. Int. Multi-Conf. Ind. Eng. Mod. Technol.*, Oct. 2019, pp. 1–6, doi: 10.1109/FarEast- Con.2019.8934243.

[2] C. S. Wright, ‘‘Bitcoin: A peer-to-peer electronic cash system,’’ *SSRN Electron. J.*, pp. 1–9, Jan. 2019, doi: 10.2139/ssrn.3440802.

[3] C.K.Frantz and M.Nowostawski,‘‘Frominstitutionstocode:Towards automated generation of smart contracts,’’ in *Proc. IEEE 1st Int. Workshops Found. Appl. Self Syst.*, Sep. 2016, pp. 210–215.

[4] M. In and F. Of, ‘‘Dubai aims to be a city built on blockchain where financial regulation goes in a republican era,’’ *Wall Str. J.*, vol. 4, pp. 3–6, Apr. 2017. [Online]. Available: https://www.wsj.com/articles/dubai-aims-to-be-a-city-built-on-blockchain-1493086080

[5][5] S.KimandG.C.Deka,*AdvancedApplicationsofBlockchainTechnology*, vol. 60. Singapore: Springer, 2020.

[6] M. Singh and S. Kim, ‘‘Blockchain technology for decentralized autonomous organizations,’’ in *Proc. Adv. Comput.*, vol. 115, Oct. 2019, pp. 115–140.

[7] C. E. Brown, O. Kuncar, and J. Urban, ‘‘Formal verification of smart contracts,’’ in *Proc. ACM Workshop Program. Lang. Anal. Secur.*, 2017, pp. 91–96. [Online]. Available: http://proofmarket.org

[8] L. Clohessy, H. Treiblmaier, T. Acton, N. Rogers, “Antecedents of blockchain adoption: an integrative framework”, StratChange 29 (2020) 501e515, https://doi.org/10.1002/jsc.2360.

[9] X. Li, A.B. Whinston, "Analyzing cryptocurrencies", Inf SystFront22(2020) 17e22, https://doi.org/10.1007/s10796-019-09966-2.

[10] J. Liu, A. Serletis, "Volatility in the cryptocurrency market",open econ, Rev 30 (2019) 779e811, https://doi.org/10.1007/s11079-019-09547-5.

[11] R. Aziz, M.F. Baluch, S. Patel, A.H. Ganie, "LGBM: a machine learning approach for Ethereum fraud detection" ,IntJ Inf Technol14(2022) 1e11 , https://doi.org/10.1007/s41870-022-00864-6.

[12] K. Ajay, K. Abhishek, P. Nerurkar, M.R. Ghalib, A. Shankar, X. Cheng, "Secure smart contracts for cloud based manufacturing using Ethereum blockchain", Trans Emerg Telecommun 13 (2020) 4121e4129,

https://doi.org/10.1002/ ett.4129.

[13] Teng Hu, Xiaolei Liu, Ting Chen, Xiaosong Zhang, Xiaoming Huang, Weina Niu, Jiazhong Lu, Kun Zhou, Yuan Liu, "Transaction-based classification and detection approach for ethereum smart contract", Inf Process Manag 58 (2021) 102457e102462, https://doi.org/10.1016/j.ipm.2020.102 462.

[14] Q. Yuan, B. Huang, J. Zhang, J. Wu, H. Zhang, X. Zhang, "Detecting phishing scams on ethereum based on transaction records", in: IEEE intl. Symposium on circuits and systems (ISCAS), IEEE, 2020, pp. 1e5,

https://doi.org/10.1109/ ISCAS45731.2020.9180815.

[15] R.F. Ibrahim, A. Mohammad Elian, M. Ababneh, "Illicit account detection in the ethereum blockchain using machine learning", in: IEEE intl. Conf. On information technology, ICIT, 2021, pp. 488e493. https://doi: 10.1109/ICIT52682.2021. 9491653.

[16] Madhuparna Bhowmik, Tulasi Sai Siri Chandana, Dr. Bhawana Rudra , "Comparative Study of Machine Learning Algorithms for Fraud Detection in Blockchain" , IEEE Xplore Part Number: CFP21K25-ART , DOI: 10.1109/ICCMC51019.2021.9418470 .

[17] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur and H. -N. Lee, "Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract", in IEEE Access, vol. 10, pp. 6605-6621, 2022, doi: 10.1109/ACCESS.2021.3140091.

[18] Kaggle dataset,

"https://www.kaggle.com/code/soheiltehranipour/how-to-detect-fraud-in-crypto?scriptVersionId=118280787&cellId=4"

[19] Naïve Bayes, "https://scikit-learn.org/stable/modules/naive_bayes.html"

[20] Support Vector Machines (SVM),

"https://www.tutorialspoint.com/scikit_learn/scikit_learn_support_vector_machines.htm"

[21] von Winterfeldt, Detlof; Edwards, Ward (1986). "Decision trees". *Decision Analysis and Behavioral Research*. Cambridge University Press. pp. 63–89. ISBN 0-521-27304-8.

[22] Kamiński, B.; Jakubczyk, M.; Szufel, P. (2017). "A framework for sensitivity analysis of decision trees". *Central European Journal of Operations Research*. 26 (1): 135–159. doi:10.1007/s10100-017-0479-6. PMC 5767274. PMID 29375266

[23] Tolles, Juliana; Meurer, William J (2016). "Logistic Regression Relating Patient Characteristics to Outcomes". *JAMA*. 316 (5): 533–4. doi:10.1001/jama.2016.7653.

ISSN 0098-7484. OCLC 6823603312. PMID 27483067 .

[24] Hosmer, David W.; Lemeshow, Stanley (2000). "*Applied Logistic Regression"* (2nd ed.). Wiley. ISBN 978-0-471-35632-5.

[25] Ho, Tin Kam (1995). "*Random Decision Forests*" (PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.Archived from the original (PDF) on 17 April 2016.

[26] Ho TK (1998). "The Random Subspace Method for Constructing Decision Forests" (PDF). *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 20 (8): 832–844. doi:10.1109/34.709601. S2CID 206420153

[27] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). "*The Elements of Statistical Learning*" (2nd ed.). Springer. ISBN 0-387-95284-5.

[28] Piryonesi, S. Madeh; El-Diraby, Tamer E. (2020-03-01). "Data Analytics in Asset Management: Cost-Effective Prediction of the Pavement Condition Index". *Journal of Infrastructure Systems*. 26 (1): 04019036. doi:10.1061/(ASCE)IS.1943-555X.0000512. ISSN 1943-555X. S2CID 213782055.

[29] Hastie, T.; Tibshirani, R.; Friedman, J. H. (2009). "10. Boosting and Additive Trees". "*The Elements of Statistical Learning"* (2nd ed.). New York: Springer. pp. 337–384. ISBN 978-0-387-84857-0. Archived from the original on 2009-11-10.

[30] Piryonesi, S. Madeh; El-Diraby, Tamer E. (2021-02-01). "Using Machine Learning to Examine Impact of Type of Performance Indicator on Flexible Pavement Deterioration Modeling". *Journal of Infrastructure Systems*. 27 (2): 04021005. doi:10.1061/(ASCE)IS.1943-555X.0000602. ISSN 1076-0342. S2CID 233550030.

[31]A brief introduction to XGBoost , "https://towardsdatascience.com/a-brief-introduction-to-xgboost-3eaee2e3e5d6#:~:text=XGBoost%20vs%20Gradient%20Boosting,can%20be%20parallelized%20across%20clusters".

[32] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," CoRR, vol. abs/1603.02754, 2016. [Online]. Available: http://arxiv.org/abs/1603.02754

[33] Intro to Numpy, "https://numpy.org/"

[34] Intro to Pandas, "https://en.wikipedia.org/wiki/Pandas_(software)"

[35] Matplotlib, "https://en.wikipedia.org/wiki/Matplotlib"

[36] Seaborn, "https://seaborn.pydata.org/"

[37] Scikit-learn, "https://pypi.org/project/scikit-learn/"

[38] Pycaret, " https://pycaret.readthedocs.io/en/stable/"

[39] Plotly, "https://plotly.com/python/"

[40] Types of Software Testing, "https://www.javatpoint.com/types-of-software-testing"