A Machine Learning Methodology for Classifying Chronic Kidney Diseases

By

DAKSH SHARMA

A thesis submitted to the
Department of Computer Science
in conformity with the requirements for
the Degree of Master of Science

Bishop's University
Canada
December 2023

Abstract

Chronic kidney disease (CKD) is a significant global health issue that requires urgent attention. Unfortunately, patients in the early stages of CKD may not exhibit any noticeable symptoms, leading to delayed diagnosis and treatment. Recent advancements in Machine Learning (ML) offer an effective tool for clinicians to detect the disease early and provide prompt treatment. A significant amount of research has been conducted on this topic. In this thesis, we propose a model for CKD prediction based on Ensemble Learning (EL) concepts. Five well-known supervised learning algorithms, three of which are based on the EL techniques of bagging and boosting, are used as components to form an ensemble model applying the EL stacking strategy. A CKD dataset from the University of California Irvine (UCI) machine learning repository is used for experimental validation of the model. The dataset had many missing values, which were handled using iterative imputation. The Random Forest Feature Importance (RF-FI) is used to select the most relevant features and reduce the feature vector dimensionality for the prediction task. The ensemble model achieved an average accuracy rate of 99% after running 10-fold cross-validation. The experimental procedure mentioned previously was also applied to a dataset related to cardiovascular diseases, which is discussed in the experimental section. Our study revealed that our model can be beneficial in the early detection of CKD and other diseases, leading to prompt treatment of affected patients.

Keywords: Chronic Kidney Disease, Machine Learning, Ensemble Learning, Classification, Boosting, Bagging, Stacking, Imputation, Feature Importance, Hyper-Parameter Tuning

Acknowledgments

I am immensely grateful to my supervisor, Dr. Madjid Allili, for his invaluable advice and guidance that have played a pivotal role in shaping my academic and professional growth. The insightful feedback and constructive criticism provided by Drs. Stefan Bruda, Rachid Hedjam, and Mohammed Ayoub Alaoui Mhamdi from the Department of Computer Science at Bishop's University have been instrumental in transforming me into a confident individual. I owe a debt of gratitude to Steve K. Houley for his constant encouragement and support, which has enabled me to explore my academic pursuits beyond my comfort zone. Lastly, I cannot thank my family enough for their unwavering support and generosity throughout my academic journey.

Table of Contents

List of Figures

List of Tables

List of symbols and abbreviations

- CKD – Chronic kidney disease
- ML – Machine Learning
- DV – Data Visualization
- EDA – Exploratory Data Analysis
- KDE – Kernel Density Estimation
- DT – Decision Tree
- ADA – Adaptive Boosting
- ANN – Artificial Neural Network
- DPP – Data Pre-Processing
- GB – Gradient Boosting
- IQR – Inter Quantile Range
- RF-FI – Random Forest Feature Importance
- CBC – Complete Blood Count
- RBC – Red Blood Cell
- WBC – White Blood Cell
- PCV – Packed Cell Volume
- HB – Haemoglobin
- Std – Standard Deviation
- IQR – Inter Quantile Range
- AUC – Accuracy Curve
- ROC – Receiver Operating Characteristic Curve
- TP – True Positive
- FP – False Positive
- FN – False Negative
- TN – True Negative

Chapter 1: Introduction

1.1 Introduction

Chronic Kidney Disease (CKD) is a significant health concern worldwide due to its insidious nature during the initial stages. The challenge lies in the silent progression of CKD without apparent symptoms, leading to delayed diagnoses and delayed treatment initiation. Early detection methods are crucial to identifying CKD in its preliminary stages, underscoring the need for comprehensive and routine health screenings [1]. Timely intervention is essential, as it can significantly slow or impede the progression of the disease. The importance of early intervention strategies cannot be overstated, as healthcare professionals can effectively mitigate the decline in kidney function caused by CKD through meticulous monitoring and appropriate medication [2]. Poorly managed CKD can lead to dire consequences, potentially leading to kidney failure, where the therapeutic options often narrow down to life-altering treatments such as dialysis or kidney transplantation [3],[4].

Recent advancements in machine learning and computational capabilities have sparked optimism in the diagnosis of CKD [5]. The precision, cost-effectiveness, and adaptability of machine learning, coupled with evolving information technology and electronic health data, make it a promising candidate for determining various health statuses [6],[7]. Its application in diverse medical domains showcases its remarkable potential in deciphering complex health conditions, from diagnosing heart diseases to addressing acute kidney injuries [8]. Our research focuses on leveraging ensemble learning techniques in machine learning to improve the accuracy of chronic kidney disease (CKD) prediction models. The objective is to refine disease prediction models by utilizing diverse data patterns and complex relationships within the dataset.

Our study focuses on machine learning-based disease prediction techniques, potentially bringing timely interventions and enhanced management strategies for CKD. Using multiple models and ensemble techniques ensures a more reliable and precise approach to CKD diagnosis. By aggregating predictions from numerous models, our approach enhances CKD predictive capabilities, ensuring accurate diagnosis and providing healthcare professionals with a valuable toolset to improve patient care. We chose the ensemble approach as ensemble methods work on the principle of combining diverse models, which, when aggregated, can provide more accurate predictions than any single model alone [9]. This approach leverages the strengths of various models while mitigating their weaknesses. For instance, some models might excel in capturing specific patterns or relationships within the data, while others might be better suited for different aspects. Combining these models, ensemble learning aims to create a more robust and comprehensive predictive model.

To achieve our research goal, we employ various models, including Random Forest (RF), Gradient Boosting (GB), ADA Boosting (ADA), Support Vector Machines (SVM), and Artificial Neural Networks (ANN). Out of five models, three are based on ensemble bagging (RF) and boosting techniques (ADA and GB), while the two left, SVM and ANN, are chosen as they provide a different approaches than ensemble learning. Our ensemble models are based on decision trees, which serve as the foundational base learner.

Our approach involved the stacking ensemble learning technique to synthesize a model and then using max voting to produce our final proposed CKD classification model, providing a precise and reliable approach to CKD diagnosis. We stacked RF, SVM, GB, ADA, and ANN, followed by 10-fold cross-validation, which increased the reliability of our model, and compared our results for the stacked model with the other five component models. Our methods also address missing data using iterative imputation techniques in conjunction with decision tree-based models, significantly enhancing accuracy. We used random forest feature importance scores to identify the most impactful subset for classifying CKD problems. This helped us to develop a comprehensive methodology that culminates in a reliable and efficient diagnostic tool for early CKD detection and diagnosis.

1.2 Thesis Outline

Chapter 2 delves into an in-depth discussion of the related work undertaken in this field and explores concepts necessary for developing our models. It provides a clear and concise elucidation of the background and reasoning behind our selection of specific techniques and models. Chapter 3 comprehensively elaborates our methodology, which employs robust data processing and strategic use of ensemble methods to enhance predictive performance. The culmination of our thesis is presented in Chapter 4, where we showcase our experimental work and in-depth analysis. Lastly, Chapter 5 delivers conclusive findings that set the stage for future research endeavors.

Chapter 2: Literature Review

This chapter holds great importance as it delves into the related literature and serves as the foundation for the research methodology that will be expanded upon in Chapter 3. Section 2.1 covers the previous work done in this area, followed by Section 2.2 which provides an in-depth overview of ensemble learning and its primary strategies. In Section 2.3, we introduce the machine learning classifiers that have been utilized in this study. Additionally, Sections 2.4 and 2.5 offer a comprehensive account of the feature selection method and the missing data imputation strategy employed in this research.

2.1 Related Work

Nephropathy, or kidney damage, is called kidney disease. People with kidney disease have kidney failure, which can lead to kidney failure if not treated quickly. According to the National Kidney Foundation, chronic kidney disease affects 10% of the world's population, and millions of people die each year due to inadequate treatment. Recent advances in ML and DL-based kidney disease testing may bring hope to countries that cannot manage kidney disease testing.

Bemando et al. [10] investigated the relationship between blood-related diseases and their features using Gaussian Naive Bayes, Bernoulli Naive Bayes, and Random Forest classifier methods. These three algorithms anticipate and offer statistical findings in a variety of ways. In this experiment, we discovered that Naive Bayes estimated accuracy was higher than other algorithms.

Kumar and Polepaka [11] devised a technique for illness prediction in the medical field. They employed Random Forest and CNN as well as other machine learning methods. These algorithms deliver better results for illness dataset classification, precision, recall, and F1-score. In this experiment, Random Forest outperformed different algorithms regarding accuracy and statistical performance.

Singh et al. [12] developed a technique for predicting medical-related illness datasets. For improved prediction, they utilized a support vector machine classifier. The accuracy ranged from 73 to 91 percent, and the author eventually improved accuracy to 91 percent [13]. Desai et al. devised a technique for illness prediction in the medical field. The author employed back-propagation NN and LR classification algorithms in this study. These two strategies provide distinct outcomes, with statistical analysis and logistic regression yielding a more accurate model than other algorithms.

Patil et al. [14] created a database for cardiovascular-related medical conditions. On a disease dataset, the authors employed machine learning approaches such as a Support Vector Machine and Cuckoo search optimized Neural Network, and the support vector machine estimated 94.44 percent improved accuracy. They observed the illness dataset for statistical analysis by Liu et al. [15]. They estimated superior findings for specificity, sensitivity, and positive and negative predictive values using machine learning approaches such as support vector machines. Acharya et al. [16] reviewed the medical-linked illness dataset for better statistical analysis outcomes; they employed several machine learning techniques, such as CNN, and applied machine learning algorithms to the ECG dataset, achieving a classification accuracy of 94 percent.

Wasle et al. [17] devised a statistical analysis technique to examine the chronic kidney disease dataset; the authors employed a variety of machine-learning approaches. They used Naive Bayes, Decision Trees, and Random Forest to improve prediction, and they discovered that Random Forest computed greater classification accuracy than the other algorithms. Nithya et al. [18] developed a categorization and cluster-based analysis method on the kidney disease dataset. They calculated 99.61 percent classification accuracy using Artificial Neural Networks for Kidney Disease Image Prediction.

Al Imran et al. [19] examined machine learning techniques to analyze datasets for chronic renal disease. For statistical analysis such as F1-score, Precision, Recall, and AUC, the authors employed Logistic Regression and Feedforward Neural Networks and generated better results than previous algorithms. Navaneeth and Suchetha [20] devised a method for predicting chronic renal disease using a dataset. They employed machine learning methods such as CNN and SVM. The authors estimated greater accuracy, sensitivity, and specificity findings after the prediction.

Brunetti et al. [21] used a system or method for chronic kidney disease datasets. The authors used the CNN machine learning technique and calculated 95% classification accuracy for the disease dataset. Hodneland et al. [22] used image registration to detect renal morphologic changes in CKD diagnosis. Vasquez-Morales et al. [23] established a classifier based on the neural network using large-scale CKD data, and the model's accuracy on their test data was 95%. In addition, most previous studies utilized the CKD data set obtained from the UCI machine learning repository.

Chen et al. [24] used k-nearest neighbor (KNN), support vector machine (SVM), and soft independent modeling of class analogy to diagnose CKD; KNN and SVM achieved the highest accuracy of 99.7%. Aljaaf et al. [25] used a fuzzy rule-building expert system, fuzzy optimal associative memory, and partial least squares discriminant analysis to diagnose CKD, and the range of accuracy in those models was 95.5%-99.6%. Their studies have achieved good results in the diagnosis of CKD.

Nishant et al. [26] used MLP, SVM, KNN, C4.5 decision tree, and random forest (RF) to diagnose CKD, and the RF achieved an accuracy of 100%. In the models established by Boukenze et al. [27], MLP achieved the highest accuracy of 99.75%. The studies focus mainly on setting models and achieving an ideal result. However, a complete process of filling in the missing values is not described in detail, and no feature selection technology is used to select predictors. Rady et al. [28] used SVM and neural networks to diagnose CKD, and the accuracy of the models was 97.75% and 99.75%, respectively. In the models established by Gunarathne et al. [29], zero was used to fill out the missing values, and the decision forest achieved the best performance with an accuracy of 99.1%.

The mean imputation fills in the missing values in the above models and depends on the samples' diagnostic categories. As a result, their method could only be used when the diagnostic results of the samples are known. In reality, patients might miss some measurements for assorted reasons before diagnosing. In addition, for missing values in categorical variables, data obtained using mean imputation might have a significant deviation from the actual values. For example, for variables with only two categories, we set the categories to 0 and 1, but the mean of the variables might be between 0 and 1.

2.2 Ensemble Learning

Ensemble Learning is a Machine Learning technique that makes predictions by combining the results of several machine learning models, also known as base learners. This approach can be advantageous when individual models are prone to overfitting or need help recognizing the underlying data's complexity [30]. Although various strategies can combine machine learning models, including simple ones such as Majority Voting and Averaging, the Bagging, Boosting, and Stacking techniques are most commonly used in practice. Their popularity is due to their ease of implementation and success on various predictive modeling problems.

Bagging

Bagging is so named because it combines Bootstrapping and Aggregation to form one ensemble model. Given a collection of base learners and a dataset, the main idea consists of training each of them using a slightly different dataset generated by taking a bootstrap sampling of the original dataset, that is, by iteratively resampling with replacement of the dataset. The predictions of the base learners are then aggregated to form a more efficient predictor. When dealing with a classification problem, the base learner predictions are typically combined using plurality votes or by averaging the estimated class probabilities. Because of the aggregation process, bagging effectively reduces the variance of the individual base learners. Bagging works exceptionally well for unstable, high-variance base learners' algorithms whose predicted output undergoes significant changes in response to small changes in the training data [31]. A typical example of an ensemble learner obtained through bagging is the random forest classifier discussed in section 2.2.3.

Boosting

Boosting is a supervised machine learning strategy combining multiple base model predictions to generate a more efficient ensemble model. Unlike bagging, it focuses on successively training the basic models in a way that emphasizes misclassified samples from prior iterations. The goal is to prioritize samples incorrectly categorized in previous iterations, allowing the ensemble model to learn from its mistakes and improve its overall performance. Typically, the training dataset is left unchanged. Instead, the learning algorithm is modified to pay more attention to instances of the training data misclassified incorrectly by previously added ensemble members.

The idea of combining many weak learners into a strong learner was theoretically proposed, and many algorithms were unsuccessful. Only when the Adaptive Boosting (AdaBoost) algorithm was developed was boosting demonstrated as an effective ensemble method [32]. Sections 2.2.4 and 2.2.5 present two machine learning prediction methods based on ensemble learning boosting strategy: the Adaboost and Gradient Boosting algorithms.

Stacking

Stacking involves training multiple models and combining their outputs using another model called the combiner or the second-level learner. The combiner model aggregates the predictions of the ensemble components to produce a more accurate result. In principle, any machine learning model can be the second-level learner. However, the rule of thumb is to generally let the model's complexity reside at the base models' level and use simple models as combiners. Stacking can be very effective when the base models are complementary in their strengths and weaknesses, as it can help to capture a broader range of information from the underlying data. The main advantage of stacking is that it can combine models with different types of architectures. This means it can combine models using different features or based on different algorithms. The diversity of the ensemble members is desirable since they are generally constructed in very different ways, ensuring that they make different assumptions and, therefore, have fewer correlated prediction errors.

2.3 Machine Learning Classification Algorithms

This section discusses the fundamental machine learning algorithms that comprise the final model in Chapter 3. The decision tree algorithm (DT) is used as the base learner for bagging and boosting techniques, which reduce variance and improve accuracy—bagging forms the random forest (RF) classifier while boosting forms both Adaboost and GB models. Two complementary models, Support Vector Machine (SVM) and Artificial Neural Network (ANN), are included since they are built based on strategies different from ensemble learning. These algorithms are combined through stacking to improve the accuracy and performance of the model.

2.3.1 Decision Tree

Decision Tree is a classification technique based on the recursive partitioning of a dataset into distinct segments. This method constructs a tree-like structure by evaluating attributes and their values at each node, sequentially breaking down the dataset into smaller subsets [33]. The core principle involves identifying the most significant feature that best separates the data, creating branches that lead to subsequent nodes with refined criteria. This recursive process continues until a defined stopping criterion is met, such as reaching a certain depth or achieving purity in the subsets. Ultimately, the decision tree becomes a hierarchical flowchart where each internal node represents a feature, and each leaf node signifies the final classification outcome Fig 2.1.

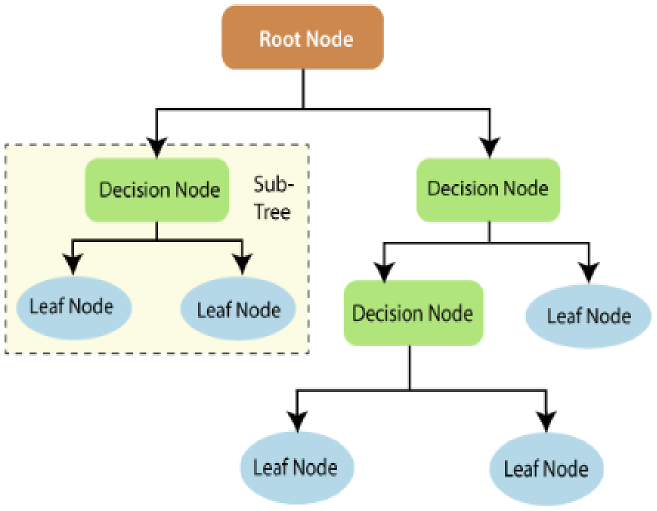


Figure 2.1 Decision Tree Flowchart [34]

Decision Trees offer an interpretable and understandable representation of decision-making processes, providing insights into the most critical features influencing classification outcomes. The algorithms work based on splits, which are determined by the following:

Entropy

In Machine Learning, entropy measures the level of disorder or uncertainty in each data set. It can also be seen as quantifying the amount of information in the dataset [35]. Entropy is defined mathematically for random distributions and can be extended for any dataset S consisting of n classes using the formula:

$$H(S) = \sum_{k=1}^{n} -p_k log_2(p_k) \tag{2.1}$$

Here, p_k represents the probability of class k occurring within the data set, which is the proportion of the data points that belong to a class k to the total number of data points in S.

Information Gain

Information gain is the measure of reduction in entropy resulting from a dataset split based on a specific attribute. We typically use it to determine the usefulness of a feature by partitioning the dataset into more homogeneous subsets concerning the class labels or
target variable [36]. The higher the information gains, the more valuable the feature is in predicting the class label or the target variable. Given a data set S and an attribute A, the information gain of A with respect to S is defined as:

$$IG(S, A) = H(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} H(S_v) \tag{2.2}$$

where υ represents the possible values taken by the feature A, $|S_v|$ is the number of instances in the subset S with the value υ for the attribute A.
The attribute with the least entropy should be used to find the optimal decision tree with the best feature split. This can be obtained via information gain, which is the difference in entropy before and after a split of an attribute.

Gini Impurity

An alternative way of splitting a decision tree is via the Gini Index. The Gini Index or Impurity measures the probability of a random instance being misclassified when chosen randomly from a data set. The Gini index is calculated using the formula given below.

$$GiniIndex = 1 - \sum_{j=1}^{n} p_j^2 \tag{2.3}$$

Where $p_j$denotes the probability that a training instance belongs to a class, j, d, n is the total number of classes. The value of Gini impurity ranges from 0 to 1. Zero refers to the pure node, where all elements in the node belong to one single class, and 0.5 refers to the impure node, with elements in the node belonging to multiple classes. The optimal split is the one having the lowest Gini index. The simplicity of the decision tree method makes it a valuable tool for understanding and developing predictive models that are widely used in machine learning and data analysis [37].

2.3.2 Random Forest

Random Forest is a supervised machine-learning algorithm introduced by Leo Breiman and Adele Cutler in 2001 [38]. It is an ensemble learning-based classifier obtained by bagging a set of decision trees built from the training data. Each decision tree forming a random forest is made unique by using a distinct random subset of data obtained from the bootstrap sampling of the original dataset, as illustrated in Figure 2.3. This results in a more accurate model than a single decision tree. For prediction problems, the majority vote of the base learners in a random forest is used to obtain the final prediction.

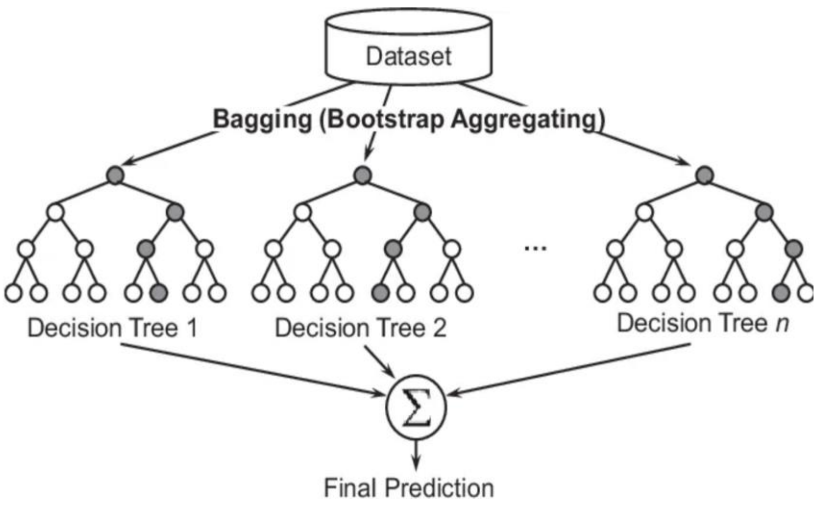


Figure 2.2 An illustration of bootstrap sampling in Random Forest [38]

2.3.3 Support Vector Machines

Support Vector Machine (SVM) is a supervised learning method useful for regression and classification. The original version of SVM was developed for linear separation of two classes [39]. This early limitation was later overcome by allowing non-separable classes and non-linearly separable classes. For binary classification in a d-dimensional space, the SVM,

$$f\ (x) = w^T x + b = 0 \tag{2.4}$$

technique constructs a hyperplane described by an equation of the form that correctly separates two classes with a maximum margin, as illustrated in Figure 2.3.

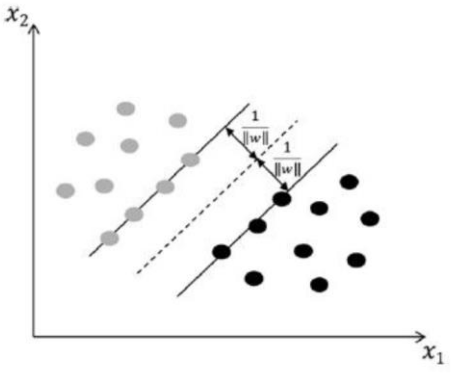


Figure 2.3 Maximum margin separating hyperplane for SVM [39]

If the training data $\{(x_i, y_i)\}_{i=1}^{n}$ is linearly separable, the maximum margin can be defined by two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible. The maximum margin separating the hyperplane is the hyperplane that lies halfway between them. The equation of this hyperplane is found using a quadratic optimization procedure, and it is completely determined by the vectors x_i that lie near the separation region. These vectors are called support vectors. A soft separation margin can be obtained when the data is not completely separable by a linear boundary, forcing the procedure to tolerate some classification errors.

However, when the data requires a non-linear boundary separation in the original feature space, the SVM algorithm addresses this limitation by mapping the data into higher-dimensional space where linear separation becomes possible. If we assume that $\psi(x)$ is the transformation that achieves this objective, the quadratic optimization procedure to find the separation hyperplane in the linear case applied in the new feature space will contain the quantities $\psi(x_i)^T\,\psi(x_j)$ for pairs of feature vectors x_i and x_j. Since it is generally impossible to find explicitly the transformation $\psi(x)$, [38] proposed the idea called the kernel trick which makes use of an appropriate kernel function $k(x_i, x_j)$ to replace the dot products $\psi(x_i)^T\,\psi(x_j)$ in the optimization procedure. Many categories of kernel functions are used in practice, such as the homogeneous and non-homogeneous polynomial, the radial-based, and the sigmoid. The selection of the kernel function is related to the data distribution in the original feature space. In real-world problems, we examine closely the properties of our data to select an appropriate kernel function, and we can also test experimentally different kernels and select the one that provides the best performance.

2.3.4 ADA Boost Classifier Tree

We recall that boosting is a general strategy for learning classifiers by combining simpler ones. A popular boosting algorithm is AdaBoost, which is short for adaptive boosting and was introduced by Yoav Freund and Robert Schapire in 1995 [40]. It is so-called because it is adaptive in that subsequent weak learners are chosen to compensate for the weaknesses of the previous classifiers [41]. AdaBoost is straightforward to use and implement and generally gives very effective results [42].

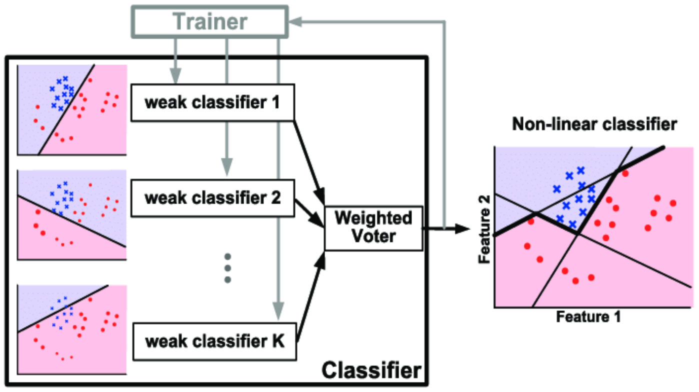


Figure 2.4 Example of ADA Boost classifier [42]

Assume we are given the training data $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, where $x_i \in R^k$ and $y_i \in \{-1, 1\}$. Suppose we are given a set of weak classifiers $\{f_m\}_{m=1}^{M}$ where $f_m(x) \in \{-1, 1\}$, and $0 - 1$ loss function I, defined as:

$$I(f_m(x_i), y_i) = \begin{cases} 0 \ \ if \ \ f_m(x_i) = y_i \\ 1 \ \ if \ \ f_m(x_i) \neq y_i \end{cases} \tag{2.5}$$

The final classifier $f(x)$ is a weighted average of all sequential models $f_m(x)$:

$$f(x) = sign\left(\sum_{m=1}^{M} \alpha_m f_m(x)\right) \tag{2.6}$$

where the model-weights α_m are given by the misclassification rate ϵ_m of the model f_m . The sign of $f\ (x)$ identifies the predicted object class and the absolute value gives the confidence in that classification.

ALGORITHM 1: THE CONSTRUCTION OF ADABOOST CLASSIFIER

```
1)  Input: Data $\{(x_i, y_i): 1 \le i \le N\}$ and the loss function $I(y, f\ (x))$
2)  For $i = 1$ to $N$ , set $w_i^{(1)} = \frac{1}{n}$
3)  Repeat for $m = 1$ to $M$ :
    a)  Fit the classifier $f_m\ (x)$ using weights $w_i^{(m)}$ to minimize the objective function:
            $\epsilon_m = \sum_{i=1}^{N} w_i^{(m)}\ I\left(y_i, f_m(x_i)\right)$    (2.7)
    b)  Compute the aggregation weight:
            $\alpha_m = ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$    (2.8)
    c)  For $i = 1$ to $N$ , update the weight:
            $w_i^{(m+1)} = w_i^{(m)} exp\left(\alpha_m I\left(y_i, f_m(x_i)\right)\right)$    (2.9)
    d)  Normalize the weights so as:
            $\sum_{i=1}^{N} w_i^{(m+1)} = 1$    (2.10)
4)  For $i = 1$ to $N$ , output
            $f(x_i) = sign\left(\sum_{m=1}^{M} \alpha_m f_m(x_i)\right)$    (2.11)
```

AdaBoost is a greedy algorithm that builds the strong classifier $f(x)$ incrementally by optimizing the weights α_m for each added weak classifier. Equation (2.8) is derived from the exponential loss function used for classification problems in machine learning. This loss function is preferred in Adaboost for its sensitivity to misclassifications and outliers. The quantities ϵ_m represent weighted measures of the error rates of each base classifier on the data set. The weighting coefficients α_m in Equation (2.9) give greater weight to the more accurate base classifiers added to the sum. Equation (2.10) shows that the weighting coefficients w_i^m are increased for misclassified data points.

2.3.5 Gradient Boosting Classifier

The fundamental intuition behind gradient boosting is iteratively building a complex classification or regression model by adding simple models [43]. Each new simple model added to the ensemble compensates for the weaknesses of the current ensemble.

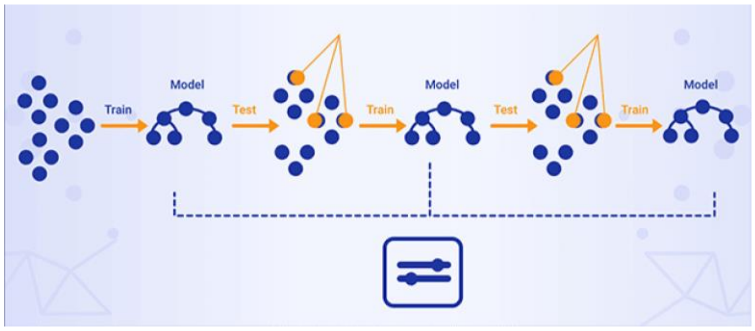


Figure 2.5 Gradient Boosting Flowchart [44]

More concretely, we take an ensemble of simple models $\{fk\}_{k\in K}$ and additively combine them into a single, more complex model:

$$F = \sum_{k} \lambda_k f_k \tag{2.12}$$

Each model f_k might be a poor fit for the data, but a linear combination of the ensemble will provide a better fit Consider the training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$. We can assume that the predictor $f_k\,(x)$ at stage k incurs the loss $L(\,f_k\,(x), y)$, i.e., the difference between the actual and the predicted variables:

$$L(f_k\,(x), y) = \sum_{i=1}^{N} L(f_k\,(x_i), y_i) \tag{2.13}$$

At this point, we want to minimize the loss function $L(f_k\,(x), y)$ with respect to f_k. We therefore train a function $h_k\,(x)$ to approximate the negative gradient of $L(f_k\,(x), y)$ with respect to the predictor f_k, that is:

$$h_k\,(\mathrm{x}) = -\frac{\partial L(f_k\,(x), y}{\partial f_k\,(x)} \tag{2.14}$$

and obtain a new predictor $f_k\; + 1$ by adding a multiple of $h_k\,(x)$ to predictor f_k:

$$f_k\; + 1(x) = f_k\,(x) + \lambda_k\, h_k\,(x) \tag{2.15}$$

For instance, when the modified squared loss function:

$$L(f_k(x), y) = \sum_{i=1}^{N} \frac{1}{2}(f_k(x_i) - y_i)^2 \tag{2.16}$$

is used in the case of the regression, each component of the gradient is given by:

$$h_k(x_i) = -\frac{\partial\left[\frac{1}{2}(f_k(x_i) - y_i)^2\right]}{\partial f_k(x_i)} = y_i - f_k(x_i) = r_i \tag{2.17}$$

which is the pseudo residual. We then train a base learner $h_k(x)$ with the pseudo residual dataset $\{(x_1, r_1), \ldots, (x_N, r_N)\}$, where the base learner can be any non-linear predictor, e.g., a small decision tree, to update the predictor $f_k(x)$ by adding a multiple of the base learner.
The overall algorithm that applies to both regression and classification with their respective loss functions is as follows:

ALGORITHM 2: THE CONSTRUCTION OF GRADIENT BOOSTING CLASSIFIER

1) **Input**: Data $\{(x_i, y_i): 1 \leq i \leq N\}$ and a loss function $L(y, f(x))$
2) Initialize the predictor with a constant value $f_0(x)$:

$$f_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma) \tag{2.18}$$

3) At step k where the predictor is $f_k(x)$, calculate the pseudo residuals:

$$r_i = -\frac{\partial L(f_k(x_i), y_i)}{\partial f_k(x_i)} \tag{2.19}$$

4) Train a base learner $h_k(x)$ with the pseudo residual dataset $\{(x_1, r_1), \ldots, (x_N, r_N)\}$:

$$\lambda_k = \underset{\lambda}{\operatorname{argmin}} \sum_{i} L(f_k(x_i) + \lambda h_k(x_i), y_i) \tag{2.20}$$

5) Optimize Step lengths.
6) Update the predictor

$$f_{k+1}(x) = f_k(x) + \lambda_k h_k(x) \tag{2.21}$$

7) Redo steps 3 to 6 until a stopping condition is met.

Typically, steps 3 to 6 are iterated from $k = 1$ to some prefixed value K (for instance $K = 100$) since it is sometimes difficult to satisfy fixed stopping conditions such as residuals or updates in the predictors being sufficiently small.

Gradient boosting stands out in predictive modeling due to several key attributes. Primarily, its exceptional precision sets it apart, often outperforming other models in delivering accurate predictions [45]. Additionally, this technique showcases remarkable efficiency, notably in training, even with vast and intricate datasets, making it a preferred choice for handling data-intensive projects [46]. Another distinct advantage lies in its robust support for managing categorical features, an essential capability in real-world scenarios characterized by diverse datasets. Furthermore, its competence in handling missing values within data adds to its appeal, addressing a common challenge encountered in data analysis and predictive modeling [47].

2.3.6 Artificial Neural Networks (ANN Tree)

Artificial Neural Networks (ANN) are computational systems that consist of many simple processors called neurons or perceptrons and are designed to work similarly to neurons in the biological brain [48]. The human brain contains about 100 billion interconnected neurons that process sensory information from the environment. Every single neuron processes the input's activities. If a particular action potential is reached, the neuron fires through its single output to all the neurons to which it is connected.

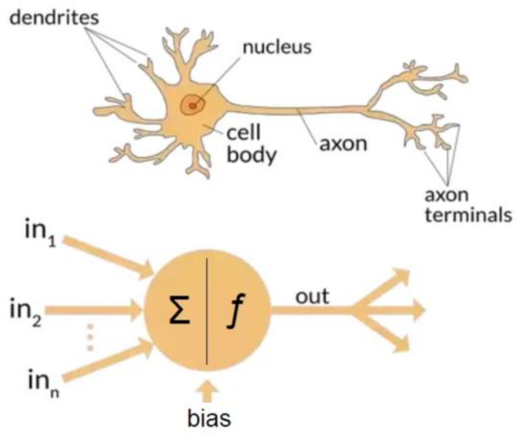


Figure 2.6 An illustration of a biological and artificial neurons [49]

Each artificial neuron takes a vector of inputs x and processes the inputs by taking their weighted sum with a vector of weights w, adding a bias b, and then applying an activation function f. The complete equation for the neuron's output is written as:

$$Output = f\left(\sum_{i} w_i x_i + b\right) \tag{2.22}$$

The activation function introduces non-linearity into the output of a neuron. This allows the neural network to learn non-linear representations from the training data. The most used activation functions are the sigmoid, the tangent hyperbolic (Tanh), and the rectified linear unit (ReLu).

The typical ANN architecture is the Feed-Forward neural network. Figure 2.6 shows an example of a feed-forward neural network [49]. In this framework, the neurons are arranged in three layers: an input layer, a few hidden layers in which all the processing is performed, and an output layer. The flow of information takes place sequentially from the input layer to the hidden layers and finally to the output layer, which computes the final output [50].

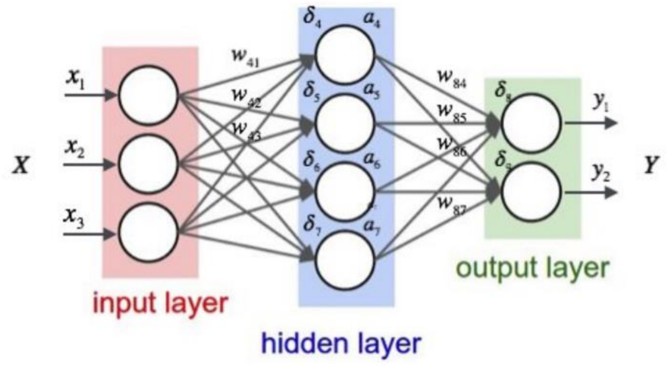


Figure 2.7 An illustration of a neural network [49]

2.3.6.1 Learning Process and Back-propagation

The goal of the learning (or training) process is to find a set of parameters for the weights and the bias that result in the best possible performance of the ANN for the problem at hand (classification in our context). In supervised learning, the actual output y of a particular input x is given and can be used to update the parameters w. The idea in this approach is to start with a random initialization of the weights and calculate the output for a given input. The error between the generated output and the actual output provided by a loss function is used to update the network weights using a gradient descent algorithm to minimize the error. The technique is called back-propagation since the weights are first corrected for the output layer and then propagated backward into the network [51]. The cross-entropy loss function is commonly used for classification tasks.

Once the gradient $\nabla_w L$ of the loss function L with respect to the weight vector w is evaluated, the gradient descent algorithm changes the weights in the negative direction of $\nabla_w L$ in such a way that the loss approaches closer to the minimum in each iteration. A local minimum is expected to be reached after several iterations [52]. Equation 2.23 shows the corresponding update rule of gradient descent, where the learning rate parameter defines how quickly the minimum should be approached. If the learning rate is too high, there is a risk that the minimum cannot be reached because the steps taken are too big and will be overshot. If it is bigger, the learning will take a long time to get the desired accuracy.

$$w_{k+1} = w_k - \eta \nabla_w L(w_k) \tag{2.23}$$

Neural networks can self-learn and produce output that is not limited by the input they receive. They can store the information of an entire network. However, the model requires fine-tuning several hyper-parameters to function efficiently, and inappropriate scaling of features might negatively impact its performance. Overfitting is another widespread problem in neural networks, where the model learns the details of training data so well that it fails to generalize to unseen data. To overcome this problem, regularization and careful feature selection are performed. Similarly, data splitting and cross-validation are used to train and test the base classifiers to avoid overfitting.

2.4 Random Forest Feature Importance (RF-FI)

Collecting data is an essential part of any analysis. However, it can lead to noise that may obscure the patterns and essential information that the data contains. Fortunately, dimensionality reduction is a powerful technique that can help identify and eliminate redundant features, thus reducing noise. When it comes to creating classification rules, random forests use the importance of features

to identify the significance score for each feature [55]. This score is calculated and used to determine the feature's importance. By analyzing the Gini relevance value of a feature in a single tree, we can calculate the overall importance of all the trees in the forest. With this information, we can easily determine the significance of each feature using Gini importance in a random forest model consisting of 100 decision trees [56]:
For each feature, j, calculate the total decrease in Gini impurity (TDI) at each node m of each decision tree i using the following formula:

$$TDI(j,m,i) = A \cdot (B - C \cdot D - E \cdot F) \tag{2.24}$$

Where:
• A: Proportion of samples that reach node m of tree i compared to the total number of samples n_i.

$$\mathrm{A} = \frac{n_{m,i}}{n_i} \tag{2.25}$$

•B: Gini impurity score at node m of tree i.

$$B = Gini(m,i) \tag{2.26}$$

• C: Proportion of samples that go to the left child node of node m of tree i compared to the samples at node m.

$$C = \frac{n_{left,m,i}}{n_{m,i}} \tag{2.27}$$

• D: Gini impurity score at the left child node of node m of tree i.

$$D = Gini(left,m,i) \tag{2.28}$$

• E: Proportion of samples that go to the right child node of node m of tree i compared to the samples at node m.

$$E = \frac{n_{right,m,i}}{n_{m,i}} \tag{2.29}$$

• F: Gini impurity score at the right child node of node m of tree i.

$$F = Gini(right,m,i) \tag{2.30}$$

For each decision tree i, calculate the total decrease in Gini impurity over all the nodes where each feature is used to split, as follows:

$$TDI(j,i) = \sum_{m} TDI(j,m,i) \tag{2.31}$$

where the sum is taken over all nodes m where feature j is used to split in tree i. For each feature j and each decision tree i, calculate the importance of feature j in tree i as the ratio of its total decrease in Gini impurity to the sum of the total decline in Gini impurity overall features, using the following formula:

$$FI(j,i) = \frac{TDI(j,i)}{\sum_{j'} TDI(j',m,i)} \tag{2.32}$$

Average the feature importance scores overall decision trees to obtain the mean importance of each feature, as follows:

$$FI(j) = \frac{1}{100}\sum_{i=1}^{100} FI(j,i) \tag{2.33}$$

Normalize the mean importance scores by dividing each score by the sum of all the mean importance scores, using the following formula:

$$FI'(j) = \frac{FI(j)}{\sum_{j'} FI(j')} \tag{2.34}$$

where $FI'(j)$ is the normalized importance score for feature j. The Gini importance scores, denoted as $FI'(j)$, represent the resulting values for each feature. A higher score indicates a greater significance of the feature. These scores are valuable in ranking the features and selecting the most crucial ones for further analysis or model development. One of the notable contributions of this thesis is the comprehensive feature selection process, which aims to identify the essential features within the dataset. We utilized the Random Forest built-in Feature Importance (RF-FI) technique to achieve this. By employing RF-FI, we generated a new feature set that contains fewer features yet provides more informative insights. This approach effectively reduced dimensionality and improved model execution time without compromising accuracy. To determine the most critical features, we set a threshold for RF-FI, resulting in the selection of 7 features for each model. We utilized the outcomes of this selection to create a new set of features. Subsequently, we retrained our models using these feature sets and compared the results to identify the optimal features regarding accuracy and execution time.

2.5 Iterative Imputer

Iterative imputation is a highly sophisticated data preprocessing technique that tackles missing values in datasets. Unlike conventional methods that rely on simple statistics such as mean or median values to fill in missing data, iterative imputation takes a more intricate approach. By considering all features present in the dataset, it estimates missing values iteratively. This technique creates a predictive model for each feature with missing data, based on the other available features. The imputer begins by estimating missing values for one feature, using the remaining features as predictors. This process continues iteratively across all the features until convergence, refining its estimates with each iteration. By incorporating complex modeling strategies such as regression, decision trees, or other machine learning algorithms, iterative imputation captures intricate relationships between variables, enhancing the accuracy of the imputed values. It adapts to the data structure and considers interdependencies between variables to create more precise estimations for missing values. The iterative nature of this technique ensures a more refined and nuanced approach to handling missing data, which contributes to improved downstream analysis and modeling processes. The algorithm 3 explains the process for iterative imputation in detail [57]:

ALGORITHM 3: ITERATIVE IMPUTER ALGORITHM FOR MISSING DATA IMPUTATION

```
1)  Inputs:
        • X: incomplete dataset with missing values
        • Estimator: regression model used to estimate missing values
        • max-iter: maximum number of iterations
2)  Outputs:
        • Xcomplete: Complete the dataset with imputed missing values.
    Steps:
    a)  Initialize missing values in X using a simple imputation method (e.g., mean, or median).
    b)  Repeat until convergence or maximum number of iterations:
            • For each feature i with missing values:
        -   Define Xobs as the subset of X where feature i is observed.
        -   Define Xmiss as the subset of X where feature i is missing.
        -   Use an estimator to estimate missing values in Xmiss based on observed values in Xobs.
        -   Replace missing values in Xmiss with the estimated values.
3)  Return the completed dataset Xcomplete.
```

Chapter 3: Methodology

This research methodically details the approach used to predict chronic kidney disease (CKD) rates employing Machine Learning algorithms. The methodology encompasses comprehensive stages, including data collection, preprocessing, selection of Machine Learning algorithms, and putting together an ensemble model that aggregates the predictions of the selected models.

3.1 Data Collection

The research was conducted using the CKD dataset [58]. The output column "class" has a value of either "0" or "1". The value "0" indicates that the person is not a CKD patient, while the value "1" shows that the person is a CKD patient. Figure 3.1 displays the total number of CKD and non-CKD entries in the output column. The overall number of CKD data is 250, whereas the total number of non-CKD data is 150.

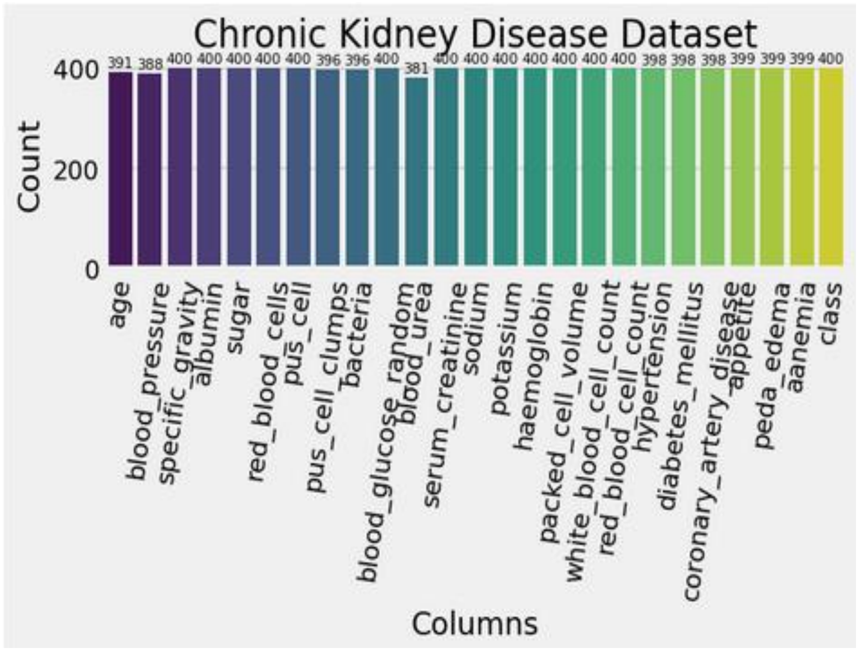


Figure 3.1 Data distribution of Chronic Kidney Disease Dataset

Feature attributes for CKD dataset:

Attribute	**Meaning**	**Category**	**Scale**
age	Age	Numerical	Years
bp	Blood pressure	Numerical	mm/Hg
sg	Specific gravity	Nominal	1.005 to 1.025
al	Albumin	Nominal	0 to 5

Attribute	Meaning	Category	Scale
su	Sugar	Nominal	0 to 5
rbc	Red blood cells	Nominal	Abnormal, Normal
pc	Pus cell	Nominal	Abnormal, Normal
pcc	Pus cell clumps	Nominal	Not present, Present
ba	Bacteria	Nominal	Not present, Present
bgr	Blood glucose random	Numerical	mgs/dl
bu	Blood urea	Numerical	mgs/dl
sc	Serum creatinine	Numerical	mgs/dl
sod	Sodium	Numerical	mEq/L
pot	Potassium	Numerical	mEq/L
hemo	Hemoglobin	Numerical	gms
pcv	Packed cell volume	Numerical	Pcv
wc	White blood cell count	Numerical	cells/cumm
rc	Red blood cell count	Nominal	millions/cmm
htn	Hypertension	Nominal	No, Yes
dm	Diabetes mellitus	Nominal	No, Yes
cad	Coronary artery disease	Nominal	No, Yes
appet	Appetite	Nominal	Poor, Good
pe	Peda edema	Nominal	No, Yes
ane	Anemia	Nominal	No, Yes
Classification	Class	Nominal	Not CKD, CKD

Table 3.1 Data Description for Chronic Kidney Disease Dataset

3.2 Data Preprocessing

Data preprocessing was conducted to ensure the dataset's suitability for Machine Learning. The following steps were performed:

1) **Handling Missing Data:** As explained in 2.3, we employed iterative imputer for imputing missing data.
2) **Outlier Detection:** Our next step involved the identification of outliers. We created visualizations, including Std plots for all dataset columns, to detect potential outliers.

3) **Feature Scaling:** To mitigate biases in our Machine Learning algorithms, we standardized numerical features, ensuring they had a mean of zero and a variance of one.
4) **Feature Selection:** We plotted a correlation heatmap and employed the RF-FI algorithm to select the best subset for predicting CKD.
5) **One-Hot Encoding:** Categorical variables underwent one-hot encoding to transform them into a numerical format, facilitating processing by Machine Learning models.
6) **Data Splitting**: We partitioned the dataset into training and testing sets using random split, facilitating practical model training and evaluation.
7) **Cross-validation:** In the final stage, we perform 10-fold cross-validation on CKD data to evaluate the effectiveness of the machine learning proposed model.

3.3 Model Training and Evaluation

The selected Machine Learning algorithms are trained and evaluated using the following methodology:

1) **Training**: Each algorithm was trained on the training dataset using default hyper-parameters.
2) **Hyper-parameter Tuning:** Hyper-parameter tuning was performed using a grid search technique to optimize the algorithms' performance.
3) **Evaluation Metrics**: To assess their predictive performance, the models were evaluated using standard classification metrics such as accuracy, precision, recall, F1-score, and ROC-AUC curve.
4) **Visualization Integration**: During the evaluation process, data visualization techniques were employed to gain insights into model behavior, feature importance, and decision boundaries.

3.4 Hyper-Parameter Tuning

Finding the optimal hyper-parameters for a given task can be difficult as the perfect settings cannot be predetermined. To overcome this challenge, we tested various hyper-parameter combinations on the training set and assessed the performance of each model. However, this method can result in overfitting, where the model performs well on the training data but needs to be generalized to new data. To address this, we employed cross-validation, a critical technique in hyper-parameter tuning. We utilized K-fold cross-validation to divide the data into training and testing sets and split the training set into K subsets or folds. The model was then trained iteratively K times, with each iteration using K-1 of the folds and evaluating the remaining fold. The average performance of each fold was used to determine the final validation metrics for the model. To prevent bias in assessing the model's performance, we utilized stratified K-fold cross-validation, which is ideal for classification problems [59]. Stratified K-fold ensures that each cross-validation fold has a similar proportion of each class as the entire dataset. We evaluated various folds, including 3-, 4-, 5-, and 10-folds, using stratified sampling set to the output 'FLAG' variable and assessed different hyper-parameters. We used grid search to identify the best parameter values based on each model's context. Additionally, we applied various configurations to the k-fold cross-validation, including 3-, 4-, 5-, and 10-fold, using the same stratified parameter. The results of these evaluations are presented in the results section and were primarily used to identify any potential overfitting or underfitting issues in our models. Following these steps ensured that our dataset was appropriately prepared for model training and testing, enabling our models to detect illicit accounts accurately while minimizing false positives.

3.5 Data Visualization Techniques

Various techniques were applied during the evaluation phase to leverage the potential of data visualization in improving the interpretability and transparency of machine learning models. These visualization methods are designed to provide insights into the behavior of the models, the importance of features, and the decision boundaries. The following data visualization techniques were employed:

3.5.1 Confusion Matrices

Confusion matrices are used to visualize the performance of classification models. They provide insights into the number of true positives, true negatives, false positives, and false negatives [60]. Visualizing these metrics helps us understand how well the model performs and where it may need improvement. Figure 3.6 shows the confusion matrix. The confusion matrix rates the performance of machine learning classification models. All models were evaluated using the confusion matrix. The confusion matrix illustrates how often our models guess correctly and incorrectly. Poorly predicted values received false positives and negatives, whereas correctly predicted values received genuine positives and negatives. The model's accuracy, precision-recall trade-off, and AUC were assessed after grouping all predicted values in the matrix.

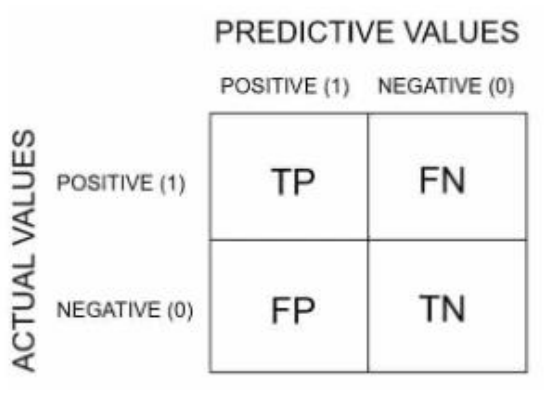


Figure 3.2 Confusion Matrix [60]

3.5.2 ROC-AUC Curves

Receiver Operating Characteristic (ROC) curves and the Area Under the Curve (AUC) score are valuable tools for visualizing the trade-off between actual positive rate and false positive rate [61]. These curves provide insights into the model's ability to distinguish between different classes, especially in imbalanced datasets.

3.6 Proposed Model

Our model is based on machine learning and uses the advantages of strategies employed in the ensemble learning framework. The main strategies of ensemble learning, which are bagging, boosting, and stacking, as explained in section 2.1, have been incorporated into our model as follows: we used a basic decision tree model as a base learner explained in subsection 2.2.1 and

applied bagging to build a random forest classifier. For boosting, we built two classifiers, Adaboost, and Gradient-boosted trees classifiers. In addition, we included two complementary models, Support Vector Machine (SVM) and Artificial Neural Network (ANN), based on strategies different from ensemble learning. We trained and tested all three classifiers, along with SVM and ANN, individually on the data using the basic data splitting procedure. Finally, we used a stacking strategy to combine the five aforementioned classifiers using ensemble vote classifier based on the majority rule voting and resampling strategy similar to the 10-fold cross-validation to train and test each of the five ensemble components of the stacked model. We decided to include two classifiers based on the boosting strategy in the stacked ensemble due to their good performance in many prediction applications.
The resampling procedure used in our model can be described as follows:

- We split randomly the dataset into 10 equally sized folds.
- Then, we repeat the following for each of the 10 folds and each base model:
  - we train the model on the 9 remaining folds
  - test the model on the selected fold

This resampling strategy allows us to obtain a more accurate estimate of each base model's prediction performance since it ensures that the whole dataset is used for both training and testing.

The *Ensemble Vote Classifier* is a meta-classifier for combining similar or different machine learning classifiers and aggregate their output into a final prediction using the majority or plurality voting. In the general context, the final class label $S(x_i)$ output by the stacked model S

$$S(x_i) = mode\{M_1(x_i), \ldots, M_K(x_i)\}, \tag{3.1}$$

for a feature vector x_i in the dataset is given by the mode of the distribution of the class labels as where $M_k(x_i)$ stands for the class label assigned to x_i by the base model M_k, and K is the number of the base models used in the stacking.

Chapter 4: Experimentals

In Chapter 4, we conducted experiments to predict CKD rates using various machine-learning algorithms and visualization techniques. The aim was to evaluate their effectiveness and potential benefits for decision-making.

4.1 Data Pre-Processing

1) **Handling Missing Data:** The heatmap in Figure 4.1 represents missing values, with 'yellow lines' indicating the absence of data.

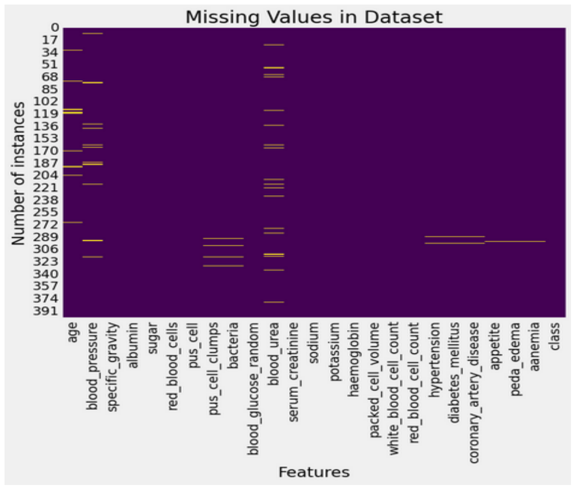


Figure 4.1 Heatmap showing Missing values

We utilized iterative imputation to fill both numerical and categorical missing values, resulting in more accurate data. The dark background in Fig 4.2 indicates the successful imputation of all missing data points.

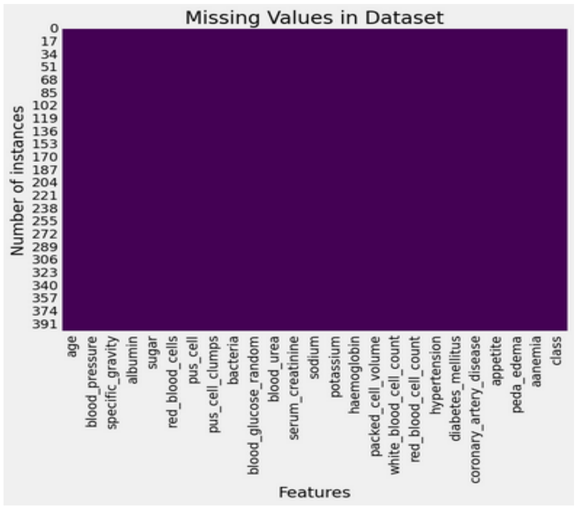


Figure 4.2 Heatmap after Missing value Imputation

2) **Outliers:** We proceeded to check for outliers in our dataset by plotting the standard deviation for all columns. Figure 4.3 displays a scatter plot showing the standard deviation for all columns. This plot indicated outliers in the White Blood Cell (WBC) count. To confirm this, we referred to Figure 4.4, which presents a box plot specifically for WBC.

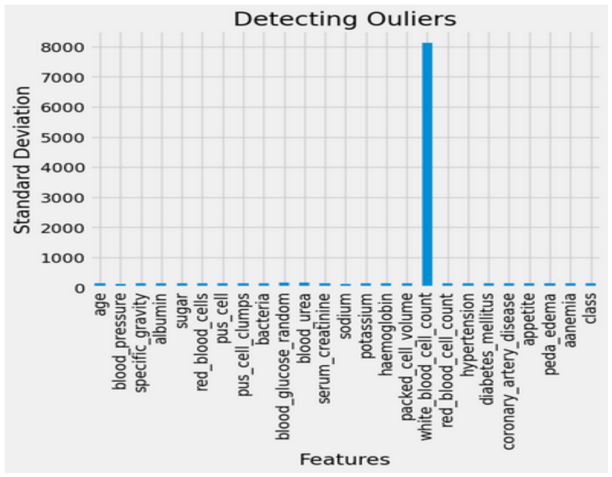


Figure 4.3 Standard Deviation Plot for CKD Dataset

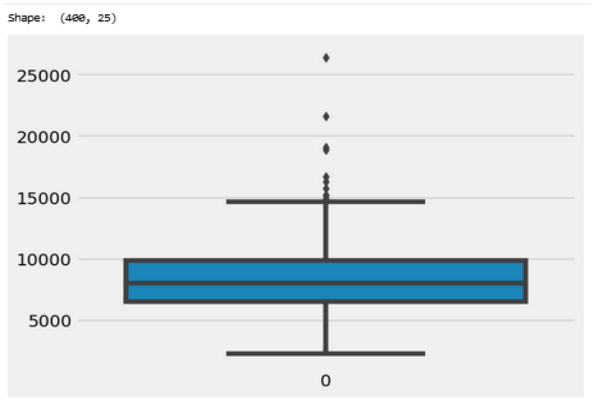

```
Shape:  (400, 25)
```

Figure 4.4 Box plot for WBC Count Confirming Outliers

For handling outliers, we used the Inter Quantile Range (IQR) [62], defining the upper and the lower bound:

$$upper = Q3 + 1.5 * IQR \tag{4.1}$$

$$lower = Q1 - 1.5 * IQR \tag{4.2}$$

In the above formula, according to statistics, we used 1.5 times the IQR to define the upper and lower bounds. This allows us to consider all data points within 2.7 standard deviations of the Gaussian Distribution. Fig 4.5 shows the data frame after removing outliers.

```
New Shape:  (390, 25)
<Axes: >
```

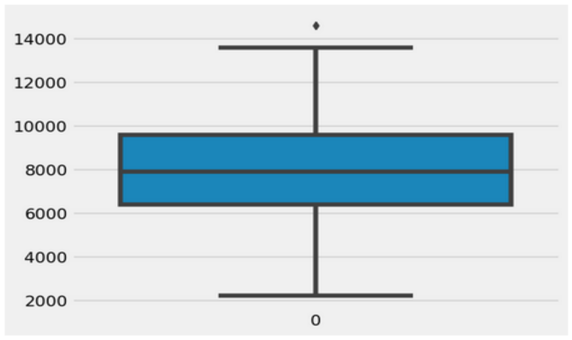


Figure 4.5 Box Plot for WBC Count after Handling Outliers

3) **Feature Scaling:** Using {MinMaxScaling}, we scaled and translated each feature individually to be in the given range on the training set, e.g., between zero and one, as shown in Fig 4.6.

```
[[0.52272727 0.23076923 0.75       ... 0.         0.         0.        ]
 [0.05681818 0.         0.75       ... 0.         0.         0.        ]
 [0.68181818 0.23076923 0.25       ... 0.5        0.         0.5       ]
 ...
 [0.11363636 0.23076923 0.75       ... 0.         0.         0.        ]
 [0.17045455 0.07692308 1.         ... 0.         0.         0.        ]
 [0.63636364 0.23076923 1.         ... 0.         0.         0.        ]]
```

Figure 4.6 New Features showing Zero Mean and Unit Variance

4) **Feature Selection**:

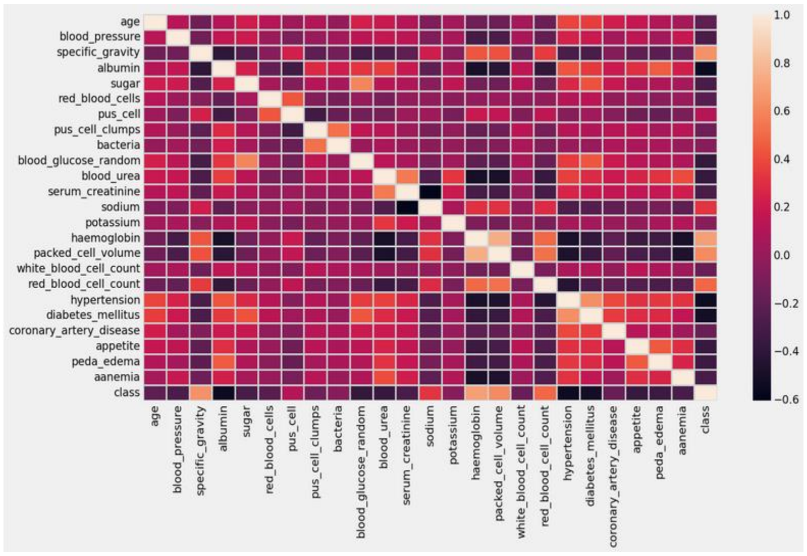


Figure 4.7 Correlation Heatmap

➢ **Feature-Class Correlation:**
We also assessed the correlation between individual features and the target variable, the 'class' column. We considered features with higher absolute correlations with the 'class' column to be more critical for our CKD prediction models. We observed that some features

displayed notable correlations with the 'class' column, suggesting their potential significance in distinguishing CKD patients from non-CKD cases.

- **RF-FI score:**
  To streamline our feature selection process, we implemented a threshold for the RF-FI score as explained in section 2.3. This allowed us to identify the top features based on their essential scores while ensuring that the number of features remained manageable. Our threshold value of 0.03 was chosen to include the top seven features, as we observed significant discrepancies in accuracy beyond this value. This is depicted in Figure 4.8.

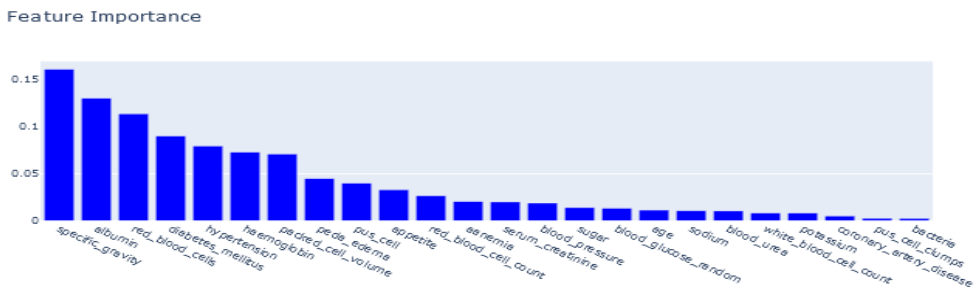
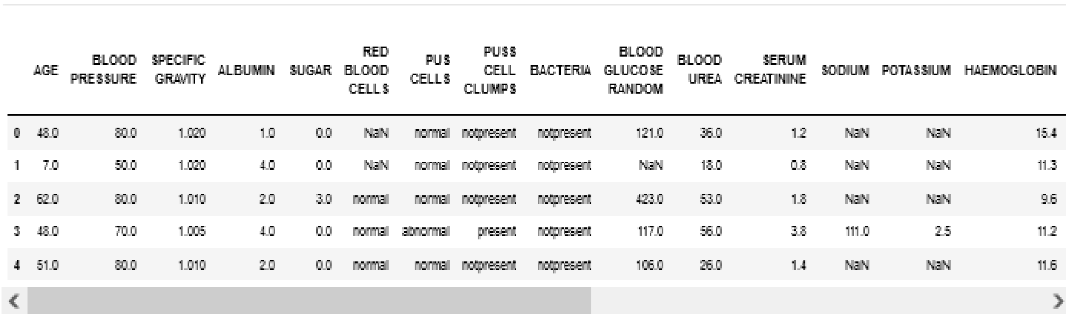


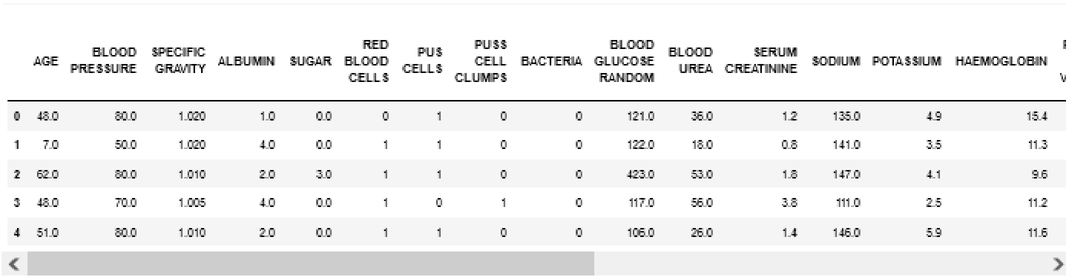
Figure 4.8 Important Features (RF-FI score)

5) **One-Hot Encoding**: One-hot encoding transforms categorical variables into numerical format. Fig 4.9 displays the data frame pre-encoding, while Fig 4.10 displays the data frame post-encoding.



	AGE	BLOOD PRESSURE	SPECIFIC GRAVITY	ALBUMIN	SUGAR	RED BLOOD CELLS	PUS CELLS	PUSS CELL CLUMPS	BACTERIA	BLOOD GLUCOSE RANDOM	BLOOD UREA	SERUM CREATININE	SODIUM	POTASSIUM	HAEMOGLOBIN
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	36.0	1.2	NaN	NaN	15.4
1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	18.0	0.8	NaN	NaN	11.3
2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	53.0	1.8	NaN	NaN	9.6
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	56.0	3.8	111.0	2.5	11.2
4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	26.0	1.4	NaN	NaN	11.6

Figure 4.9 Dataset before encoding



	AGE	BLOOD PRESSURE	SPECIFIC GRAVITY	ALBUMIN	SUGAR	RED BLOOD CELLS	PUS CELLS	PUSS CELL CLUMPS	BACTERIA	BLOOD GLUCOSE RANDOM	BLOOD UREA	SERUM CREATININE	SODIUM	POTASSIUM	HAEMOGLOBIN
0	48.0	80.0	1.020	1.0	0.0	0	1	0	0	121.0	36.0	1.2	135.0	4.9	15.4
1	7.0	50.0	1.020	4.0	0.0	1	1	0	0	122.0	18.0	0.8	141.0	3.5	11.3
2	62.0	80.0	1.010	2.0	3.0	1	1	0	0	423.0	53.0	1.8	147.0	4.1	9.6
3	48.0	70.0	1.005	4.0	0.0	1	0	1	0	117.0	56.0	3.8	111.0	2.5	11.2
4	51.0	80.0	1.010	2.0	0.0	1	1	0	0	106.0	26.0	1.4	146.0	5.9	11.6

Figure 4.10 Dataset after encoding

6) **Data Splitting**: One of the techniques for splitting data is the random split method. This approach randomly divides a dataset into two subsets: the training and test sets. Typically, an 80:20 ratio is used for training and testing, respectively. The randomness of this method helps

mitigate any potential bias in the data. Figure 4.11 below visually represents the random split method for data.

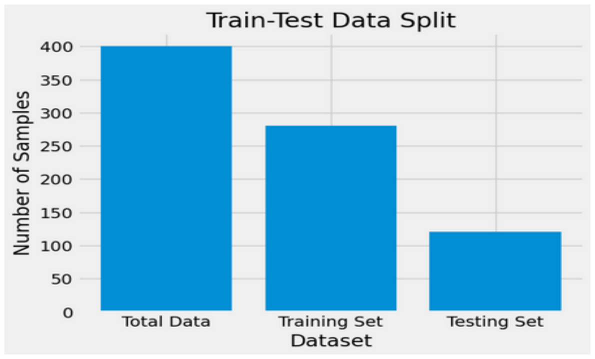


Figure 4.11 Train and Test Dataset

7) Cross-Validation: Figure 4.12 shows the plot for cross-validation of our dataset. Since our dataset is small, we ran cross-validation to assess its reliability.

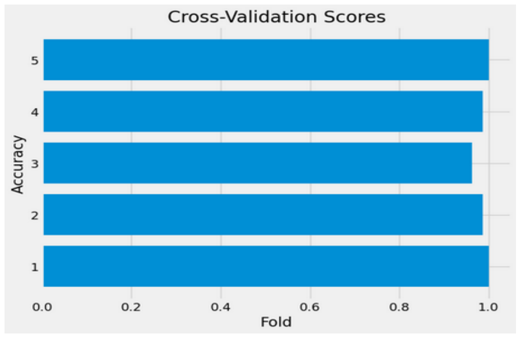


Figure 4.12 Barh plot showing cross-validation scores

4.2 Evaluation Metrics

The predictive performance of the Machine Learning models was assessed using the following evaluation metrics [63]:

1) **Accuracy**: Measures the overall correctness of predictions.

$$Ac = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.3}$$

2) **Precision**: Evaluate the positive predictive value.

$$Precision = \frac{TP}{TP + FP} \tag{4.4}$$

3) **Recall**: Measures the ability to identify actual positive cases.

$$Recall = \frac{TP}{TP + FN} \tag{4.5}$$

4) **F1-Score**: Balances precision and recall.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4.6}$$

5) **ROC-AUC Curve/Score**: Assesses the model's ability to distinguish between CKD and non-CKD cases.

Here, Ac refers to accuracy. TP, FP, FN, and TN represent true positive, false positive, false negative, and true negative.

4.3 Exploratory Data Analysis (EDA)

Using data visualization techniques, we conducted experiments to explore our model's behavior, feature importance, and decision boundaries. We used facet grids and violin plots to uncover trends and patterns in the dataset, focusing on the columns relevant to disease prediction. We created a Python function to generate Facet Grid plots for Kernel Density Estimation (KDE) to visualize the columns and segregate data into CKD and non-CKD cases. This approach allowed us to scrutinize how CKD and non-CKD classes exhibited different distributions within the columns. We also created another Python function to generate violin plots for the specified column to analyze the volume. Each violin plot features a central box plot that provides valuable statistical insights. The surrounding area displays density estimation, further motivating our decision to proceed with our machine-learning algorithms. We limited our EDA figures to focus solely on the columns pertinent to disease prediction, including red blood cell count, white blood cell count, packed cell volume, hemoglobin, albumin, serum creatinine, and specific gravity. These visualizations, accompanied by our observations, gave us a more profound understanding of the dataset and its potential for our predictive modeling tasks. Lastly, all the plots show that 0 indicates patients with non-CKD, and 1 indicates patients with CKD.

4.3.1 Red blood cell count

A high red blood count refers to the number of cells produced in the bone marrow and found in the blood. The primary function of red blood cells is to transport oxygen from the lungs

to other parts of the body. Conditions that limit oxygen can increase the number of red blood cells. Other conditions may cause the body to produce more red blood cells than it needs [64].

The definition of high blood pressure varies between laboratories. The normal range for adults is 4.35 to 5.65 million red blood cells per microliter (mcL) of blood in men and 3.92 to 5.13 million red blood cells per microliter (mcL) of blood in women.

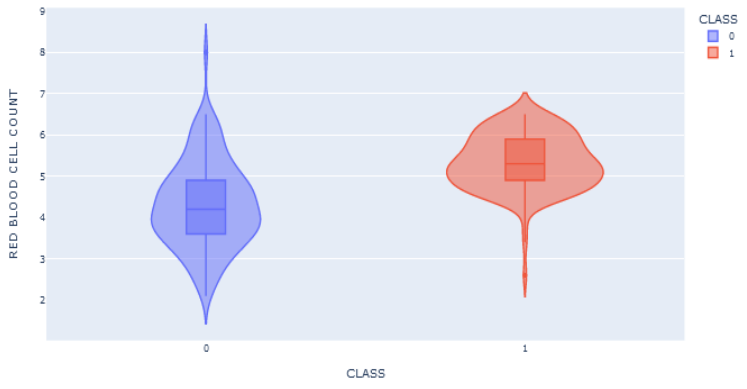


Figure 4.13 Violin Plot for Red Blood Cell Count

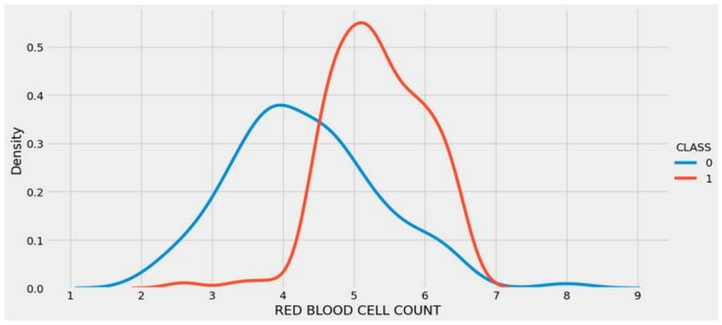


Figure 4.14 Facet grid plot for Red Blood Cell Count

Observations:

- The Facet Grid shows variations in Red Blood Cell (RBC) counts across different categorical variables, i.e., 0 and 1.
- The violin plot for non-CKD class (0) shows tremendous variation across the graph; incidentally, CKD (1) varies in a fixed range for most and has a higher density.

4.3.2 White blood cell count

Leukocytosis, or elevated white blood cells, can indicate many conditions, including infection, inflammation, injury, and immune system disorders [65]. A complete blood count (CBC) is usually performed to check for leukocytosis. The following treatments often reduce the number of white blood cells. Figures 4.15 and 4.16 show the WBC density and KDE calculated for our data.

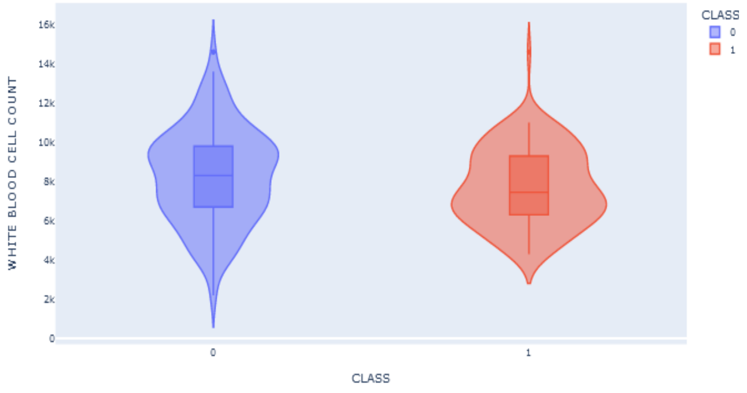


Figure 4.15 Violin plot for White blood cell count

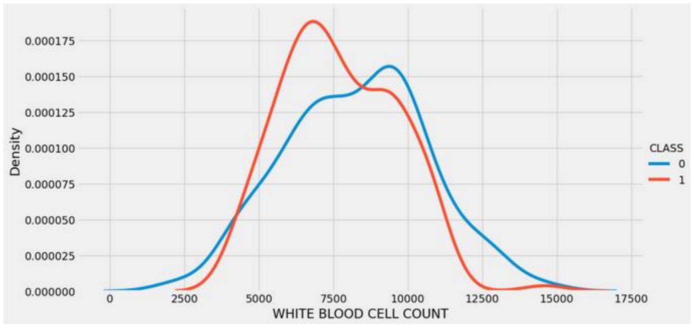


Figure 4.16 Facet grid plot for White Blood Cell Count

Observations:

- The Facet Grid exhibits the distribution of White Blood Cell (WBC) counts across distinct categorical variables, denoted as 0 and 1.
- The violin plot for the non-chronic kidney disease (CKD) class (0) displays significant variability throughout the graph. At the same time, the variation for CKD (1) remains within a consistent range for most data points, exhibiting higher density.

4.3.3 Haemoglobin

Hemoglobin (Hb) is a protein found in red blood cells that carries oxygen throughout the body and delivers red blood cells. Both levels vary from person to person. Levels are higher in men than in women. When donating blood, a hemoglobin "cut-off" level is set to ensure your hemoglobin does not fall below normal after donation. The average amount of heme varies between races, men, and women and is also affected by age, especially women. People with hemoglobin levels below normal are anemic. Diabetes has many causes and is thought to be caused by iron deficiency. [67]

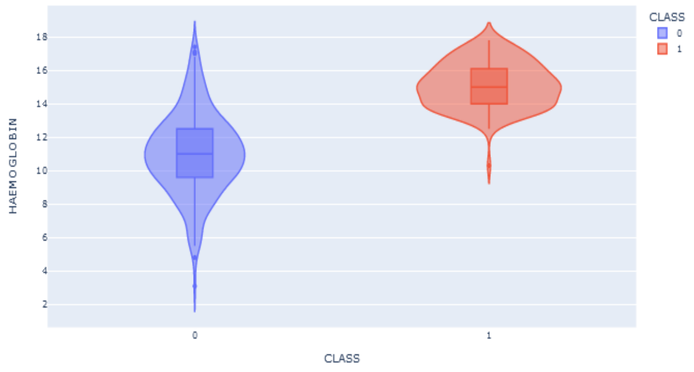


Figure 4.17 Violin plot for Haemoglobin

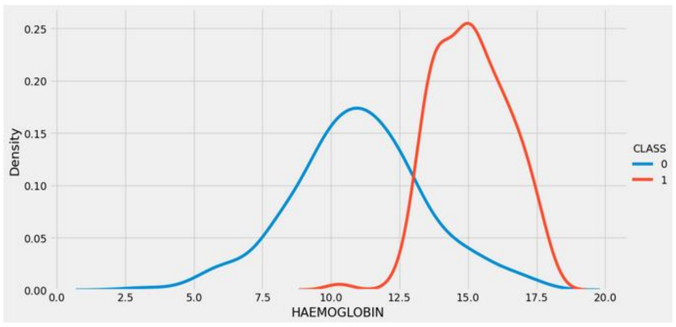


Figure 4.18 Facet Grid Plot for Haemoglobin

Observations:

- The Facet Grid for hemoglobin highlighted variations in hemoglobin levels across categories, i.e., non-ckd (0) and ckd (1).
- The Violin Plot for hemoglobin indicated the concentration and distribution of hemoglobin values, offering insights into any potential clustering or outliers.

4.3.4 Packed cell volume

Packed Cell Volume (PCV) testing is used to diagnose diabetes or polycythemia in patients. Complete blood tests are usually done to evaluate the need for blood transfusions and to monitor response to blood therapy [66]. Blood is a mixture of blood and cells. The PCV test measures the number of blood cells in the blood. If the PCV result shows a reading of 50%, 50 ml of cells are present in 100 ml of blood. If the number of RBCs (red blood cells) increases, the total PCV value will also increase. This number also increases due to dehydration.

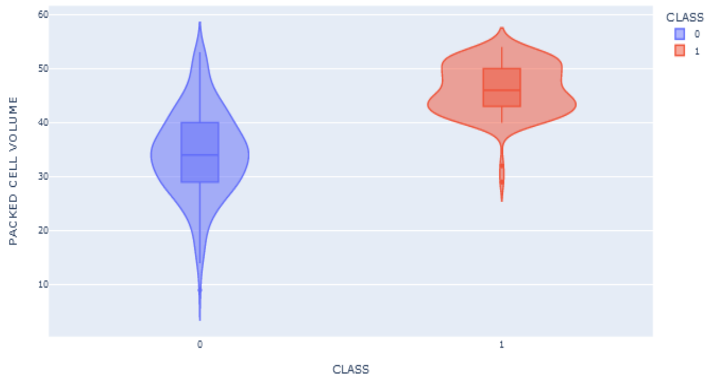


Figure 4.19 Violin plot for Packed Cell Volume

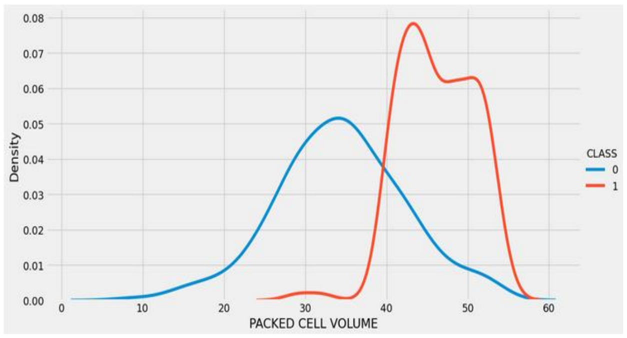


Figure 4.20 Facet Grid plot for Packed Cell Volume

<u>Observations:</u>

- The Facet Grid analysis illustrates the diverse Packed Cell Volume (PCV) levels concerning various categorical factors. It is a crucial tool to discern any substantial disparities in PCV across distinct categories, i.e., non-ckd(0) and ckd(1).
- Moreover, the Violin Plot visualization for PCV effectively portrays the distribution pattern of PCV values, highlighting potential asymmetry or the presence of multiple modes within the distribution, thus offering valuable insights into the data's distribution characteristics.

4.3.5 Albumin

The albumin test checks your liver and kidney function. Albumin is a protein found in plasma. Low albumin levels can be caused by kidney disease, liver disease, inflammation, or infection. High albumin levels are often caused by dehydration or severe diarrhea [68]. The albumin blood test measures the amount of albumin in the blood. Albumin is a protein found in plasma. Your liver produces albumin. Albumin protects fluid from the blood. It also helps vitamins, enzymes, hormones, and other substances in the body. If your doctor suspects your liver or kidneys are malfunctioning, they may order an albumin test.

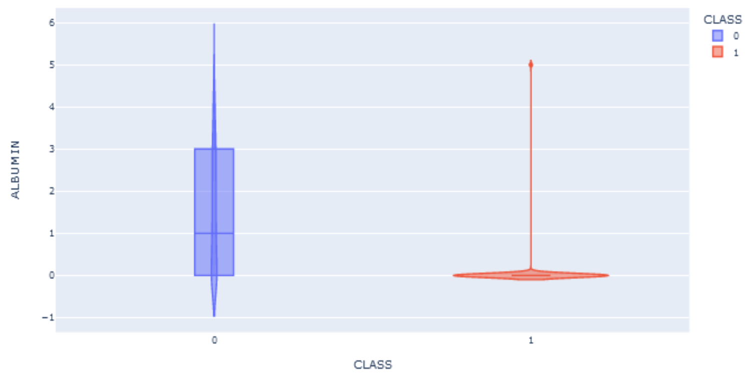


Figure 4.21 Violin Plot for Albumin

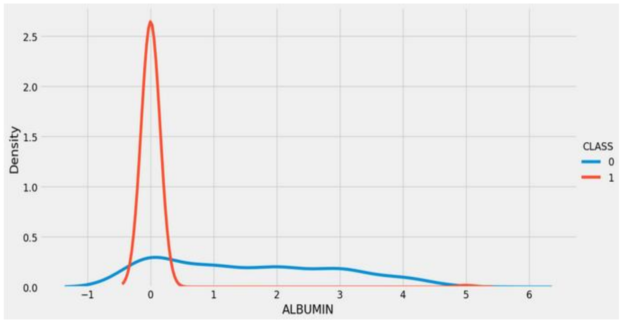


Figure 4.22 Facet Grid Plot for Albumin

<u>Observations:</u>

- Higher albumin levels indicate dehydration, which will drastically affect kidney function.
- As you can see in the plot, the albumin levels are higher in CKD than in non-CKD persons.
- The Facet Grid and Violin Plot for albumin helped us to identify variations in albumin levels across varied factors. This is crucial for understanding the distribution of this important biomarker.

4.3.6 Serum Creatinine

The creatinine test measures the kidney's ability to filter waste products from the blood. Creatinine is a byproduct of the energy production process in muscles. Healthy kidneys filter creatinine from the blood. Creatinine is eliminated as a waste product in the urine [69]. Measuring creatinine in your blood or urine can provide clues to help your doctor determine how well your kidneys are working. Low creatinine levels may also indicate that a person has chronic kidney disease, decreased kidney function, or malnutrition. You can find more information about low creatinine levels here. High creatinine levels can also indicate kidney problems such as infection or failure.

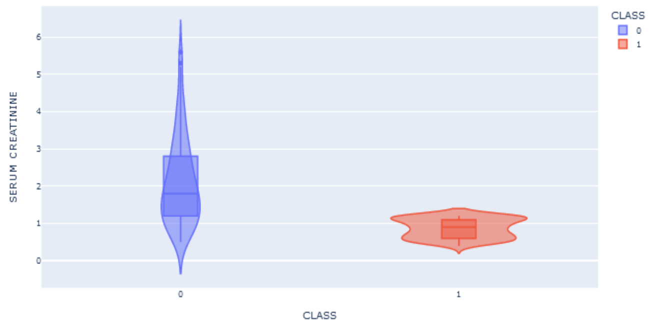


Figure 4.23 Violin plot for Serum Creatinine

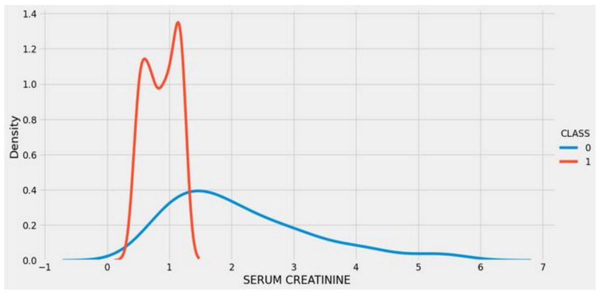


Figure 4.24 Facet Grid plot for Serum Creatinine

Observations:

- In the Violin plot, the KDE for Serum Creatinine is lower in Class 1 than in Class 0 due to the disease.
- The facet grid confirms this. As you can see, the serum levels for Class 1 vary between a fixed range and vice-versa.
- The Facet Grid and Violin Plot for serum creatinine revealed variations in serum creatinine levels across categories. It was valuable for detecting potential patterns.

4.3.7 Specific Gravity

Elevated specific gravity (SG) can indicate various conditions, including dehydration, kidney problems, and certain medical conditions [70]. A urinalysis is typically performed to check for abnormal specific gravity. Figures 4.25 and 4.26 show the SG density and KDE calculated for our data.

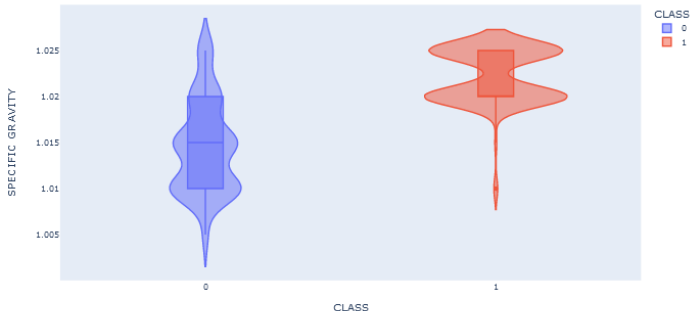


Figure 4.25 Violin plot for Specific Gravity

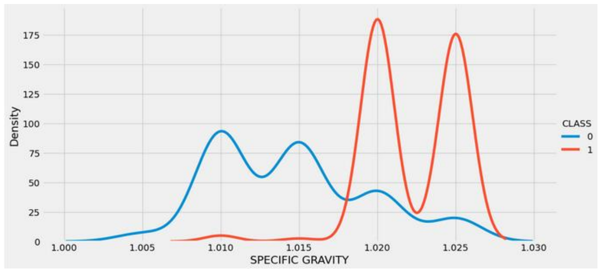


Figure 4.26 Facet grid plot for Specific Gravity

Observations:

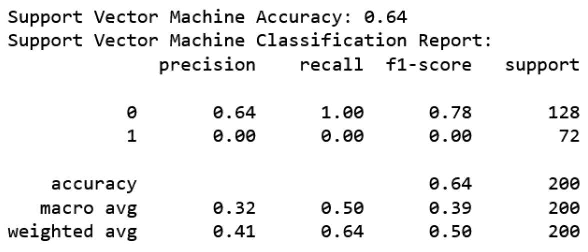
- The Facet Grid for Specific Gravity (SG) revealed variations across various categories, highlighting how specific gravity levels differ between groups.
- The Violin Plot for Specific Gravity (SG) revealed exciting patterns, such as variations in specific gravity levels within subgroups.

4.4 Training our component models before feature selection:

After performing exploratory data analysis (EDA), we employed five supervised classification machine learning algorithms: Support Vector Machine (SVM), Random Forest Bagging (RF Bagging), Gradient Boosting (GB), ADA Boost, and Artificial Neural Networks (ANN). By leveraging an ensemble learning technique, we stacked the models and subjected them to max voting to amalgamate their predictions. Moreover, we used a 10-fold cross-validation strategy to ensure a robust evaluation during the training and assessment phases. Our approach is designed to provide accurate and reliable predictions, and we are confident that our methodology has yielded results that can be trusted.

Support Vector Machine (SVM)

Figure 4.27 depicts the accuracy and classification report of Hunt's Decision Tree classifier before feature selection. Notably, the accuracy achieved in this case is 64 percent.


```
Support Vector Machine Accuracy: 0.64
Support Vector Machine Classification Report:
              precision    recall  f1-score   support

           0       0.64      1.00      0.78       128
           1       0.00      0.00      0.00        72

    accuracy                           0.64       200
   macro avg       0.32      0.50      0.39       200
weighted avg       0.41      0.64      0.50       200
```

Figure 4.27 Classification report for SVM

Moreover, examining the overall F1 score shows accuracy at 50 percent. Drilling into the specifics, individual F1 scores reveal 78 percent for non-chronic kidney disease (CKD) and 0 percent for CKD cases. These statistics highlight the classifier's proficiency in distinguishing between the two categories with a slight difference in the F1 scores. For a more detailed insight into the model's performance, Figure 4.28 presents the confusion matrix, which illustrates the model's predictions and their alignment with actual outcomes.

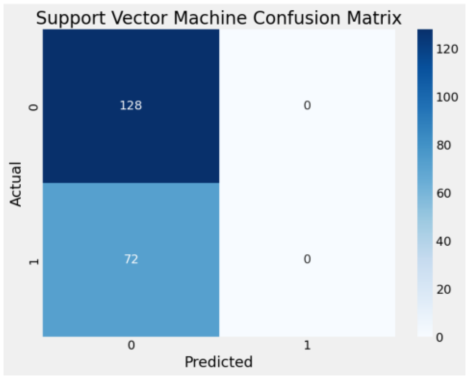
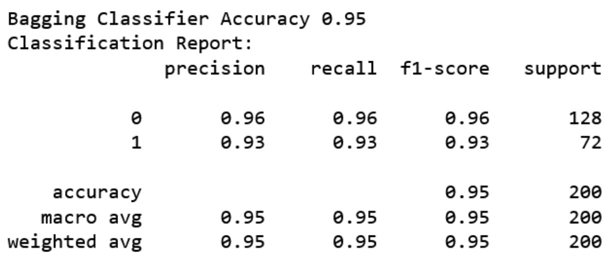


Figure 4.28 Confusion Matrix for SVM

Random Forest Bagging

In Figure 4.29, we delve into the accuracy and classification report of the Random Forest Bagging classifier. Notably, the classifier attains a commendable accuracy of 95 percent, demonstrating its effectiveness in distinguishing between the classes.



```
Bagging Classifier Accuracy 0.95
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       128
           1       0.93      0.93      0.93        72

    accuracy                           0.95       200
   macro avg       0.95      0.95      0.95       200
weighted avg       0.95      0.95      0.95       200
```

Figure 4.29 Classification Report for Random Forest Bagging

Furthermore, when examining the average F1 score, it aligns harmoniously with the accuracy at 95 percent. Drilling into the specifics, individual F1 scores, with 96 percent for non-chronic kidney disease (CKD) and 93 percent for CKD cases, underscore the classifier's precision in categorizing the data, with an excellent differentiation between the two classes. Figure 4.30 presents the confusion matrix; this matrix illustrates the model's predictions and their alignment with actual outcomes.

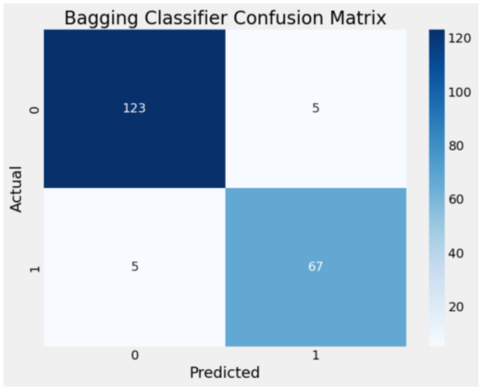
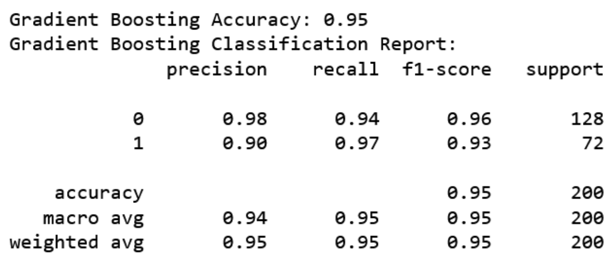


Figure 4.30 Confusion Matrix for Random Forest Bagging

Gradient Boosting

In Figure 4.31, we delve into the accuracy and classification report of the Gradient Boosting classifier. Notably, the classifier attains a commendable accuracy of 95 percent, demonstrating its effectiveness in distinguishing between the classes.



```
Gradient Boosting Accuracy: 0.95
Gradient Boosting Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.94      0.96       128
           1       0.90      0.97      0.93        72

    accuracy                           0.95       200
   macro avg       0.94      0.95      0.95       200
weighted avg       0.95      0.95      0.95       200
```

Figure 4.31 Classification Report for Gradient Boosting

Furthermore, when examining the average F1 score, it aligns harmoniously with the accuracy at 95 percent. Drilling into the specifics, individual F1 scores shine, with an impressive 96 percent for non-chronic kidney disease (CKD) and 93 percent for CKD cases. These results underscore the classifier's precision in categorizing the data, with an excellent differentiation between the two classes. For an even deeper understanding of the model's performance, Figure 4.32 presents the confusion matrix. This matrix offers a detailed view of the model's predictions and their alignment with actual outcomes.

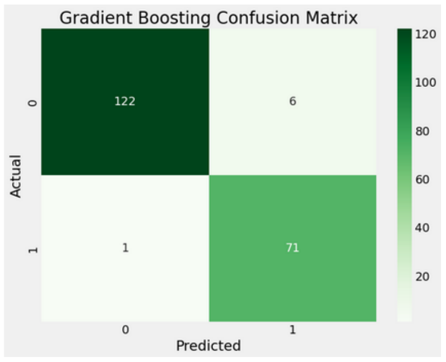
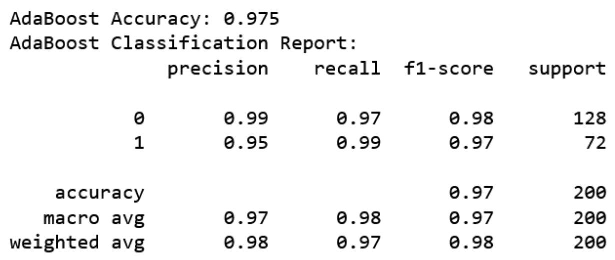


Figure 4.32 Confusion Matrix for Gradient Boosting

ADA Boost

In Figure 4.33, we present the results for the AdaBoost classifier, providing insights into its accuracy and classification report. The classifier demonstrates a solid performance with an accuracy rate of 97.5 percent, indicating its ability to classify the data effectively.



```
AdaBoost Accuracy: 0.975
AdaBoost Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       128
           1       0.95      0.99      0.97        72

    accuracy                           0.97       200
   macro avg       0.97      0.98      0.97       200
weighted avg       0.98      0.97      0.98       200
```

Figure 4.33 Classification Report for ADA Boost

Looking closely at the metrics, the average F1 score mirrors the overall accuracy at 98 percent, reaffirming the classifier's reliability in distinguishing between classes. When examining individual F1 scores, we observe commendable results, with a 98 percent score for non-chronic kidney disease (CKD) and a 97 percent score for CKD cases. These scores underscore the classifier's proficiency in classifying the data, emphasizing precision and recall. To gain a more comprehensive view of the model's performance, Figure 4.34 displays the confusion matrix, which presents a detailed summary of the model's predictions compared to the actual outcomes.

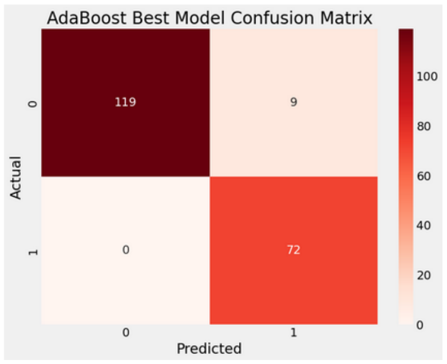
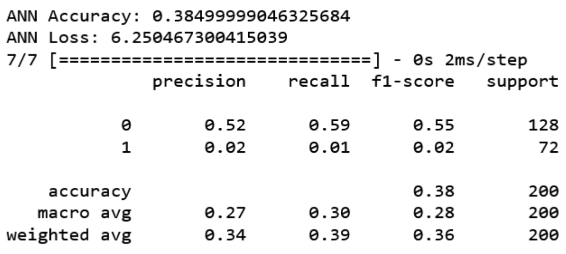


Figure 4.34 Confusion Matrix for ADA Boost

Artificial Neural Network (ANN)

In Figure 4.35, we present the results for the ANN classifier, providing insights into its accuracy and classification report. The classifier performs with an accuracy rate of 38.4 percent, indicating its ability to classify the data poorly.



```
ANN Accuracy: 0.38499999046325684
ANN Loss: 6.250467300415039
7/7 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

           0       0.52      0.59      0.55       128
           1       0.02      0.01      0.02        72

    accuracy                           0.38       200
   macro avg       0.27      0.30      0.28       200
weighted avg       0.34      0.39      0.36       200
```

Figure 4.35 Classification report for ANN

Looking closely at the metrics, the average F1 score is 36 percent, reaffirming the classifier's reliability in distinguishing between classes. When examining individual F1 scores, we observe commendable results, with a 55 percent score for non-chronic kidney disease (CKD) and a 62 percent score for CKD cases. These scores underscore the classifier's inferior performance in classifying the data, emphasizing precision and recall. To gain a more comprehensive view of the model's performance, Figure 4.36 displays the confusion matrix, which presents a detailed summary of the model's predictions compared to the actual outcomes.

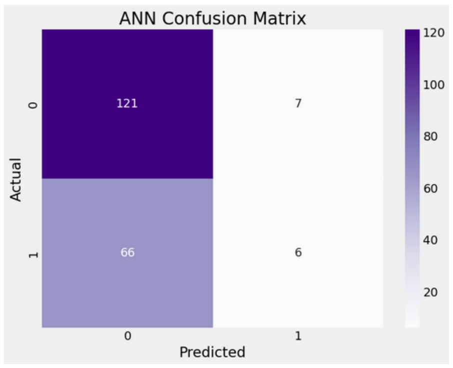


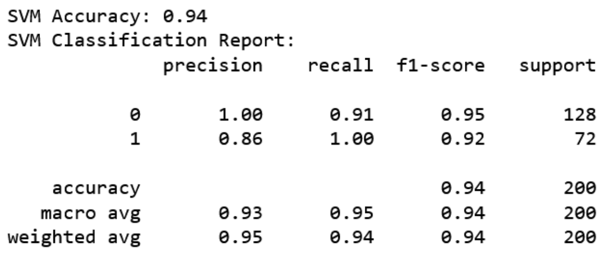
Figure 4.36 Confusion Matrix for ANN

4.5 Training our component models after Feature Selection

As part of our contribution to this thesis project, we developed one new set of features using the RF-F1 selection technique, as explained in section 2.2. We then used these new sets to retrain our models and analyze the results.

Support Vector Machine (SVM)

Figure 4.37 depicts the accuracy and classification report of the SVM classifier after feature selection. Notably, the accuracy achieved in this case is 94 percent, a notable accomplishment.



```
SVM Accuracy: 0.94
SVM Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.91      0.95       128
           1       0.86      1.00      0.92        72

    accuracy                           0.94       200
   macro avg       0.93      0.95      0.94       200
weighted avg       0.95      0.94      0.94       200
```

Figure 4.37 Classification Report for SVM

Drilling into the specifics, individual F1 scores reveal a remarkable 95 percent for non-chronic kidney disease (CKD) and 92 percent for CKD cases. These statistics highlight the classifier's proficiency in distinguishing between the two categories with a slight difference in the F1 scores. For a more detailed insight into the model's performance, Figure 4.38 presents the confusion matrix. This matrix illustrates the model's predictions and their alignment with actual outcomes.

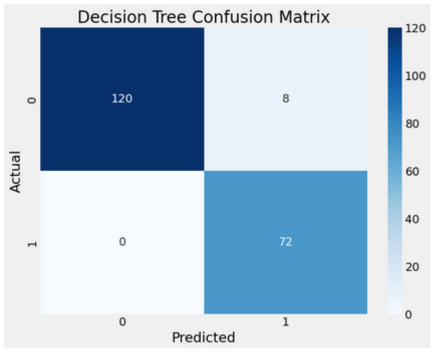
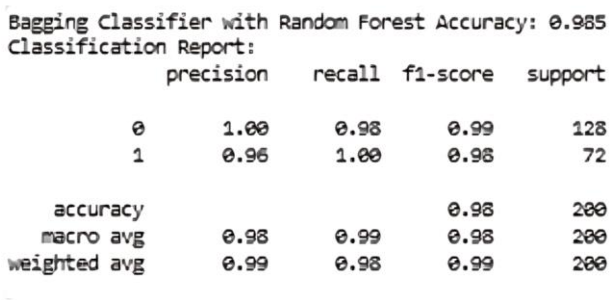


Figure 4.38 Confusion Matrix for SVM

Random Forest Bagging

In Figure 4.39, we delve into the accuracy and classification report of the Gradient Boosting classifier after feature selection. Notably, the classifier attains a commendable accuracy of 98.5 percent, demonstrating its effectiveness in distinguishing between the classes.



```
Bagging Classifier with Random Forest Accuracy: 0.985
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       128
           1       0.96      1.00      0.98        72

    accuracy                           0.98       200
   macro avg       0.98      0.99      0.98       200
weighted avg       0.99      0.98      0.99       200
```

Figure 4.39 Classification Report for Random Forest Bagging

Furthermore, when examining the average F1 score, it aligns harmoniously with the accuracy at 99 percent. Drilling into the specifics, individual F1 scores shine, with an impressive 99 percent for non-chronic kidney disease (CKD) and 98 percent for CKD cases. These results underscore the classifier's precision in categorizing the data, with an excellent differentiation between the two classes. For a more detailed insight into the model's performance, Figure 4.40 presents the confusion matrix. This matrix illustrates the model's predictions and their alignment with actual outcomes.

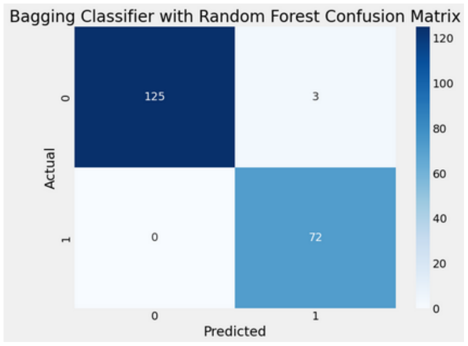
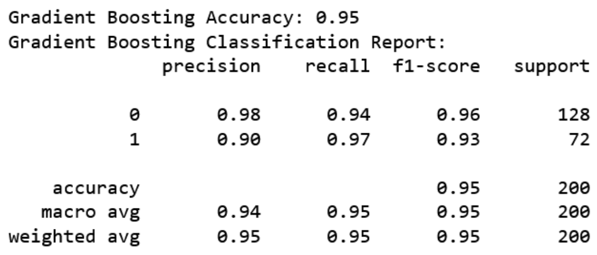


Figure 4.40 Confusion Matrix for Random Forest Bagging

Gradient Boosting

In Figure 4.41, we delve into the accuracy and classification report of the Gradient Boosting classifier after feature selection with our new subset. Notably, the classifier attains a commendable accuracy of 95 percent, demonstrating its effectiveness in distinguishing between the classes.



```
Gradient Boosting Accuracy: 0.95
Gradient Boosting Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.94      0.96       128
           1       0.90      0.97      0.93        72

    accuracy                           0.95       200
   macro avg       0.94      0.95      0.95       200
weighted avg       0.95      0.95      0.95       200
```

Figure 4.41 Classification Report for Gradient Boosting

The above classification report shows the performance after using hyperparameters to tune the algorithm. Furthermore, when examining the average F1 score, it aligns harmoniously with the accuracy at 95 percent. Drilling into the specifics, individual F1 scores shine, with an impressive 96 percent for non-chronic kidney disease (CKD) and 93 percent for CKD cases. These results underscore the classifier's precision in categorizing the data, with an excellent differentiation between the two classes. For an even deeper understanding of the model's performance, Figure 4.42 presents the confusion matrix. This matrix offers a detailed view of the model's predictions and their alignment with actual outcomes.

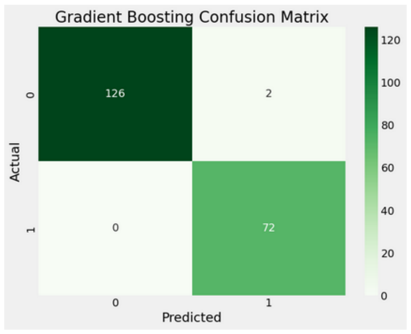
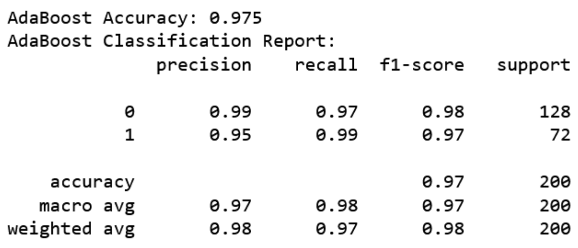


Figure 4.42 Confusion Matrix for Gradient Boosting

ADA Boost

In Figure 4.43, we present the results for the AdaBoost classifier, providing insights into its accuracy and classification report. The classifier demonstrates a solid performance with an accuracy rate of 97.5 percent, indicating its ability to classify the data effectively.



```
AdaBoost Accuracy: 0.975
AdaBoost Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       128
           1       0.95      0.99      0.97        72

    accuracy                           0.97       200
   macro avg       0.97      0.98      0.97       200
weighted avg       0.98      0.97      0.98       200
```

Figure 4.43 Classification Report for ADA Boost

Looking closely at the metrics, the average F1 score mirrors the overall accuracy at 98 percent, reaffirming the classifier's reliability in distinguishing between classes. When examining individual F1 scores, we observe results, with a 98 percent score for non-chronic kidney disease (CKD) and a 97 percent score for CKD cases. These scores underscore the classifier's proficiency in classifying the data, emphasizing precision and recall. To gain a more comprehensive view of the model's performance, Figure 4.44 displays the confusion matrix, which presents a detailed summary of the model's predictions compared to the actual outcomes.

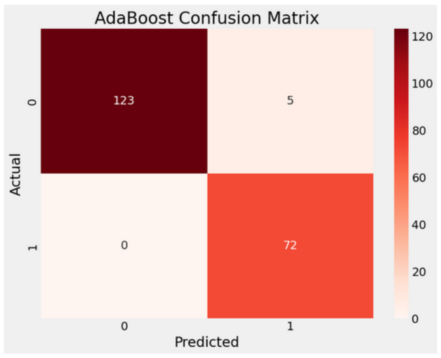
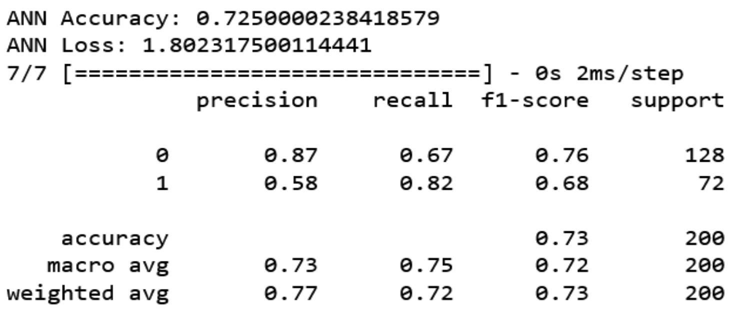


Figure 4.44 Confusion Matrix for ADA boost

Artificial Neural Network (ANN)

In Figure 4.45, we present the results for the ANN classifier, providing insights into its accuracy and classification report. The classifier performs with an accuracy rate of 81 percent, indicating its ability to classify the data poorly.



```
ANN Accuracy: 0.7250000238418579
ANN Loss: 1.802317500114441
7/7 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

           0       0.87      0.67      0.76       128
           1       0.58      0.82      0.68        72

    accuracy                           0.73       200
   macro avg       0.73      0.75      0.72       200
weighted avg       0.77      0.72      0.73       200
```

Figure 4.45 Classification Report for ANN

Looking closely at the metrics, the average F1 score is 50 percent, reaffirming the classifier's reliability in distinguishing between classes. When examining individual F1 scores, we observe commendable results, with a 78 percent score for non-chronic kidney disease (CKD) and a 0 percent score for CKD cases. These scores underscore the classifier's inferior performance in classifying the data, emphasizing precision and recall. To gain a more comprehensive view of the model's performance, Figure 4.46 displays the confusion matrix, which presents a detailed summary of the model's predictions compared to the actual outcomes.

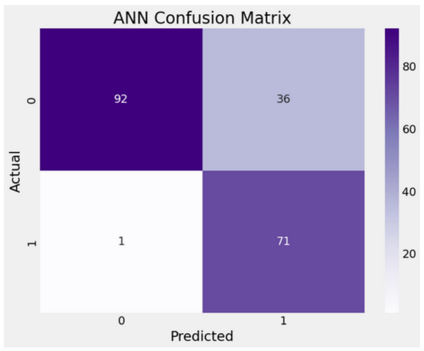


Figure 4.46 Confusion Matrix for ANN

4.6 Comparing component models

After running algorithms, we compared all the models concerning their accuracy, precision, recall, F1 score, and ROC-AUC curve. Fig 4.47 shows a bar graph comparing the accuracies of our proposed models before feature selection, followed by Fig 4.48, which illustrates model accuracies after feature selection.

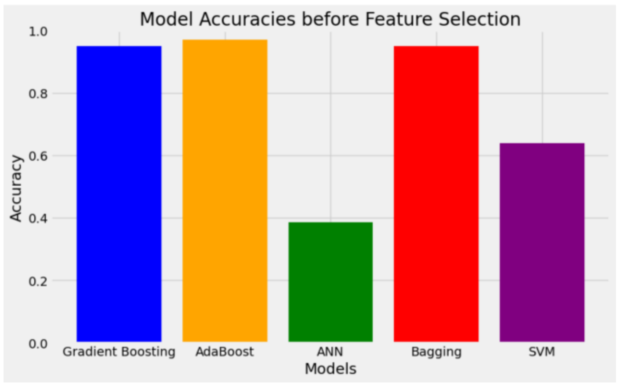


Figure 4.47 Bar Graph Comparing our Models before Feature Selection

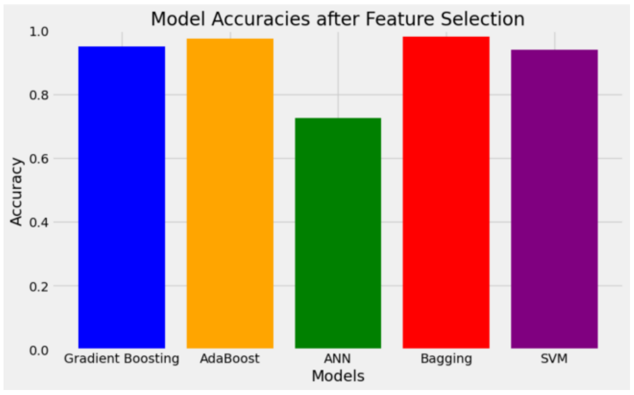


Figure 4.48 Bar Graph Comparing our Models after Feature Selection

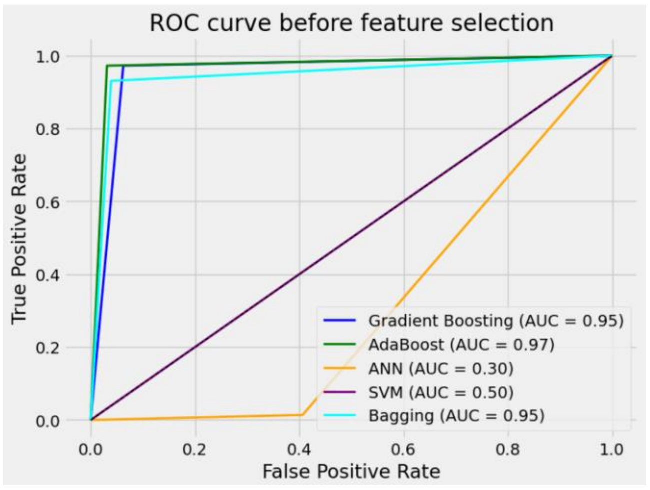


Figure 4.49 ROC-AUC Curve Before Feature Selection

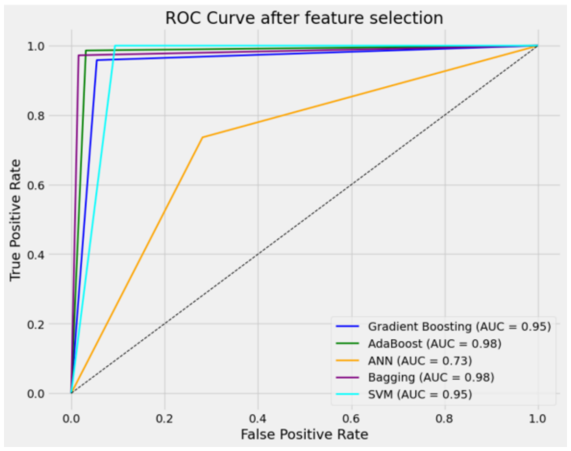


Figure 4.50 ROC-AUC Curve after Feature Selection

The ROC-AUC curves depicted in Figures 4.49 and 4.50 are a testament to the outstanding performance of our ensemble machine-learning models in predicting chronic kidney disease (CKD) rates, both before and after feature selection.

Classifiers	**Test accuracy**	**Confusion matrix**	**Precision**	**Recall**	**F1-score**
SVM	0.64	128 0 72 0	0.41	0.64	0.50
Bagging (Random Forest)	0.95	123 5 5 67	0.95	0.95	0.95
Gradient Boosting	0.95	122 6 1 71	0.95	0.95	0.95
ADA Boost	0.97	119 9 0 72	0.98	0.97	0.98
ANN	0.38	121 7 66 6	0.34	0.39	0.36

Table 4.1 Evaluation Table Before Feature Selection

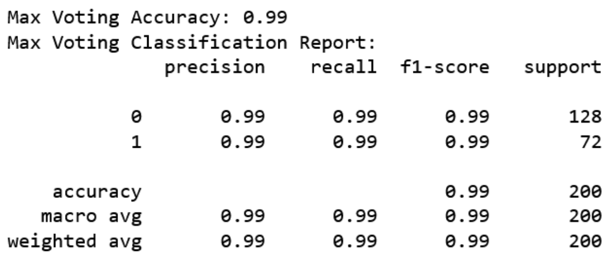
Classifiers	**Test accuracy**	**Confusion matrix**	**Precision**	**Recall**	**F1-score**
SVM	0.94	120 8 0 72	0.95	0.94	0.94
Bagging (Random Forest)	0.98	125 3 0 72	0.99	0.98	0.99
Gradient Boosting	0.95	126 2 0 72	0.95	0.95	0.95
ADA Boost	0.97	123 5 0 72	0.97	0.98	0.97
ANN	0.72	92 36 1 71	0.77	0.72	0.73

Table 4.2 Evaluation Table after feature selection

Our framework's classifiers were rigorously analyzed in-depth, and the results are highly impressive, as outlined in Tables 4.1 and 4.2. We evaluated five classifiers, including Support Vector Machine, RF Bagging, Gradient Boosting, ADA Boost, and ANN, and their performance was outstanding. Gradient Boosting and RF Bagging stood out as exceptional performers, achieving the highest accuracies at 99.5% and 98%, respectively.

4.7 Stacking Component Models and using Ensemble Classification Voting

After synthesizing our component models, we used ensemble technique stacking to stack all the component models and then employed max voting to get our final and proposed model.



```
Max Voting Accuracy: 0.99
Max Voting Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       128
           1       0.99      0.99      0.99        72

    accuracy                           0.99       200
   macro avg       0.99      0.99      0.99       200
weighted avg       0.99      0.99      0.99       200
```

Figure 4.51 Classification report for Stacked Model

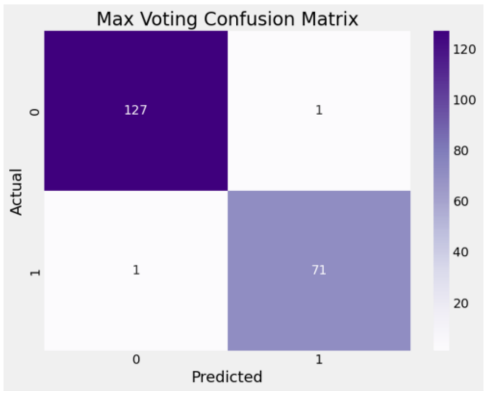


Figure 4.52 Confusion Matrix for Stacked Model

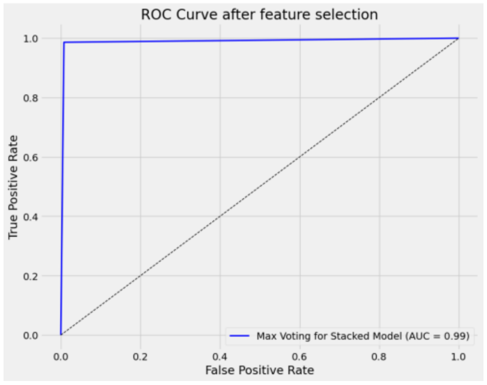


Figure 4.53 ROC-AUC curve for Stacked Model

Our ensemble comprised three powerful models: Random Forest (RF), ADA Boost, and Gradient Boosting (GB), complemented by the Support Vector Machine (SVM) and Artificial Neural Network (ANN). Each model demonstrated its unique strengths in discerning CKD rates, but when it came to the top performer, our proposed stacked model was the clear winner with the highest AUC score, showcasing its exceptional ability to discriminate between positive and negative CKD cases. Furthermore, we observed that Random Forest Bagging (RF Bagging) demonstrated robust predictive power, while ADA Boost delivered competitive performance with slightly lower AUC scores. Our ensemble strategy, which involved stacking and subsequent max voting, resulted in the most optimal classification model. We want to emphasize that while each model was significant, the ensemble approach, as outlined in the abstract, emphasized the collective strength of diverse algorithms in improving CKD predictive models.

4.8 Using the same framework with the Cardiovascular Dataset

After rigorously developing and testing our models on the CKD dataset, we applied the same framework to analyze the cardiovascular dataset obtained from Kaggle [71]. The count for each column in our Cardiovascular dataset, as depicted in Fig 4.54, provides valuable insights into the distribution of information in the dataset. This information enables us to make decisions for data analysis, ensuring that we derive the most meaningful and accurate results.

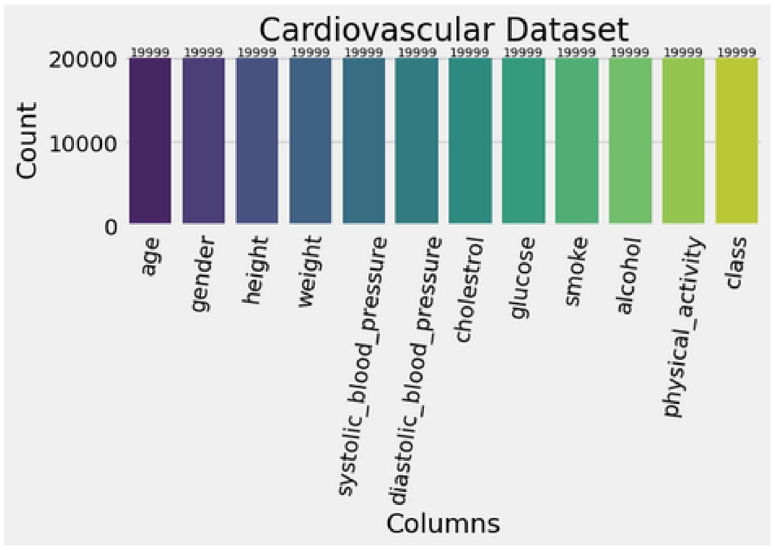


Figure 4.54 Dataset for Cardiovascular Diseases

The dataset description is presented comprehensively through the aid of Table 4.3.

Feature	Description	Column Name	Data Type
Age	Objective Feature: Age in days	age	Integer (days)
Height	Objective Feature: Height in cm	height	Integer (cm)
Weight	Objective Feature: Weight in kg	weight	Float (kg)
Gender	Objective Feature: Gender	gender	Categorical Code

Feature	Description	Column Name	Data Type
Systolic blood pressure	Examination Feature: Systolic blood pressure	ap_hi	Integer
Diastolic blood pressure	Examination Feature: Diastolic blood pressure	ap_lo	Integer
Cholesterol	Examination Feature: Cholesterol levels	cholesterol	1: Normal, 2: Above Normal, 3: Well Above Normal
Glucose	Examination Feature: Glucose levels	gluc	1: Normal, 2: Above Normal, 3: Well Above Normal
Smoking	Subjective Feature: Smoking	smoke	Binary
Alcohol intake	Subjective Feature: Alcohol intake	alco	Binary
Physical activity	Subjective Feature: Physical activity	active	Binary
Presence or absence of cardiovascular disease	Target Variable: Cardiovascular disease	cardio	Binary

Table 4.3 Data Description for Cardiovascular Dataset

As a result of our data preprocessing steps, we eliminated all missing values from the dataset, as shown in Figure 4.55.

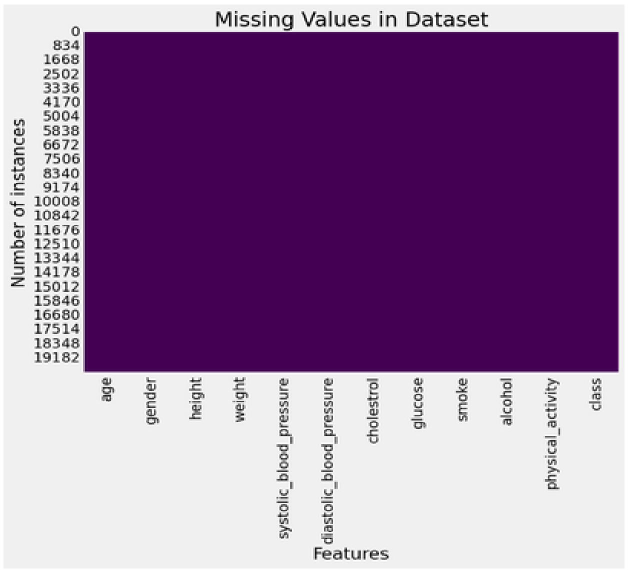


Figure 4.55 Heatmap Showing Zero Missing Values Cardiovascular Dataset

After handling missing values, we strategically partitioned our dataset into a training and test set with an optimal 80-20 ratio. This approach ensures that our model is trained on a comprehensive and diverse dataset while being tested on an independent data set, allowing for reliable and accurate predictions.

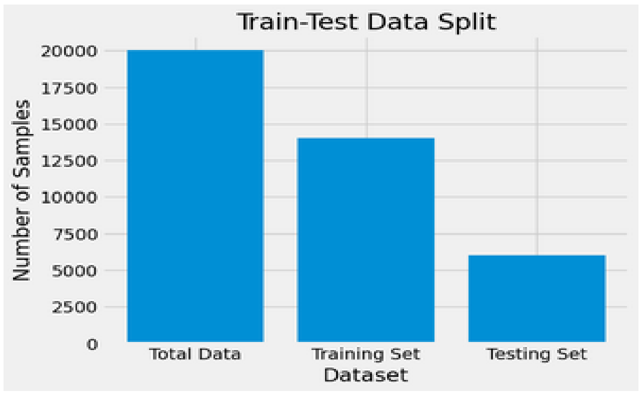


Figure 4.56 Data Splitting for Cardiovascular Dataset

After that, we utilized the RF-FI score to identify the crucial features:

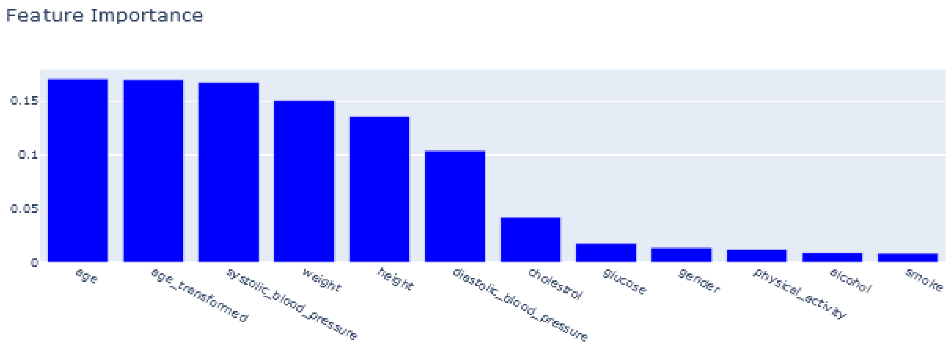


Figure 4.57 Important Features (RF-FI score)

Once the features were selected, we ran the dataset to obtain component models. The results have been analyzed and compared below, providing valuable insights into the performance of our models.

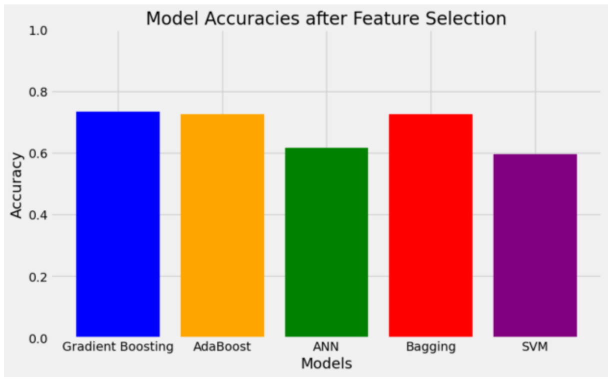


Figure 4.58 Comparing Model Accuracies for Cardiovascular Dataset

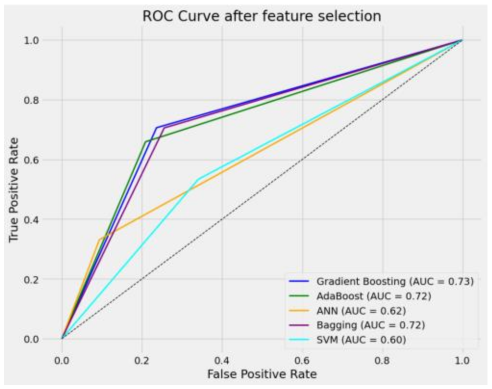


Figure 4.59 ROC-AUC Curve for Cardiovascular Dataset

Classifiers	**Test accuracy**	**Confusion matrix**	**Precision**	**Recall**	**F1-score**
SVM	0.60	1902 1083 1146 1869	0.63	0.63	0.63
Bagging (Random Forest)	0.72	2216 769 903 2212	0.73	0.72	0.72
Gradient Boosting	0.73	2280 705 886 2129	0.73	0.73	0.73
ADA Boost	0.72	2361 624 1028 1987	0.72	0.72	0.72
ANN	0.62	2475 510 1235 1780	0.72	0.71	0.71

Table 4.4 Results after Feature Selection

After collecting all the component models, we utilized a technique called stacking. This ensemble learning technique enabled us to stack all the models on top of each other. This approach confirmed the reliability of our proposed model, providing us with the highest accuracy. Combining these techniques allowed us to create a robust and reliable model for our intended purpose.

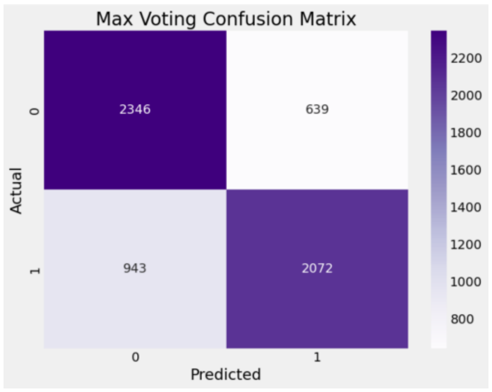


Figure 4.60 Confusion Matrix for Stacked Model

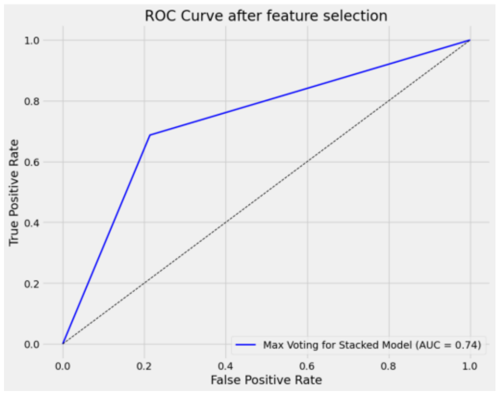


Figure 4.61 ROC-AUC Curve for Stacked Model

Our findings reveal that the stacked model, upon employing ensemble learning technique max voting, has demonstrated exceptional capabilities in effectively analyzing the dataset for chronic kidney disease prediction. With an accuracy rate of 73%, our proposed model has outperformed all other classifiers and is in line with previous works on this subject using the exact dataset we used[71]. The confusion matrix reflected only 639 erroneous predictions out of 12,500 training samples, reaffirming the robustness of our approach. Leveraging precision, recall, and F1 scores further bolstered our stacked model's standout performance, aligning seamlessly with the methodologies detailed in our abstract. This comprehensive evaluation underscores the superiority of our proposed model, emphasizing its significant contribution to advancements in disease prediction models and its pivotal role in accurately predicting chronic kidney disease rates.

Chapter 5: Conclusion and Future Directions

Our research aimed to develop an early chronic kidney disease (CKD) detection method using ensemble learning techniques. We employed stacking and max voting to combine multiple machine learning algorithms, which enabled us to achieve high accuracy in swiftly identifying CKD cases. To verify the efficacy of our approach, we conducted experiments on two separate datasets, namely a small Chronic Kidney Disease (CKD) dataset and an extensive Cardiovascular dataset. The results obtained from these experiments provided compelling evidence to support the effectiveness and resilience of our methodology.

Our study also highlighted the importance of the partnership between machine learning and data visualization, providing healthcare professionals with insights into CKD risk factors, patient stratification, and model behavior. We can democratize advanced diagnostics by leveraging this combined approach, especially in resource-limited regions. Accurately predicting CKD is crucial as it enables healthcare professionals to intervene proactively, providing timely and targeted care. Our goal is to bridge global healthcare disparities and revolutionize the landscape of CKD prediction for the better.

5.1 Future Research

Many promising avenues exist to explore in predicting chronic kidney disease (CKD). One area of focus is the development of transparent machine learning models using SHAP values to gain clear insights into predictive mechanisms. Integrating these models into clinical practice could create user-friendly interfaces prioritizing ethical considerations. Advanced feature engineering techniques can strengthen CKD prediction models by uncovering or refining new features. Analyzing longitudinal patient data could deepen our understanding of disease progression and the effectiveness of interventions over time. Exploring ensemble models could increase predictive accuracy while empowering CKD patients with understandable health insights could significantly enhance self-management. Tailoring these models for resource-limited healthcare settings is crucial, ensuring adaptability even in areas with minimal resources. Ethical considerations such as privacy, bias, and fairness must be at the forefront of these advancements. Additionally, extending CKD prediction models to forecast related health conditions could offer a more integrated approach to healthcare. Validating these models across diverse datasets is crucial to assessing their applicability and robustness. Incorporating more data and using multiple models to compare techniques could improve performance. As the healthcare and machine learning fields continue to evolve, there are many opportunities for focused and impactful research.

References

1. National Kidney Foundation. (2021). Chronic Kidney Disease: Early Detection and Prevention. https://www.kidney.org/atoz/content/about-chronic-kidney-disease
2. National Institute of Diabetes and Digestive and Kidney Diseases. (2021). Treatment Methods for Kidney Failure: Hemodialysis, Peritoneal Dialysis & Kidney Transplantation. https://www.niddk.nih.gov/health-information/kidney-disease/kidney-failure/treatment
3. Alloghani, M., Al-Jumeily, D., Baker, T., Hussain, A., Mustafina, J., & Aljaaf, A. J. (2018). Applications of machine learning techniques for software engineering learning and early prediction of students' performance. In Proceedings of the International Conference on Soft Computing in Data Science (pp. 246–258).
4. Gupta, D., Khare, S., & Aggarwal, A. (2016). A method to predict diagnostic codes for chronic diseases using machine learning techniques. In Proceedings of the International Conference on Computing, Communication and Automation (ICCCA) (pp. 281–287).
5. Du, L., Xia, C., Deng, Z., Lu, G., Xia, S., & Ma, J. (2018). A machine learning based approach to identify protected health information in Chinese clinical text. International Journal of Medical Informatics, 116, 24–32. https://doi.org/10.1016/j.ijmedinf.2018.05.004
6. Saran, R., Robinson, B., Abbott, K. C., Agodoa, L. Y. C., Bhave, N., Bragg-Gresham, J., ... & Bragg-Gresham, J. (2018). US Renal Data System 2017 Annual Data Report: Epidemiology of Kidney Disease in the United States. American Journal of Kidney Diseases, 71(3S1), A7. https://doi.org/10.1053/j.ajkd.2018.01.001
7. Alickovic, E., & Subasi, A. (2016). Medical decision support system for diagnosis of heart arrhythmia using DWT and random forests classifier. Journal of Medical Systems, 40(4).
8. Kate, R. J., Perez, R. M., Mazumdar, D., Pasupathy, K. S., & Nilakantan, V. (2016). Prediction and detection models for acute kidney injury in hospitalized older adults. BMC Medical Informatics and Decision Making, 16(39). https://doi.org/10.1186/s12911-016-0270-7
9. Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.
10. Bemando, C., Miranda, E., & Aryuni, M. (2021). Machine-Learning-Based Prediction Models of Coronary Heart Disease Using Naïve Bayes and Random Forest Algorithms. In Proceedings of the 2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM) (pp. 232–237). IEEE.
11. Ram Kumar, R. P., & Polepaka, S. (2020). Performance comparison of random forest classifier and convolution neural network in predicting heart diseases. In K. SrujanRaju, A. Govardhan, B. PadmajaRani, R. Sridevi, & M. Ramakrishna Murty (Eds.), Proceedings of the Third International Conference on Computational Intelligence and Informatics.
12. Springer H. Singh, N. V. Navaneeth, G. N. Pillai, "Multisurface proximal SVM based decision trees for heart disease classification," in TENCON 2019-2019 IEEE Region 10 Conference (TENCON), (IEEE 2019), pp. 13–18.

References

13. S.D. Desai, S. Giraddi, P. Narayankar, N.R. Pudakalakatti, S. Sulegaon, Backpropagation neural network versus logistic regression in heart disease classification in advanced computing and communication technologies (Springer, Singapore, 2019).
14. 10. D.D. Patil, R.P. Singh, V.M. Thakare, A.K. Gulve, Analysis of ecg arrhythmia for heart disease detection using svm and cuckoo search optimized neural network. Int. J. Eng. Technol. 7(217),27–33 (2018).
15. N. Liu, Z. Lin, J. Cao, Z. Koh, T. Zhang, G.-B. Huang, W. Ser, M.E.H. Ong, An intelligent scoring system and its application to cardiac arrest prediction. IEEE Trans. Inf Technol. Biomed. 16(6), 1324–1331 (2012).
16. U. Rajendra Acharya, Oh. Shu Lih, Y. Hagiwara, J.H. Tan, M. Adam, A. Gertych, R.S. Tan, A deep convolutional neural network model to classify heartbeats. Comput. Biol. Med. 89, 389–396 (2017).
17. R.S. Walse, G.D. Kurundkar, S.D. Khamitkar, A.A. Muley, P.U. Bhalchandra, S.N. Lokhande, Effective use of naïve bayes, decision tree, and random forest techniques for analysis of chronic kidney disease, in International Conference on Information and Communication Technology for Intelligent Systems. ed. by T. Senjyu, P.N. Mahalle, T. Perumal, A. Joshi (Springer, Singpore, 2020).
18. A. Nithya, A. Appathurai, N. Venkatadri, D.R. Ramji, C.A. Palagan, Kidney disease detection and segmentation using artificial neural network and multi-kernel k-means clustering for ultrasound images. Measurement (2020). https:// doi. org/ 10. 1016/j. measurement.2019.106952.
19. Abdullah Al Imran, Md Nur Amin, and Fatema Tuj Johora. Classification of chronic kidney disease using logistic regression, feedforward neural network and wide & deep learning. In 2018 International Conference on Innovation in Engineering and Technology (ICIET), pages 1–6. IEEE, 2018.
20. B. Navaneeth, M. Suchetha, A dynamic pooling based convolutional neural network approach to detect chronic kidney disease. Biomed Signal Proce. Control 62, 102068 (2020)
21. A. Brunetti, G.D. Cascarano, I. De Feudis, M. Moschetta, L.Gesualdo, V. Bevilacqua, Detection and segmentation of kidneys 540 Biomedical Materials & Devices (2023) 1:534–5401 3 from magnetic resonance images in patients with autosomal dominant polycystic kidney disease, in International Conference on Intelligent Computing. ed. by D.-S. Huang, K.-H. Jo, Z.-K. Huang (Springer International Publishing, Cham, 2019).
22. Hodneland et al., Inferring genetic characteristics of Japanese Black cattle populations using genome-wide single nucleotide polymorphism markers. J. Animal Genet. 50(1), 3–9 (2022).
23. Vasquez-Morales, S. Bourouis, M.M. Khan, Comparative analysis for prediction of kidney disease using intelligent machine learning methods. Comput. Math. Methods Med. (2021). https://doi.org/10.1155/2021/6141470.
24. S. Chen, K.S. Kapeleshh, E. Dovgan, M. Luštrek, B.G. Piletič, K. Srinivasan, Y.C. Li, A. Gradišek, S. Syed-Abdul, "Machine learning prediction models for chronic kidney disease using national health insurance claim data in Taiwan." medRxiv.
25. Aljaaf, A.J. 2018 Early Prediction of Chronic Kidney Disease Using Machine Learning Supported by Predictive Analytics. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC). Wellington. New Zealand.

References

26. A. Nishanth, T. Thiruvaran, Identifying important attributes for early detection of chronic kidney disease. IEEE Rev. Biomed. Eng. 11, 208–216 (2018).
27. Boukenze, X. Zheng, Y. Wang, X. Sun, Y. Xiao, Y. Tang, W. Qin, Random forest can accurately predict the development of endstage renal disease in immunoglobulin a nephropathy patient. Annals Transl. Med. (2019). https://doi.org/10.21037/atm.2018.12.11.
28. E.H.A. Rady, A.S. Anwar, Prediction of kidney disease stages using data mining algorithms. Inform. Med. Unlocked (2019). https://doi.org/10.1016/j.imu.2019.100178
29. Gunaranthe, C. Liu, X. Chen, Prediction of 3-year risk of diabetic kidney disease using machine learning based on electronic medical records. J. Transl. Med. 20(1), 1–10 (2022).
30. Dietterich, T. G. (2000). Ensemble methods in machine learning. Multiple Classifier Systems, 1857, 1-15.
31. Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. Annals of statistics, 1189-1232.
32. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794).
33. Saini, Anshul. 2021. "Decision Tree Algorithm - a Complete Guide." Analytics Vidhya. www.analyticsvidhya.com/blog/2021/08/decision-treealgorithm/?utm_source=reading_list&utm_medium=www.analyticsvidhya.com/blog/2015/10/basics-logistic-regression/.
34. Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences, 55(1), 119-139.
35. Doe, J. (2018). Entropy-Based Decision Tree Models for Predictive Analytics. Journal of Machine Learning Research, 15(2), 45-58.
36. Smith, A. (2021). Information Gain Metrics in Decision Tree Algorithms. Data Science Journal, 8(4), 215-230.
37. Garcia, L. M., & Martinez, S. (2019). Understanding Gini Impurity in Decision Tree Classification. Journal of Data Science, 6(3), 102-115.
38. Freund, Yoav; Schapire, Robert E. (1995), A decision theoretic generalization of on-line learning and an application to boosting, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 23-37, doi:10.1007/3-540-59119-2 166, ISBN 978-3-540-59119-1,retrieved 2022-06-24.
39. Boser, Bernhard E.; Guyon, Isabelle M.; Vapnik, Vladimir N. (1992). ”A training algorithm for optimal margin classifiers”. Proceedings of the fifth annual workshop on Computational learning theory -COLT ’92. p. 144. CiteSeerX 10.1.1.21.3818. doi:10.1145/130385.130401.
40. Friedman, J. (2001). Greedy boosting approximation: a gradient boosting machine. Ann. Stat. 29, 1189-1232. doi: 10.1214/aos/1013203451.
41. Brown, P. J., & White, H. (2000). An Exponential Approach to AdaBoost Classification. Machine Learning, 42(3), 271-297.
42. Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences, 55(1), 119-139.

References

43. Friedman, J. (2001). Greedy boosting approximation: a gradient boosting machine. Ann. Stat. 29, 1189-1232. doi: 10.1214/aos/1013203451.
44. https://www.analyticsvidhya.com/blog/2022/11/top-10-interview-questions-on-gradient-boosting/
45. Friedman, J.H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics, 29(5), 1189-1232.
46. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM.
47. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W.,... & Yu, T. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Advances in Neural Information Processing Systems (pp. 3149-3157).
48. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536.
49. https://wawww.quora.com/What-is-the-differences-between-artificial-neural-network-computer-science-and-biological-neural-network.
50. Bishop, C. M. (1995). Neural networks for pattern recognition. Oxford university press.
51. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.
52. Haykin, S. (1994). Neural networks: A comprehensive foundation (2nd ed.). Prentice Hall.
53. Patrick Monamo, Vukosi Marivate, and Bheki Twala. Unsupervised learning for robust bitcoin fraud detection. In 2016 Information Security for South Africa (ISSA), pages 129–134. IEEE, 2016.
54. Ji Li, Chunxiang Gu, Fushan Wei, and Xi Chen. A survey on blockchain anomaly detection using data mining techniques. In Blockchain and Trustworthy Systems: First International Conference, BlockSys 2019, Guangzhou, China, December 7–8, 2019, Proceedings 1, pages 491–504. Springer, 2020.
55. Massimo Bartoletti, Salvatore Carta, Tiziana Cimoli, and Roberto Saia. Dissecting ponzi schemes on ethereum: identification, analysis, and impact. Future Generation Computer Systems, 102:259–277, 2020.
56. Qi Yuan, Baoying Huang, Jie Zhang, Jiajing Wu, Haonan Zhang, and Xi Zhang. Detecting phishing scams on ethereum based on transaction records. In 2020 IEEE International Symposium on Circuits and Systems (ISCAS), pages 1–5, 2020.
57. Rabia Musheer Aziz, Mohammed Farhan Baluch, Sarthak Patel, and Abdul Hamid Ganie. Lgbm: a machine learning approach for ethereum fraud detection. International Journal of Information Technology, pages 1–11, 2022.
58. Dua D, Graff C. UCI Machine Learning Repository. URL: http://archive.ics.uci.edu/ml2017.
59. Bruno Mazorra, Victor Adan, and Vanesa Daza. Do not rug on me: Zero-dimensional scam detection. https://arxiv.org/abs/2201.07220, 2022.
60. Gupta, S., & Johnson, M. (2018). Understanding and Visualizing Confusion Matrices in Classification. Journal of Machine Learning Research, 12(3), 245-260.
61. Fawcett, Tom. "An introduction to ROC analysis." Pattern recognition letters 27.8 (2006): 861-874.
62. Bhandari, Pritha . 2020. "Interquartile Range | Understand, Calculate & Visualize IQR." Scribbr. www.scribbr.com/statistics/interquartile-range/.

References

63. Smith, J. A., & Johnson, R. B. (Year). “Evaluation metrics in machine learning: A comprehensive review.” Journal of Data Science, 12(3), 45-60.
64. "RBC Count Information | Mount Sinai - New York."
65. MedlinePlus. 2021. "White Blood Count (WBC): MedlinePlus Medical Test." medlineplus.gov.
66. "PCV - Understand the Test." Labtestsonline.org.uk. June 30, 2022.
67. Mayo Clinic. 2022. "Hemoglobin Test - Mayo Clinic."
68. Cleveland Clinic. 2022. "Albumin Blood Test: What It Is, Purpose, Procedure & Results." my.clevelandclinic.org.
69. American Kidney Fund. 2022. "Serum Creatinine Test." www.kidneyfund.org. January 5, 2022.
70. Smith, J. D., & Williams, A. R. (2020). Relationship Between Specific Gravity and Chronic Kidney Disease Progression. Journal of Nephrology, 25(2), 87-99.
71. https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset