**Enhancing Chronic Kidney Disease Detection Through Machine Learning Predictive Modeling**

**Contents**

[**1. Introduction 2**](#_Toc144499911)

[**2. Literature Review 3**](#_Toc144499912)

[**3. Methodology 9**](#_Toc144499913)

[**3.1 Data Collection 9**](#_Toc144499914)

[**3.2 Data Preprocessing 11**](#_Toc144499915)

[**3.3 Exploratory Data Analysis 11**](#_Toc144499916)

[**3.4 Feature Encoding 12**](#_Toc144499917)

[**3.5 KNN Imputation for Missing Values 12**](#_Toc144499918)

[**3.6 Hypothesis Testing 12**](#_Toc144499919)

[**3.7 Feature Scaling 12**](#_Toc144499920)

[**3.8 Data Splitting 12**](#_Toc144499921)

[**3.9 Model Building 12**](#_Toc144499922)

[**3.10 Model Training 16**](#_Toc144499923)

[**3.11 Model Evaluation 16**](#_Toc144499924)

[**4. Results 16**](#_Toc144499925)

[**4.1 Preparing Data 16**](#_Toc144499926)

[**4.2 Exploring the Data 20**](#_Toc144499927)

[**4.3 Feature Encoding 32**](#_Toc144499928)

[**4.4 Imputing Missing Values with KNN Imputer 33**](#_Toc144499929)

[**4.5 Hypothesis Testing 33**](#_Toc144499930)

[**4.6 Scaling 42**](#_Toc144499931)

[**4.7 Splitting 42**](#_Toc144499932)

[**4.8 Model Building 42**](#_Toc144499933)

[K-Nearest Neighbor Classifier 42](#_Toc144499934)

[Decision Tree Classifier 44](#_Toc144499935)

[Random Forest Classifier 45](#_Toc144499936)

[AdaBoost/Adaptive Boost Classifier 47](#_Toc144499937)

[Gradient Boosting Classifier 48](#_Toc144499938)

[Stochastic Gradient Boosting Classifier 49](#_Toc144499939)

[XgBoost Classifier 50](#_Toc144499940)

[CatBoost Classifier 51](#_Toc144499941)

[Extra Trees Classifier 53](#_Toc144499942)

[LightGBM Classifier 54](#_Toc144499943)

[**4.9 Performance Evaluation 54**](#_Toc144499944)

[**5. Conclusion and Recommendations 55**](#_Toc144499945)

[**5.1 Conclusion 55**](#_Toc144499946)

[**5.2 Recommendations 56**](#_Toc144499947)

[**References 57**](#_Toc144499948)

[**Appendix 59**](#_Toc144499949)

# **1. Introduction**

The kidneys' capacity to work normally and filter waste from the blood is affected by the chronic kidney disease (CKD), a disorder that worsens over time. It is a problem for world health that has several risk factors, including diabetes, hypertension, obesity, and ageing. Cardiovascular issues, anaemia, bone problems, and finally end-stage of renal disease which is necessitating the dialysis or of the kidney transplantation are just a few of the serious implications of CKD (Aworinde et al., 2023). To stop the progression of CKD, avoid complications, and enhance patient outcomes, early detection and therapy are essential. A major global public health concern, chronic kidney disease (CKD) is characterized through the progressive turn down in the function of the kidney over a period of time (Virk et al., 2023). It is a silent killer that often remains undiagnosed until the disease has reached advanced stages, leading to severe complications and increased mortality rates. Swift detection and intervention are the crucial things in managing the CKD and averting its progression to the final stage of the renal disease.

The emergence of machine learning (ML) techniques has revolutionized the healthcare sector, offering new opportunities for predictive modelling and early disease detection. In this dissertation, I aim to develop a predictive model by using various the algorithms of ML to accurately identify the occurrence of CKD at early stages. By exploiting the potential of ML, this research seeks to contribute to the development of effective tools that can aid healthcare professionals in making informed decisions and improving patient outcomes.

The proposed research will employ a comprehensive range of ML algorithms, including K-nearest neighbors (KNN), Decision Tree Classifier, Random Forest Classifier, Ada Boost Classifier, Gradient Boosting Classifier, Stochastic Gradient Boosting (SGB), XGBoost, Cat Boost Classifier, Extra Trees Classifier, and LGBM Classifier. By exploring multiple algorithms, I aim to compare their performance and identify the most effective model for CKD prediction. The primary dataset for this study will consist of anonymized patient records, including demographic information, medical history, laboratory test results, and clinical findings. These all features will assist as input variables in ML algorithms, enabling the models to learn patterns and relationships between the input variables and for the occurrence of CKD. By training models on a large dataset, I aim to achieve high accuracy and robustness in the prediction of CKD.

To ensure the reliability and generalizability of the predictive model, rigorous evaluation techniques will be employed. The dataset will be classfied into the training, validation, and testing subsets, using the techniques such as k-fold cross-validation in order to test the performance of the ML algorithms. Evaluation metrics such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC) will be utilized to measure the effectiveness of the models. The anticipated outcomes of this research include the development of a robust and accurate predictive model for the swift detection of the CKD. The key results obtained from this study may assist healthcare professionals in identifying individuals at high risk of CKD, enabling timely interventions and improved patient care. Additionally, the research findings may provide insights into the most effective ML algorithms for CKD prediction, contributing to the broader field of ML applications in healthcare.

In conclusion, this dissertation aims to leverage ML techniques to develop the predictive model for swift or early detection of the CKD. By utilizing various ML algorithms and evaluating their performance, this research seeks to contribute to the advancement of healthcare practices, enabling timely interventions and improved outcomes for patients at risk of CKD.

# **2. Literature Review**

The study of machine learning focuses on creating algorithms and models that let computers learn from data and make predictions or judgements without having to be explicitly programmed. In other words, machine learning enables computers to recognise links and patterns in data and then use those patterns to draw conclusions or forecast outcomes regarding brand-new, unforeseen data.

The idea of building a model from scratch using a dataset lies at the heart of machine learning. During this training, a sizable amount of data including input features and related target outcomes are presented to the model. By locating underlying patterns and correlations, the model learns from this data. After being trained, this model can be used to make the predictions which are based on fresh data.

There are 3 main branches of the machine learning which are either supervised or unsupervised and reinforced learning.

A diagram of machine learning

Description automatically generated

As mentioned in (*Machine Learning Algorithms - A Review | Enhanced Reader*, n.d.), Arthur Samuel, who was famous for developing the program for playing checkers described the machine learning like a field study which gives an ability to computers to study without being specifically programmed. The author highlights that it is the basic nature of the human tendency to find simpler ways to do tasks. In this process of simplifying things, humans have invented machines which have helped mankind in many ways and machine learning was one such invention. In (Jiang et al., 2020), the authors highlighted Hernán and colleagues’ framework, per whom description, causal inference and prediction are the 3 major data science tasks. Depending on the specific achievable, traditional statistical methods could be used or the methods of the machine learning could also be used. In the machine learning literature, features are used to predict targets. I will now discuss the three main branches of machine learning.

**Supervised Learning:**

A diagram of a model training

Description automatically generated

In this method, the input features are linked with the appropriate output or goal values, and the model has been trained on the labelled data. The model required to learn a relationship between the inputs and the outputs to be capable enough to predict outcomes accurately for brand-new datasets. This is again divided into 3 types, namely probabilistic classifiers, linear classifiers, and others (Saravanan & Sujatha, 2019). The probabilistic classifier employs mixture models to categorize data. Every class is taken into account as a component of the combination. Every component of a mixture is the generic model that offers chance to sample the particular term of component. Whereas the purpose of a linear classifier is to arrange objects into groups based on similar feature values. The above image illustrates the flow of supervised learning.

**Unsupervised learning:**

A diagram of gears and machine

Description automatically generated

The process of unsupervised learning is building a model from unlabelled data. Without explicit target labels, the model must find patterns or structures in the data. Common unsupervised learning tasks include dimensionality reduction and clustering. This is also called clustering analysis. There are two basic clustering strategies hierarchical clustering and partitioning (Gentleman & Carey, 2008).

**Reinforcement Learning:**

A diagram of a person's head

Description automatically generated

An agent who gains the skills of decision-making through the interactions with its environment through the reinforcement learning. Then the agent learns to take the activities that maximise cumulative rewards over the time by receiving feedback by the means of rewards or penalties based on its actions. Technology for reinforcement learning has a very long history and is derived from fields like psychology, control theory, statistics, and more. The theory of Animal learning is usually the foundation for the reinforcement learning. Through the knowledge it learns by trial and error and by the continuous interaction with the dynamic environment, RL can independently determine optional policy without the need for prior knowledge (Wang & Zhan, 2011).

**In Medical Field:**

A number of research have been done to incorporate machine learning into medicine, and this is proving to be important, especially in light of the numerous medical problems humanity has been dealing with recently.

In one such attempt, machine learning was used to create a tool ‘C-Path’ which improved breast cancer identification. What’s interesting is that numerous competitions are being conducted to better identify cancer risks (Deo, 2015)

Another research highlights many applications like use of SVMs in identifying Type-1 diabetes patients, identifying patients with Crohn’s disease and other scenarios where early identification could prove useful (Maceachern & Forkert, 2021)

**In Our Scope:**

Machine learning (ML) algorithms can usually be applied for a variety of fields, and for predicting chronic kidney disease (CKD) they have shown promise for early detection and preventive therapy. Worldwide, CKD is a major cause for concern regarding health, and early detection can greatly enhance patient outcomes. In this review of the literature, I seek to examine the body of knowledge on prediction of the CKD using machine learning (ML) algorithms, such as K-nearest neighbours, Decision Tree Classifier, Random Forest Classifier, Ada Boost Classifier, Gradient Boosting Classifier, Stochastic Gradient Boosting (SGB), XgBoost, Cat Boost Classifier, Extra Trees Classifier, and LGBM Classifier. I aim to identify the research's strengths, weaknesses, and gaps by a critical review of earlier studies. In doing so, I hope to advance the creation of efficient methods for CKD early detection. This review, which will serve as the basis for my research project, will give a thorough grasp of the use of ML approaches in CKD prediction.

In (Debal & Sitote, 2022), three prediction models are employed: Decision Tree (DT), Support Vector Machine (SVM), and Random Forest (RF). For feature selection, variance and recursive feature analysis removal with the cross-validation is used. The models were assessed using tenfold cross-validation. The experiment findings showed that the RF based on the recursive feature reduction with the cross validation outperforms SVM and DT in terms of performance. The random forest with RFECV feature selection achieved an accuracy of 99.8% with binary classification. However, after hyperparameter tuning, the performance was improved to 99.83% for SVM model. Multiclass classification was also performed, RF with RFECV achieved 79% accuracy while SVM after hyperparameter tuning achieved 78.78% accuracy without feature selection.

Can principal component analysis affect the machine learning prediction models in any way? The authors of (Islam et al., 2023) have developed various ML models using algorithms like Ada boost, Decision tree, Xgboost, CatBoost, KNN, Random Forest, Naive Bayes, Gradient boosting and stochastic gradient boosting etc., after performing PCA and compared the accuracies of the models with accuracies obtained in previous studies performed on the dataset without PCA. They found out that the accuracies of all models except KNN, Naive baives and hybrid have been improved. The accuracies of all models except KNN and ANN were above 90% and the proposed XgBoost model had highest accuracy, 99.2%.

In (Bai et al., 2022), machine learning was used to predict the risk of end-stage kidney disease (ESKD) in patients with chronic kidney disease (CKD). The study found that three of the ML models, including logistic regression, naïve Bayes and random forest, showed an equivalent predictability and of greater sensitivity compared to the Kidney Failure Risk Equation (KFRE). The KFRE had the highest precision, specificity and accuracy. The study employed 5 ML algorithms which include naïve Bayes, logistic regression, K-nearest neighbors, random forest, and decision tree. The performance of models was assessed using accuracy, F1 score, area under curve (AUC), recall, precision and specificity. The Kidney Failure Risk Equation (KFRE), that calculates the five-year risk of ESKD based on patient's gender, age and eGFR, was further compared to all classifiers created in this study.

The authors of (Allen et al., 2022) developed two machine learning models (MLAs) to predict diabetic kidney disease (DKD) in patients with the type 2 diabetes mellitus (T2DM). The MLAs were trained on a dataset of patients with T2DM and were assessed on hold-out test set as well as an external validation set. The MLAs excelled the CDC CKD scoring system in terms of AUROC, sensitivity, and also specificity for all three DKD endpoints (stages 3–5, any stage, and stages 4–5). The MLAs were developed using Python libraries Scikit-learn and XGB. Features fed into the models consisted of clinical measurements, demographics, patient history, and laboratory values. The models were evaluated based on AUROC, specificity, sensitivity, positive or negative possibility ratios, and DOR. This study suggests that ML can be used to predict DKD in patients with T2DM with greater accuracy than the CDC CKD scoring system.

(Baidya et al., 2022) This research presents method that utilizes eight different algorithms of ML to promptly detect CKD based on patient health condition data. Using a dataset spanning two months, provided by a hospital, the study assesses the feasibility of predicting CKD. The ML algorithms employed include AdaBoost, EXT, XGB, Gradient Boosting, XGB, Decision Tree, Random Forest, and GNB. Following data pre-processing, the algorithms are compared based on performance measures such as F1-score, accuracy, precision, recall, and also AUC score. The analysis reveals that Extra Tree Classifier and KNN excelled other algorithms by achieving the accuracy, 99% and 98%, respectively, with K-Nearest Neighbours performing the best. This research demonstrates the potential ML algorithms that are precisely predicting CKD and highlights the significance of early detection for effective management of the disease.

The authors of (Ebiaredoh-Mienye et al., 2022) proposed a clear approach which combines information-gain-based feature selection and cost-sensitive adaptive boosting (AdaBoost) classifier. The approach reduces screening time and cost by utilizing a limited number of clinical test attributes. The proposed approach outperformed other CKD prediction methods and classifiers, achieving sensitivity, an accuracy, and specificity with 99.8%, 100%, 99.8% respectively. Furthermore, the experimental results demonstrate the positive impact of feature selection on classifier performance. The recommended approach presents an efficient model of CKD diagnosis and holds potential for application to imbalanced medical datasets, enhancing disease detection effectiveness. In this proposed cost sensitive AdaBoost algorithm, it shifts weights of training samples that were incorrectly classified upward and downward with each iteration. With this weighting strategy, examples from both classes are given the same consideration regardless of whether they were successfully or incorrectly categorised.

When many models created in the past to predict chronic kidney disease were evaluated, it was shown that the majority of them either had low accuracy or had a limited application range for the approach used to impute the missing variables. Hence, a methodology was proper with contributions such as KNN imputation, which might be used on data set alongside uncertain diagnostic categories, is used to fill in the missing values in the data set, the whole CKD data sets were utilised to create CKD diagnostic models using RF, LOG, KNN, SVM, NB, naive Bayes classifier, and FNN. For mis assessed analysis, models that performed better extracted, and after missing values were loaded in utilizing KNN imputation, the integrated model created utilising perceptron’s to merge LOG and RF, which enhanced the performance of component models for the diagnosis of CKD. Perhaps, this could be the first time where KNN imputation used to fill in the missing values (Qin et al., 2020).

The literature review provided an outline of how the random forest model beat the Support Vector Machine (SVM), and Decision Tree (DT) in binary classification by 99.8% accuracy. Principal component analysis (PCA) enhanced the accuracies of all models except KNN, Naive Bayes, and hybrid in one study. In our project, a method that uses eight different ML algorithms to detect CKD quickly based on patient health condition data would be valuable. The suggested method outperforms other CKD prediction methods and classifiers by combining the information-gain-based feature selection and a cost-sensitive adaptive boosting (AdaBoost) classifier. The method also uses KNN imputation to fill in missing information, making it a useful model for CKD diagnosis that I may use in my research.

# **3. Methodology**

It is impossible to exaggerate the significance of technique in ML initiatives. A clear methodology ensures the dependability and reproducibility of outcomes by offering an organised way to address the study objective. It includes activities including data gathering, preprocessing, model choice, training, assessing, and validating. This project follows the below methodology, and the entire project is done in Python. I am going to use Visual Studio Code as our code editor tool.

## **3.1 Data Collection**

In order to predict the CKD, I need to obtain health the data of individuals which includes various health attributes like their blood pressure, specific gravity, RBC count, puss cells, blood glucose, haemoglobin, CKD status etc., Such data was obtained from the ml repository

of UCI. The data was created by L. Jerlin Rubini under the guidance of Dr. P. Eswaran from the information obtained through various patients by Dr. P. Soundarapandian of Apollo hospital in Tamil Nadu over a period of 2 months. It consists of 25 attributes and 400 records in total and the target variable is ‘Class’.

Here's the direct URL to the dataset:

<https://archive.ics.uci.edu/dataset/336/chronic+kidney+disease>

**Metadata:**

A table with text and numbers

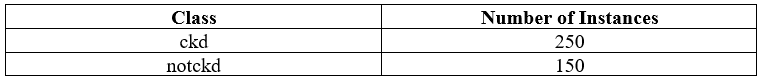
Description automatically generated

**Dependent variable or the Target variable:**

The target variable is ‘class’ which has two classifications,

1. ckd

2. notckd

The class ‘ckd’ indicates that the respective patient is in fact suffering from chronic kidney disease while ‘notckd’ indicates its absence. Below is the table which shows the dependent variable’s instances. 

## **3.2 Data Preprocessing**

Data pre-processing is a vital step that transforms the raw data to processed format for analysis and model building. It seeks to increase the data's dependability and quality, address missing values or outliers, and normalize the data distribution. One aspect of data pre-processing is handling missing values. Missing data can negatively impact model performance, and various techniques can be used to handle this issue, such as imputation or deletion of missing values based on the data and the missing percentage.

The data obtained for this project has a lot of missing values and hence omitting the null values isn’t a suitable option. I have an option to fill these null values with random sampling and mean/mode sampling methods, however, I came across KNN imputation method during literature review. As it shows promising results, I will use this to estimate and fill the missing values.

## **3.3** **Exploratory Data Analysis**

Exploratory Data Analysis (EDA) is the critical step in understanding and gaining insights from data before applying algorithms of machine learning. In the context of our project, EDA can help uncover patterns, relationships, and potential challenges in the dataset.

During the EDA process, the dataset will be analyzed and visualized to gain a comprehensive understanding of its characteristics. Here are some potential steps that I am going to undertake:

1. I will begin by examining the structure and dimensions of the dataset. Identify the number of instances (rows) and variables (columns) present. This step helps in understanding the overall size and complexity of the dataset.

2. I’ll then analyse each variable individually to understand its types, distributions, and potential issues. For example, identify numerical variables and assess their summary statistics, such as mean, median, standard deviation, and range.

3. Investigating the presence of outliers in the dataset, especially for numerical variables. Visualising the distributions using box plots or histograms to identify any extreme values which deviate considerably from majority of data. Evaluating the necessity of outlier treatment based on the characteristics of the dataset helps us to decide on specific ML algorithms being considered.

4. I’ll then explore the relationships between variables to uncover potential associations or dependencies by calculating the correlations between numerical variables and visualising them using correlation matrices. Identify any strong correlations that may require special attention during feature selection or model training to avoid multicollinearity.

5. Utilizing various visualizations, such as histograms, bar charts, scatter plots, and heatmaps, to gain insights into the data. Visualizing the distribution of variables, comparing the distributions of different classes (CKD and non-CKD), and exploring potential patterns or trends that can guide feature selection and model development.

By conducting a comprehensive exploratory data analysis, I can gain a deeper understanding of the dataset, identify potential challenges, and uncover insights that can inform subsequent steps in our project.

## **3.4 Feature Encoding**

Machine learning algorithms typically operate on numerical data, and encoding categorical or textual features is necessary to incorporate them into the model training process. By converting categorical features into numerical representations, the algorithms can capture underlying patterns as well as relationships within the data. I am going to use one hot encoding.

## **3.5 KNN Imputation for Missing Values**

The reason why I have decided to impute missing values after EDA is because I wanted to check if the imputed values will carry the same correlation as the original data indicates. I am aiming to achieve this by conducting hypothesis testing using the imputed data with the correlation information obtained through original data.

## **3.6 Hypothesis Testing**

During our exploratory data analysis, we will be exposed to a lot of information about the data, patterns will be found, and correlations would be analysed. To test their statistical significance, it is imperative for us to perform hypothesis testing using various method.

## **3.7 Feature Scaling**

Since I will be using few algorithms like KNN which is based on distance, it is necessary to scale the values to have uniformity and to avoid impact of wider spread data. I will use standard scaler function to achieve this.

## **3.8 Data Splitting**

In this step, the available dataset divided into separate subsets for training, and testing. Training set to train model, and testing set used to assess the final model's generalization ability on the unseen data. Proper data splitting ensures unbiased evaluation, prevents overfitting and helps in selecting the best-performing model.

## **3.9 Model Building**

The dominant method for classification tasks is supervised machine learning, which is able to make accurate predictions based on labeled data. Several supervised ML algorithms can be used to adapt to different classification challenges, making it an effective tool for data analysis and decision-making. Therefore, this project considers various supervised ML algorithms, such as KNN, Extra Trees Classifier, XgBoost, Stochastic Gradient Boosting (SGB), Random Forest Classifier, Decision Tree Classifier, Gradient Boosting Classifier, AdaBoost Classifier, LGBM Classifier, and Cat Boost Classifier. These algorithms are chosen based on their suitability for classification tasks and their potential to predict CKD accurately.

**K-Nearest Neighbors (KNN):**

The use of classification models is more widespread than that of regression models. Almost all analytical problems involve making decisions, which explains the bias towards classification models. There are many classification algorithms that are widely used. One of them is K-nearest neighbor (KNN). In it, similar labels or values are assumed to be associated with similar data points.

The K-nearest neighbors (KNN) method relies on the complete training dataset as a point of reference during its training phase. To make predictions, it employs a chosen distance metric, such as the Euclidean distance, to measure the separation between each of the input data point and every training example.

Subsequently, the method identifies the K nearest neighbors to the input data point, determined by their respective distances.When the task at hand is classification, the method selects the most commonly occurring class label among these K neighbors as the predicted label for the input data point. On the other hand, in regression tasks, the method forecasts the value for the input data point by calculating either the average or weighted average of the target values associated with the K neighboring data points. This approach allows the KNN method to provide predictions for a wide range of machine learning tasks effectively and intuitively.

**Decision Tree Classifier**:

Issues pertaining to regression and classification can be effectively addressed through the utilization of the decision tree methodology. In this approach, a decision tree is employed to glean uncomplicated decision rules, derived from the training data, which are subsequently utilized to construct a training model. This model is then employed to make predictions regarding the class or value of the target variable, providing a valuable tool for various machine learning tasks.

Top of Form

Bottom of Form

Within the realm of decision trees, the process commences at the tree's foundation when predicting the class label for a given record. This procedure involves a comparison between the values of the root attribute and those of the record's attribute. Subsequently, I advance to the next node by traversing the branch associated with the corresponding attribute value as determined by the comparison. This systematic approach allows for the accurate determination of class labels within the decision tree framework.

The decision tree is made up of a root node, which is typically the source data, a decision node created by splitting the root node and additional sub-nodes, a leaf node at which the splitting ends, and a branch, which is a segment of the tree.

A diagram of a algorithm

Description automatically generated

**Random Forest Classifier:**

Powerful machine learning methods like the Random Forest algorithm are employed for both classification and the regression problems. During training, it builds a variety of the decision trees. A distinct subset of data is used to train each tree, also each tree generates its own predictions. Final prediction created by combining these predictions through a voting or by an averaging procedure.

The overfitting and high variance problems that are frequently related to single decision trees are addressed by Random Forest. The approach eliminates the risk of depending too much on one tree's peculiarities by producing numerous trees with arbitrary modifications in the information and attributes utilised for splitting. Predictions are frequently more precise and stable because of this ensemble technique.

A diagram of a tree

Description automatically generated

**Ada Boost Classifier:**

AdaBoost, or adaptive boosting, is a ML ensemble technique which is mainly employed for classification tasks. It operates by aggregating the forecasts of several ineffective learners, often shallow decision trees.

Each weak learner in AdaBoost is trained using the dataset, and in subsequent iterations, the method gives misclassified data points more weights. This concentrates the weaker students' attention on the difficult-to-classify samples. Then, the learners are pooled, giving those who perform well greater weight. A weighted average of the forecasts from various poor learners makes up the final prediction.

**Gradient Boosting Classifier:**

Gradient boosting is also a ML ensemble technique which combines a number of vulnerable models, such as decision trees, to produce strong or powerful model of prediction. In order to produce accurate predictions for regression and classification tasks, it iteratively corrects mistakes caused by earlier models.

**Stochastic Gradient Boosting Classifier:**

Stochastic gradient boosting combines weak models, such as decision trees, to produce a powerful predictor. Training on random subsets of data, it introduces unpredictability while iteratively correcting faults. For regression and classification tasks, this strategy increases effectiveness and generalisation.

**XgBoost Classifier:**

XgBoost also combines weak models, and it uses a gradient-based approach for iterative learning, optimizing predictions with a mix of regularization and parallel processing. This makes XGBoost highly efficient and effective for various tasks.

**Cat Boost Classifier:**

This model succeeds by intuitively handling categorical features without any preprocessing. It effectively handles data by combining ordered boosting with gradient boosting. The characteristics of CatBoost make it precise, quick, and simple to use.

**Extra Trees Classifier:**

This algorithm uses a group of decision trees to perform classification tasks. Although it differs from Random Forest in that it adds more unpredictability to the tree-building process. This randomization increases tree diversity, resulting in a strong and reliable classifier.

**LGBM Classifier:**

Light Gradient Boosting Machine, a high-performance machine learning algorithm for the classification tasks. It uses gradient boosting and focuses on efficiency by optimizing leaf-wise tree growth. This approach makes LightGBM faster and memory-efficient, making it suitable for large datasets and complex models.

## **3.10 Model Training**

The chosen machine learning algorithms undergo training using the preprocessed dataset. This process entails the division of the data into training and validation sets, followed by the application of the selected algorithms to acquire an understanding of the patterns and relationships inherent within the dataset.

## **3.11 Model Evaluation**

Trained models are evaluated using appropriate evaluation metrics like accuracy, recall, precision, F1-score, and area under ROC curve. This step assesses performance of the models in predicting CKD and helps in selecting the best-performing algorithm(s).

# **4. Results**

As per national library of medicine (India), it is estimated that out of 10 people suffering from early chronic kidney disease 9 of them don’t even know that they have CKD. There may be a lot of reasons this, it could be because of lack of awareness, resources, no frequent check-ups etc., Hence, I am trying to leverage machine learning to predict the CKD using patient’s data which can be obtained by performing few tests. The complete analysis and the findings of the project have been mentioned below.

## **4.1 Preparing Data**

I have obtained the data from the UCI’s machine learning repository, and I will load it on VScode to begin our analysis using python language. I will be using jupyter kernel on VS code.

First, I will load few of the necessary modules, the remaining modules will be loaded when those required:

A screen shot of a computer program

Description automatically generated

The Data is then loaded into the data frame and the head of the data displaying first 5 rows is given below:

A screen shot of a computer

Description automatically generated

There are a lot of missing values in the data and from the above image, I can see that the missing values are denoted by ‘?’. To avoid confusion, the ‘?’ values have been replaced by NaN values.

I had a column called "id”, but it is not important for our analysis, so I removed it and renamed all the other column headers to make the table more readable. As you can see from the image above, only abbreviations are used which are confusing. The new column names are:

A table of medical information

Description automatically generated with medium confidenceA table with text on it

Description automatically generated

Age, blood\_pressure, blood\_glucose, blood\_urea, serum\_creatinine, sodium, potassium, haemoglobin, packed\_cell\_volume, white\_blood\_cells\_count, and red\_blood\_cell\_count are all supposed to be numeric according to the data description; however, it's possible that when the raw arff file was converted to a csv file, all the features were changed to object data types. Consequently, the aforementioned features are converted to a numeric data type.

Two new variables cat\_cols and num\_cols containing all the object features and all the numeric features have been created respectively. I then check for presence of any mistypes in the nominal features.

A screen shot of a computer code

Description automatically generated

From the above image, I can see that the nominal columns do not have typing errors and all the records are correct. If there are any typos, I may use the replace function to correct the wrong records.

A screenshot of a computer program

Description automatically generated

During literature review, I observed few instances where numeric data was showing as strings, hence, I cross checked and confirmed that our data doesn’t have any such errors. It is easy to identify any strings present in numeric columns as they are denoted in quotes. However, from the above image, I can see that all numeric features have numeric data.

## **4.2 Exploring the Data**

Descriptive statistics serves as an initial step in exploring and summarizing the dataset. This information will be valuable in building a foundation for further analysis and interpretation of the data. It provides a way to organize and understand the data by using various numerical measurements that describe its central tendency, variability, and distribution. Here’s the descriptive statistics of our dataset:

A screen shot of a black and white screen

Description automatically generated

A collage of green and black graphs

Description automatically generated

There are some features like blood\_pressure, albumin, serum\_createnine, blood\_glucose, blood\_urea, and white\_blood\_cells\_count that show incredibly far-off outlier values. The procedures of statistical analysis and modelling can be considerably impacted by these outliers, thereby skewing the results. Maintaining the accuracy and dependability of the analysis necessitates taking the proper action with regard to these outliers.

I noticed observe multimodal distribution as well in specific gravity, this is mainly because the values appear to be discrete, and histograms tend to do this for such values.

The numeric feature distributions in the dataset display a variety of shapes. While some properties closely approximate a normal distribution, others are severely skewed. This suggests that each feature needs to be addressed separately during analysis due to its unique properties even in the absence of the problem of missing values. This specific strategy is necessary for a thorough and accurate examination. I will scale these values to ensure that all the features lie in the same scale which in turn reduce the chance of outlier occurrences as well.

Many features have a significant percentage of missing values. The traditional imputation method that employs measures of central tendency, is inappropriate in such circumstances. These measures potentially skew the feature distributions when used to impute missing data. Alternative imputation methods or tactics that take into account the distinctive properties of these elements will be required.

Despite the fact that some features like blood\_pressure and albumin have discrete values, they should be considered as continuous variables due to them being biological measures. The measurement technique used is probably to blame for their discrete look.

In conclusion, the dataset provides a complex situation in which outliers, discrete-to-continuous considerations, missing data, and different distribution shapes all work together to affect the data pretreatment and analysis methodologies. To guarantee that the analytic findings truly reflect the underlying patterns and characteristics of the dataset, it is essential to approach these problems in the right way.

A screenshot of a graph

Description automatically generated

Red\_blood\_cell and pus\_cell seems to have higher number of missing values when compared to other categorical features. Our target variable, class has no missing values which could prove valuable. From the above countplots, I can see that classes are not balanced (equally distributed), It would be interesting to see their correlation with our target variable which might highlight if anyone these indicate the presence of chronic kidney disease strongly.

Better representation of outliers using boxplots:

A screenshot of a computer

Description automatically generated

From the above boxplots I can see noticeable amount outliers in blood\_pressure, sugar, blood\_glucose, blood\_urea, serum\_creatinine, sodium, potassium and wbc count. In medical field it is quite common to have outliers as it indicates the presence of a disease or could be because of the treatment being received by the patient. For example, the normal potassium level is 3.5 – 5.5 mEq/L (Rastegar, 1990), however, the outliers in potassium feature have values near 40. These outlier value could be due to blood transfusion, Addison disease(rare), kidney failure etc., Hence, it is better not to assume these values as incorrect observations rather to treat them as valid outliers.

**Correlation Analysis:**

In this vital step, I will understand how our features are correlated among themselves and with the target variables using Pearson correlation.

A graph of numbers and letters

Description automatically generated with medium confidence

In medical field, correlation thresholds vary based on the context and type of data, it is also important to consult a domain expert to understand the thresholds. In our case, I will be considering commonly used thresholds: 0 – 0.3 as low correlation, 0.3 to 0.7 moderate correlation and >0.7 as high correlation, here are the findings:

High positive correlation is observed between:

1. blood glucose, sugar

2. packed cell volume, haemoglobin

3. red blood cell count, haemoglobin

4. rbc count, packed cell volume

High negative correlation is observed in:

1. specific gravity, target

2. haemoglobin, target

3. packed cell volume, target

4. rbc count, target

Visualising correlated variables is a great way to understand the patterns between them, since all positive correlated variables are numerical variables, I will visualize them using scatter plots and understand their patterns.

1. Scatter plot between blood glucose and sugar:

A graph of blood glucose

Description automatically generated

To understand how their correlation effects the occurrence of chronic kidney disease, I have compared their relationship with our target variable, it appears that only patients with 0 sugar and around 100 blood glucose level are not impacted by chronic kidney disease and rest all are impacted.

2. Scatter plot between packed cell volume and haemoglobin:

A graph showing the amount of blood in the kidney

Description automatically generated with medium confidence

Here I can observe linearity, this looks like a case of causation, as the packed cell volume increases, the haemoglobin volume increases as well. The interesting detail is that high number of people with larger haemoglobin content and packed cell volume aren’t impacted by the disease.

3. Scatter plot between red blood cell count and haemoglobin:

A diagram of a number of dots

Description automatically generated

This almost looks like the previous relation I studied; the haemoglobin count seems to be increasing with the increase in rbc. Since the features aren’t scaled yet, there’s an outlier with rbc cell count 8. The interesting takeaway from the current and earlier observation is that people with haemoglobin values more than 13 do not appear to have the disease.

4. Scatter plot between red blood cell count and packed cell volume:

A diagram of a number of dots

Description automatically generated

This looks identical to its predecessor as well. The packed cell volume tends to increase with the increase in rbc count and higher values seems to not have the chronic kidney disease.

Now, lets see the visualizations of negatively correlated features with our target variable. I have used violin plots and here are the results.

1.Violin plot of specific gravity and target:

A diagram of a kidney disease

Description automatically generated

From the above violin plot, I can see the distribution of specific gravity based on the classes in our target variable. While the specific gravity of people with CKD seems to range throughout, the specific gravity of people without CKD seems to be concentrated from 1.020 to 1.025 with high density.

2. Violin plot of haemoglobin and target:A diagram of a disease

Description automatically generated with medium confidence

The haemoglobin of people with CKD seems to have gaussian distribution, as we can see that the highest density is at its median. The haemoglobin level of the population without CKD ranges between 12.5 and 17.5, with the highest density at 15.

3. Violin plot of packed cell volume and target:A blue and orange rhombuses

Description automatically generated

The packed cell volume of people with CKD seems to be normally distributed as well. The haemoglobin level of the population without CKD ranges between 40 and 55. This could explain why there is a negative correlation between them, as the higher the value of packed cell volume gets, the density of CKD decreases and the not-CKD density increases.

4. Violin plot of rbc count and target:

A blue and orange rhombuses

Description automatically generated

Because of the existence of one extreme value (eight), the rbc count plot with CKD extends until 8. If we exclude this, the normal range is 2–6. The RBCC without CKD ranges from 4–7. Though there appears to be an overlap, the densities vary hugely. The higher values show no CKD with higher density.

## **4.3 Feature Encoding**

As there are categorical variables in our data, including our target variables, I need to encode them. all categorical variables have only 2 classes and one hot encoding is best in this scenario as I will not be encountering the curse of dimensionality error. Here is our data set after performing one hot encoding.

A screen shot of a computer

Description automatically generated

## **4.4 Imputing Missing Values with KNN Imputer**

Before K-Nearest Neighbors (KNN) imputing the missing values, transformation should be carried out to guarantee that the imputation procedure is based on precise distances between data points. In order to determine the nearest neighbors and substitute missing values using their values, KNN imputation uses the idea of the distance between data points called as Euclidean distance. It's crucial to transform the features prior to KNN imputation.

I have transformed the data using the standard scaler and then imputed the missing values with KNN imputer technique considering 8 neighbors.

A graph with green and white lines

Description automatically generated with medium confidence

The above image shows the missing values before KNN imputation was performed and the below image shows the amount of missing values after the imputation was done.

A graph with green and white stripes

Description automatically generated

## **4.5 Hypothesis Testing**

This part delves into the creation of hypotheses, which act as the cornerstones of our study. Our investigation is guided by hypotheses, which serve as testable predictions with the purpose of illuminating the connections between important factors. I have located gaps, inconsistencies, and unexplored areas that motivate the formulation of hypotheses through a thorough examination of the work done so far. These theories not only offer a well-organised framework for study but also open the door to empirical verification. I will provide a thorough set of hypotheses as we begin this investigation in an effort to support our search for significant findings and expand the body of knowledge in the area.

I could have performed the hypothesis testing during our exploratory data analysis, however, imputing missing values before hypothesis testing is generally considered beneficial in statistical analysis due to its impact on statistical accuracy, data integrity, and robustness. It enhances the dataset's completeness, minimizing potential biases, and reduces variability, leading to more accurate estimations of statistical parameters and test statistics. By maximizing data completeness, I can draw conclusions confidently from our hypothesis tests, contributing to stronger conclusions.

While the crosstabs between categorical features and target variables don’t highlight any important observation, the numerical features show completely compelling observations which gives rise to a lot of hypotheses. Let's test them and see what we can find out from them. I am using the imputed data for more accurate results.

A screenshot of a graph

Description automatically generated

**1. Specific Gravity and CKD:**

From the plot, we can see that lower the specific gravity, the higher the chance of having the disease, this matches with the negative correlation we have observed during our correlation analysis.

Null hypothesis(H0):

That there is no significant difference in occurrence of CKD with respect to different ranges of specific gravity.

Result:

P-value of less than 0.05 was obtained through Chi-square test which made us to reject the null hypothesis. This demonstrates that specific gravity levels have a significant impact on the occurrence of CKD.

A screen shot of a computer program

Description automatically generated

**2. Albumin and CKD**

According to the boxen plot, albumin levels for people without CKD tend to range from 0 to 4, but this is not always the case when CKD patients are taken into account. For a CKD scenario, the albumin level appears to be concentrated at 0. A non-parametric test like the Mann Whitney U test would be appropriate.

Null Hypothesis (H0):

No significant change in albumin levels between the two groups (noCKD and CKD) is our null hypothesis (H0). The vice versa is of alternative hypothesis(Ha)

Result:

The hypothesis test resulted in a p-value < 0.05, which highlights that there is statistical significance between the albumin levels of two groups.

**A screen shot of a computer program

Description automatically generated**

**3. Sugar and CKD**

Most of us are already aware that sugar is crucial to kidney health and that people with diabetes are more likely than others to have renal failure. The plot suggests that the relationship between sugar and CKD and albumin and CKD is identical. Although noCKD has a sugar level range of 0 to 4, the higher density is between 0 and 1. Group with CKD displays a sugar level of 0. The Mann Whitney U test would be acceptable because this is also non-parametric.

Null Hypothesis (H0): No significant change in sugar levels between the two groups (noCKD and CKD).

Result:

The hypothesis test resulted in a p-value < 0.05, which means that there is statistical significance between the sugar levels of the two groups.

A screen shot of a computer program

Description automatically generated

**4. Blood\_glucose and CKD**

The blood glucose levels vary from 100 to 400, with a higher density between 100 and 200 for no-CKD observations, while the CKD observations vary between 100 to 150. Though the densities are concentrated at lower values, it would be interesting to see how blood glucose is impacting the disease. Since there are outliers in our original data for blood glucose, Mood's median test is one of the appropriate tests.

Null Hypothesis (H0):

No significant change in blood glucose levels between the two groups (noCKD and CKD) is our null hypothesis (H0). The vice versa is of alternative hypothesis(Ha).

Result:

The hypothesis test resulted in a p-value < 0.05, which highlights that there is statistical significance between the blood glucose levels of CKD and noCKD groups.

A screen shot of a computer program

Description automatically generated

**5. Blood\_urea and CKD**

The blood urea levels vary from 40 to 200, with a higher density between 40 and 100 for no-CKD observations, while the CKD observations vary between 25 to 50. There appears to be a slight overlap with higher densities in both, it would be interesting to see how blood glucose is impacting the disease. Since there are outliers in blood urea levels as well, I will go ahead with the Mood's median test.

Null Hypothesis (H0):

No significant change in blood urea levels between the two groups (noCKD and CKD) is our null hypothesis (H0). The vice versa is of alternative hypothesis(Ha)

Result:

The hypothesis test resulted in a p p-value < 0.05, which indicates that there is statistical significance between the blood urea levels of CKD and noCKD groups.

A screen shot of a computer program

Description automatically generated

**6. Serum\_creatinine and CKD**

The serum creatinine levels vary from 0 to 20, with a higher frequency between 0 and 10 for no-CKD observations, while the CKD observations are concentrated at 0. Since this indicates non-parametric distribution, it is better to go with the Mann-Whitney U test to understand how these features are impacted.

Null Hypothesis (H0):

No significant change in serum creatinine levels between the two groups (noCKD and CKD) is our null hypothesis (H0). The vice versa is of alternative hypothesis(Ha)

Result:

The hypothesis test resulted in a p-value below 0.05, which shows that there is statistical significance between the serum creatinine levels of CKD and noCKD groups.

A screen shot of a computer

Description automatically generated

**7. Sodium and CKD**

The sodium levels vary from 120 to 150, with a higher density closer to 140 for no-CKD observations, while the CKD observations vary between 140 to 150. There appears to be a slight overlap with higher densities in both, the interesting observation so far is that there is a very less overlap between CKD and noCKD groups suggesting a strong causation. Since there are is a extreme outlier in sodium levels as well, I will go ahead with the Mood's median test.

Null Hypothesis (H0):

No significant change in sodium levels between the two groups (noCKD and CKD) is our null hypothesis (H0). The vice versa is of alternative hypothesis(Ha)

Result:

The hypothesis test resulted in a p-value below 0.05, which shows that there is statistical significance between the sodium levels of CKD and noCKD groups.

A computer screen shot of a program

Description automatically generated

**8. Haemoglobin and CKD**

The haemoglobin levels for noCKD group ranges from 6-16 while the same levels for CKD group are ranging from 14-18. The distribution seems approximately normal distribution, I can use two sample t-test comparing means of both of these features. Though the plot suggests they are different from each other, it is always preferred to confirm the same statistically.

Null Hypothesis (H0):

No difference in haemoglobin levels between noCKD and CKD groups is our null hypothesis (H0). The vice versa is of the alternative hypothesis(Ha)

Result:

The hypothesis test resulted in a p-value < 0.05, which shows that there is statistical significance between the haemoglobin levels of noCKD and CKD groups.

A screen shot of a computer program

Description automatically generated

**9. Blood\_glucose, sugar and CKD**

During correlation analysis, I noticed strong correlation between blood glucose levels and sugar. While I plotted them based on the CKD feature, I observed that the higher values tend to be attributed to noCKD. Therefore, it would be a great find to understand their relation. Since both of these are skewed, Kruskal Wallis test would be suitable.

Null Hypothesis (H0):

There is no significance association between the levels of blood glucose, sugar and CKD. The alternate hypothesis (Ha) will be the opposite of our null hypothesis.

Result:

The P-value < 0.05 which suggests that there are significant associations between the levels of blood glucose, sugar levels and the disease.

A computer screen shot of a program

Description automatically generated

**10. Packed cell volume, rbc count and CKD**

During correlation analysis, I found out that there is a strong positive correlation was observed with packed cell volume and rbc count. While I plotted them based on the CKD feature, I observed that the higher values tend to be attributed to noCKD. Since both these features appear to be normally distributed, I can use two sample t test for this.

Null Hypothesis (H0):

There is no significance association between the levels of packed cell volume, rbc count and CKD. The alternate hypothesis (Ha) will be the opposite of our null hypothesis.

Result:

The P-value below 0.05 which suggests that there are significant associations between these features and they seem to impact our target variable.

A screen shot of a computer program

Description automatically generated

With this I conclude our hypothesis testing and proceed on to further analysis. Summary of our hypothesis findings is mentioned below:

A list of medical information

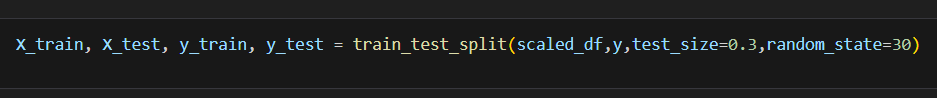
Description automatically generated with medium confidence

## **4.6 Scaling**

The need for feature scaling recurs during the model building stage, after transformations with StandardScaler and imputing missing values using K-Nearest Neighbours (KNN). Machine learning methods work well because feature scaling makes sure that the numerical attributes are consistently represented in a comparable range. Imputation filled in missing values; however, it's possible that the newly added data points have different magnitudes and distributions, which could have an impact on how well the model performs. Furthermore, some methods, such as those sensitive to feature sizes (such as distance-based and gradient descent-based algorithms), may result in biased results if not scaled properly. By scaling features once more, equity can be promoted in contributions from all characteristics, speed up algorithm convergence, and establish uniformity in their impact on the learning process, resulting in more robust and accurate model predictions. Therefore, I performed scaling using standardscaler() function again.

## **4.7 Splitting**

The ideal train-test split size depends on various factors such as the size of data, model complexity etc., Since our data size is not huge, I am using the standard 70-30 split. 70% for training the model and 30% for testing as this is recommended (Gholamy et al., 2018).



## **4.8 Model Building**

Since the problem we are trying to solve is classification, few models which are best classifiers have been selected for comparison.

### **K-Nearest Neighbor Classifier**

Our first model will be the KNN classifier model as it is widely used for classification. Unlike some other algorithms that learn explicit models from the training data, KNN makes predictions based on similar data points. I first considered default k value of 5 and built the model and the accuracy achieved was 99%, this could be a clear sign of overfitting. Therefore, various k values were considered ranging from 3 to 21 to understand how the accuracy varies. Then I plotted a graph between test prediction score and k values:

A blue squares with white text

Description automatically generated

A graph showing a line

Description automatically generated

From the above graph, it is clear that the accuracy flattened between k=9 and 15, so the optimum k-value would be 9,11,13 or 15. The scores of the model with k value 15 are given below:

A table with numbers and words

Description automatically generated

The AUC ROC score was calculated for k-value 15 as it has been determined that it is best number of neighbors for our model.

### **Decision Tree Classifier**

Decision tree is another widely used classifier. The method generates a tree-like structure in which each internal node reflects a decision based on a certain feature, leading to successive nodes (branches) until reaching leaf nodes that carry the projected conclusion. This algorithm is used in ensemble methods like random forest, gradient boosting etc.,

I have built the decision tree model by trying both entropy and gini criteria, no significant difference in the accuracy was noticed. Same observation was noticed when splitting using best and best random splits. The scores of the decision tree algorithm are impressive. Further hyperparameter tuning is not necessary.

A table with numbers and words

Description automatically generated

A graph showing a positive rate

Description automatically generated with medium confidence

A blue squares with white text

Description automatically generated

### **Random Forest Classifier**

This classifier is widely used as well. This ensemble method combines the results of many decision trees. Every single tree is constructed with random subset of data and features promoting randomness. To check how default random forest performs on our dataset, I have built a default model without providing any parameters and the accuracy and ROC AUC scores achieved are 1. Well you may think these scores are great, however in machine learning domain these scores indicate overfitting of data. Hence, the model needs to be built by supplying various parameters like entropy, max depth of each tree, min samples to be split in each node etc.,

I have tried to find the best number of trees by supplying n\_estimators values ranging from 2 to 300, the ROC AUC plot is provided below:

A graph with red and blue lines

Description automatically generated

As per the AUC ROC graph, the optimum number of tress should be less than 5, however, this is not possible as recommended range starts from 50. This may be a sign of overfitting and the algorithm may not be suited for our data. To test the accuracy, I have trained the model using recommended parameters and these are the results:

A table with text on it

Description automatically generated

The model seems to be performing perfect, however, this level of accuracy is not considered good sign and indicates overfitting. Though in one of the literature we noticed this model achieving 99.2% accuracy it is better to not consider this model for our data.

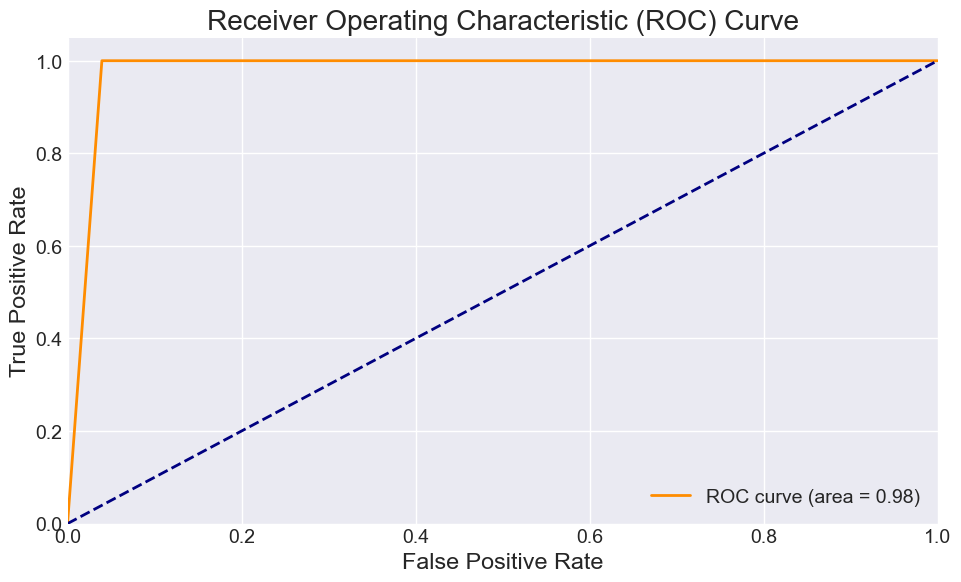
### **AdaBoost/Adaptive Boost Classifier**

This is one of the powerful ensemble algorithm which focusses on weak learners such as decision trees, by giving more weight to misclassified observations in iterations. By combining the results of weak learners to generate final prediction, with more weight attached to best performer. Thus, resulting in more accuracy, less bias and variance. I trained the model with the base estimator as decision tree algorithm, passing the same parameters as I did with decision tree (entropy criteria). The accuracy achieved is 97.5% and an AUC of 0.98. Here are the detailed metrics:

A table with numbers and letters

Description automatically generated

The models seems to be perfect and shows no sign of overfitting. The AUC ROC curve and the confusion matrix have been provided below.



A blue squares with white text

Description automatically generated

### **Gradient Boosting Classifier**

Gradient Boosting builds a strong predictive model by iteratively improving the predictions of weak learners. Unlike AdaBoost, it concentrates on minimising the residuals or mistakes of the preceding model iteration, resulting in a series of models that offer a good predictive output collectively.

I built the model with default values initially and the accuracy achieved was 100% with AUC ROC 1. This seems like an issue of overfitting which gradient boosting classifier is know to do. There, after testing various n\_estimators values 25 was found to be best. To avoid overfitting again, various parameters like max depth, validation fraction and n\_iter\_no change were given to stop early. Here are the metrics after tuning:

A table with numbers and words

Description automatically generated

The accuracy after updating the parameters is 98.3% which is excellent. The AUC and confusion matrix are provided below:

A graph with a line

Description automatically generated

A blue squares with white text

Description automatically generated

### **Stochastic Gradient Boosting Classifier**

This algorithm is an advanced ensemble technique. The "stochastic" aspect refers to the introduction of randomization during tree formation. Unlike classic gradient boosting, which uses all training samples to create each tree, stochastic gradient boosting randomly selects a portion of the training data, known as the subsample, for each tree.

Since I have already calculated the n\_estimator earlier, I have used the same value i.e, 25 again. With a subsample and max feature value 1, 98.3% accuracy is achieved with 1 AUC ROC.

A table with numbers and words

Description automatically generated

The ROC and confusion matrix are given below:

A graph with a line

Description automatically generated

A blue squares with white text

Description automatically generated

### **XgBoost Classifier**

XGBoost is known for its ability to handle complicated structured datasets. It is part of the gradient boosting architecture and is intended to improve model performance and efficiency. XGBoost combines the benefits of gradient boosting with regularisation approaches, improving predicted accuracy while lowering the danger of overfitting.

I have trained the model with best n\_estimator value of 100 and a max depth of 4 which resulted in 99% accuracy with 0.99 AUC ROC.

A table with numbers and words

Description automatically generated

A graph with a line

Description automatically generated

A blue squares with white text

Description automatically generated

### **CatBoost Classifier**

This algorithm performs well in applications ranging from classification to regression, and it is resistant to overfitting. CatBoost, unlike many other algorithms, handles category features natively, reducing dimensionality expansion. The ability of CatBoost to perform effectively with raw categorical data reduces the requirement for considerable preprocessing.

Since our dataset has lot of missing values, we had to impute using K-nearest neighbors due to which our categorical columns had to be encoded. I could use the original unencoded data, however, the missing values would impact our model. Therefore, proceeding with the processed data. Here are the metrics:

A table with numbers and words

Description automatically generated

A graph with a line

Description automatically generated

A blue squares with white text

Description automatically generated

### **Extra Trees Classifier**

This classifier uses a randomised approach to improve the performance of decision trees. It constructs numerous decision trees on various subsets of the dataset and then combines their results to draw final conclusions.

To reduce the risk of overfitting, I have performed a 5 fold cross validation with 100 estimators which resulted in 98% accuracy. Here are the detailed metrics:

A table with numbers and words

Description automatically generated

The confusion matrix is given below:

A blue squares with white text

Description automatically generated

### **LightGBM Classifier**

The LightGBM classifier (Light Gradient Boosting Machine) uses a histogram-based technique to bin continuous feature values into discrete bins, allowing for quicker training durations and lower memory consumption. LightGBM optimises the boosting process by picking the optimum split points and nodes that result in the highest loss reduction.

The model was built 100 boosting iterations and a learning rate of 2 which achieved an accuracy of 98.9%.

A table with numbers and words

Description automatically generated

A blue squares with white text

Description automatically generated

## **4.9 Performance Evaluation**

All the 10 models have been built successfully, the only problem was with the random forest model which displayed very high accuracy which could be a sign of overfitting. Hence, it is best not to consider the model as parameters made no difference to it.

Overall the remaining models performed really well and displayed high accuracy, below is the table showing the comparison of models based on metrics:

A table with numbers and symbols

Description automatically generated

Overall decision tree, Xgboost and LightGBM models performed really well and could do a great job in predicting the disease. The accuracies have been plotted below:

A graph of different colors

Description automatically generated with medium confidence

# **5. Conclusion and Recommendations**

## **5.1 Conclusion**

All of the models performed well, except for the random forest model, which appears to struggle with prediction. Despite changing the estimators in each iteration to find a good n\_estimator value, the model appears to be suffering. This could be due to a variety of the other factors. Given that it obtained 99% accuracy in one of the projects I came across in our literature review, I thought about hyperparameter optimisation for it. However, I avoided redundancy because I trained additional models that could reach the same accuracy, if not better. Aside from that, all the other models appear to have done a good job and produced good outcomes. The precision score of almost all the models is 1. In simple terms, it means that when the model predicted the disease, it was correct every time. This is vital considering the area of application.

The primary objective of project was to make use of machine learning in an area where predictive analysis could have life-changing implications. Imagine the number of people affected by chronic kidney disease without realising the fact that they have CKD until almost 80% of the kidneys lose their function. An early diagnosis could really help a lot of people.

Several hypotheses were tested during my investigation, and previously unknown facts emerged. The findings showed that albumin, specific gravity, sugar, blood urea, blood glucose, haemoglobin, sodium, and serum creatinine were all influencing the condition. The levels of these characteristics in non-CKD patients were significantly different from those in CKD patients. This indicates that a thorough evaluation by medical specialists is required to fully comprehend their relationship with the disease so that patients with aberrant levels of these traits can be cautioned or treated early. Identifying the patterns and ranges of these health characteristics in non-CKD individuals can aid in the establishment of a baseline for comparison. Furthermore, additional research and analysis are required to understand the precise pathways by which these attributes contribute to the development and progression of CKD, enabling more targeted interventions and preventive measures.

There was also a relationship between two other traits, and the increase in value of both of them resulted in no CKD. Blood glucose and sugar levels, packed cell volume and count of the RBC, packed cell volume and haemoglobin, and red blood cell count and haemoglobin were all measured. Their combined rise appears to have had an effect, as CKD was not found in these patients. It couldn't be a coincidence; these characteristics must be influencing the disease in some way. More research is needed to establish the precise mechanism by which these factors combine to prevent CKD. Understanding this link could lead to new insights and treatment techniques.

## **5.2 Recommendations**

I was introduced to numerous research publications on CKD through the literature review. Although this is excellent news, the fact that the majority of them were published using the same data suggests that there is a fundamental lack of information exchange in this industry.

Because a simple blood test could provide practically all of the attribute information we explored, a web or mobile application based on our predictive model might be built in which a user can enter their health attributes and learn if they are at risk of developing CKD. Though the model is predictive and should not be deemed 100% accurate, it may lead the user to undergo additional tests that may diagnose CKD in its early stages. This early discovery could lead to more prompt therapy and improved disease control. Furthermore, the web or mobile application could offer educational resources as well as recommendations for lifestyle modifications that can help minimise the chance of getting CKD.

Having said that, given the disease's impact and the inability to diagnose it at an early stage, a large-scale study on this subject is essential. The data on which our model was based appears to have been obtained at random, which negates its usefulness in forecasting disease. The data collected is insufficient to develop larger and more accurate predictive models. The data must be collected using correct sample techniques, such as stratified or convenient sampling, and under the supervision of a department or expert. The data obtained had several missing values, and no information was provided to explain why a few observations were missing. Even though I used KNN imputation to impute the values, a projected value can never replace a true value. This constraint has the potential to introduce bias and consequently impact the overall precision and reliability of the predictive models. Additionally, it is important to note that imputing missing values can also introduce errors and potentially impact the reliability of the predictions. Therefore, obtaining complete and high-quality data is crucial for developing robust predictive models in order to accurately predict disease outcomes.

Combining machine learning and medical disciplines has enormous promise. Although there is coordination, the rate at which it is occurring is inconvenient, and given the rapid rise of challenges, it is insufficient. Many platforms should be established so that technical professionals like data scientists, machine learning engineers etc., can engage with medical specialists like doctors, researchers etc., and information can flow freely between them. This collaboration would not only accelerate the development of innovative solutions but also ensure that advancements in machine learning are effectively applied to address pressing medical challenges. By fostering a strong interdisciplinary approach, we can unlock the full potential of machine learning to revolutionise healthcare and improve patient outcomes.

The majority of the general people appear to be uninformed about CKD and its consequences. It is critical to raise public knowledge about CKD and the benefits of early identification. Healthcare organizations and policymakers could consider public education and health screening programs. These campaigns can include providing information about the risk factors and symptoms of CKD as well as promoting the importance of maintaining a healthy lifestyle. Additionally, healthcare organisations can collaborate with community leaders and local media to reach a wider audience and ensure that the message reaches all segments of society. As was previously mentioned, these predictive models can also be used in hospitals, and they can be modified to automatically search through health databases and identify people those who are at the risk of developing the CKD. However, while doing so, ethical implications of using machine learning in healthcare must be considered. Ethical guidelines and data privacy regulations are necessary to protect patient information and ensure responsible use of AI technologies.

CKD is a global health issue. International collaboration and data sharing can amplify the impact of CKD research on a larger scale, contributing to a healthier global population. There’s a huge potential for machine learning to revolutionize healthcare, specifically in early diagnosis of the chronic kidney disease (CKD). While our models performed well overall, particularly in predicting CKD accurately, certain limitations and future directions are worth noting.

# **References**

Allen, A., Iqbal, Z., Green-Saxena, A., Hurtado, M., Hoffman, J., Mao, Q., & Das, R. (2022). Prediction of diabetic kidney disease with machine learning algorithms, upon the initial diagnosis of type 2 diabetes mellitus. *BMJ Open Diabetes Research and Care*, *10*(1), e002560. https://doi.org/10.1136/BMJDRC-2021-002560

Aworinde, O., Arif, M. S., Mukheimer, A., & Asif, D. (2023). Enhancing the Early Detection of Chronic Kidney Disease: A Robust Machine Learning Model. *Big Data and Cognitive Computing 2023, Vol. 7, Page 144*, *7*(3), 144. https://doi.org/10.3390/BDCC7030144

Bai, Q., Su, C., Tang, W., & Li, Y. (2022). Machine learning to predict end stage kidney disease in chronic kidney disease. *Scientific Reports 2022 12:1*, *12*(1), 1–8. https://doi.org/10.1038/s41598-022-12316-z

Baidya, D., Umaima, U., Islam, M. N., Shamrat, F. M. J. M., Pramanik, A., & Rahman, M. S. (2022). A Deep Prediction of Chronic Kidney Disease by Employing Machine Learning Method. *2022 6th International Conference on Trends in Electronics and Informatics, ICOEI 2022 - Proceedings*, 1305–1310. https://doi.org/10.1109/ICOEI53556.2022.9776876

Debal, D. A., & Sitote, T. M. (2022). Chronic kidney disease prediction using machine learning techniques. *Journal of Big Data*, *9*(1), 1–19. https://doi.org/10.1186/S40537-022-00657-5/FIGURES/8

Deo, R. C. (2015). Machine Learning in Medicine. *Circulation*, *132*(20), 1920–1930. https://doi.org/10.1161/CIRCULATIONAHA.115.001593

Ebiaredoh-Mienye, S. A., Swart, T. G., Esenogho, E., & Mienye, I. D. (2022). A Machine Learning Method with Filter-Based Feature Selection for Improved Prediction of Chronic Kidney Disease. *Bioengineering 2022, Vol. 9, Page 350*, *9*(8), 350. https://doi.org/10.3390/BIOENGINEERING9080350

Gentleman, R., & Carey, V. J. (2008). Unsupervised Machine Learning. *Bioconductor Case Studies*, 137–157. https://doi.org/10.1007/978-0-387-77240-0\_10

Gholamy, A., Kreinovich, V., & Kosheleva, O. (2018). Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation. *Departmental Technical Reports (CS)*. https://scholarworks.utep.edu/cs\_techrep/1209

Islam, M. A., Majumder, M. Z. H., & Hussein, M. A. (2023). Chronic kidney disease prediction based on machine learning algorithms. *Journal of Pathology Informatics*, *14*. https://doi.org/10.1016/J.JPI.2023.100189

Jiang, T., Gradus, J. L., & Rosellini, A. J. (2020). Supervised Machine Learning: A Brief Primer. *Behavior Therapy*, *51*(5), 675–687. https://doi.org/10.1016/J.BETH.2020.05.002

Maceachern, S. J., & Forkert, N. D. (2021). Machine learning for precision medicine. *Genome*, *64*(4), 416–425. https://doi.org/10.1139/GEN-2020-0131/ASSET/IMAGES/LARGE/GEN-2020-0131F1.JPEG

*Machine Learning Algorithms - A Review | Enhanced Reader*. (n.d.).

Qin, J., Chen, L., Liu, Y., Liu, C., Feng, C., & Chen, B. (2020). A machine learning methodology for diagnosing chronic kidney disease. *IEEE Access*, *8*, 20991–21002. https://doi.org/10.1109/ACCESS.2019.2963053

Rastegar, A. (1990). Serum Potassium. *BMJ*, *2*(4743), 1341–1342. https://doi.org/10.1136/bmj.2.4743.1341-c

Saravanan, R., & Sujatha, P. (2019). A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification. *Proceedings of the 2nd International Conference on Intelligent Computing and Control Systems, ICICCS 2018*, 945–949. https://doi.org/10.1109/ICCONS.2018.8663155

Virk, H., El-Sherbini, A. H., Ul Hassan Virk, H., Wang, Z., Glicksberg, B. S., Krittanawong, C., & Kern, P. E. (2023). Machine-Learning-Based Prediction Modelling in Primary Care: State-of-the-Art Review. *AI 2023, Vol. 4, Pages 437-460*, *4*(2), 437–460. https://doi.org/10.3390/AI4020024

Wang, Q., & Zhan, Z. (2011). Reinforcement learning model, algorithms and its application. *Proceedings 2011 International Conference on Mechatronic Science, Electric Engineering and Computer, MEC 2011*, 1143–1146. https://doi.org/10.1109/MEC.2011.6025669

# **Appendix**

# Importing necessary libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import math

from sklearn.preprocessing import StandardScaler

from sklearn.impute import KNNImputer

from sklearn.pipeline import make\_pipeline

from scipy.stats import chi2\_contingency, mannwhitneyu, median\_test, ttest\_ind, kruskal

from sklearn.model\_selection import train\_test\_split

import warnings

warnings.filterwarnings(action='ignore')

# importing data

ckd\_df = pd.read\_csv(r"C:\Users\visha\Desktop\Dissertation\Chronic\_Kidney\_Disease\chronic\_kidney\_disease\_full.csv")

ckd\_df.head(5)

ckd\_df.info()

# replacing '?' with NaN values

ckd\_df.replace('?', np.nan, inplace=True)

ckd\_df = ckd\_df.drop('id',axis=1)

ckd\_df.columns = ['age','blood\_pressure','specific\_gravity','albumin','sugar','red\_blood\_cells','pus\_cell','puss\_cell\_clumps','bacteria',

'blood\_glucose','blood\_urea','serum\_creatinine','sodium','potassium','haemoglobin','packed\_cell\_volume',

'white\_blood\_cells\_count','red\_blood\_cell\_count','hypertension','diabetes\_mellitus','coronary\_artery\_disease',

'appetite','pedal\_edema','anemia','chronic\_kidney\_disease']

ckd\_df.columns

ckd\_df.info()

#Since all the features are showing as object data type, converting necessary features to numeric as they should be

columns\_to\_convert = ['age', 'blood\_pressure', 'specific\_gravity', 'albumin','sugar','blood\_glucose', 'blood\_urea', 'serum\_creatinine',

'sodium', 'potassium', 'haemoglobin', 'packed\_cell\_volume',

'white\_blood\_cells\_count', 'red\_blood\_cell\_count']

ckd\_df[columns\_to\_convert] = ckd\_df[columns\_to\_convert].apply(pd.to\_numeric, errors='coerce')

ckd\_df.info()

# creating categorical columns and numerical columns variables for further use

cat\_cols = []

for col in ckd\_df.columns:

if ckd\_df[col].dtype == 'object':

cat\_cols.append(col)

num\_cols = []

for col in ckd\_df.columns:

if ckd\_df[col].dtype != 'object':

num\_cols.append(col)

# checking for unique values in each feature to understand if data needs any further processing

for col in cat\_cols:

print(f'{col} feature has {ckd\_df[col].unique()} values')

for col in num\_cols:

print(f'{col} feature has {ckd\_df[col].unique()} values')

# Exploratory Data Analysis

#descriptive statistics

ckd\_df.describe()

import matplotlib.style as style

style.use('fivethirtyeight')

n\_rows, n\_cols = (7,2)

figure, axes = plt.subplots(nrows=n\_rows, ncols=n\_cols,figsize=(20, 50))

figure.suptitle('\nDistributions of Numerical Features', fontsize=60)

for index, column in enumerate(num\_cols):

i,j = (index // n\_cols), (index % n\_cols)

miss\_perc="%.2f"%(100\*(1-(ckd\_df[column].dropna().shape[0])/ckd\_df.shape[0]))

collabel=column+"\n({}% is missing)".format(miss\_perc)

fig=sns.distplot(ckd\_df[column], color="g", label=collabel, norm\_hist=True,

ax=axes[i,j], kde\_kws={"lw":4})

fig=fig.legend(loc='best', fontsize=18)

axes[i,j].set\_ylabel("Probability Density",fontsize='medium')

axes[i,j].set\_xlabel(None)

plt.show()

style.use('seaborn-darkgrid')

n\_rows, n\_cols = (6, 2)

figure, axes = plt.subplots(nrows=n\_rows, ncols=n\_cols, figsize=(30, 50))

figure.suptitle('\nCountplots of Categorical Features', fontsize=60)

for index, column in enumerate(cat\_cols):

i, j = index // n\_cols, index % n\_cols

miss\_perc = "%.2f" % (100 \* (1 - (ckd\_df[column].dropna().shape[0]) / ckd\_df.shape[0]))

collabel = column + "\n({}% is missing)".format(miss\_perc)

fig = sns.countplot(x=column, data=ckd\_df, label=collabel, palette=sns.cubehelix\_palette(rot=-.35, light=0.85, hue=1),

ax=axes[i, j])

axes[i, j].set\_title(collabel, fontsize=30)

axes[i, j].set\_xlabel(None)

axes[i, j].set\_ylabel("Count", fontsize=20)

axes[i, j].set\_xticklabels(axes[i, j].get\_xticklabels(), fontsize=28)

plt.show()

ckd\_df.info()

ckd\_df.head()

#boxplots to understand more about outliers

n\_rows, n\_cols = (7,2)

fig, axes = plt.subplots(nrows=n\_rows,ncols=n\_cols,figsize=(18,25))

for index,col in enumerate(num\_cols):

[i,j] = index//n\_cols, index%n\_cols

fig = sns.boxplot(data=ckd\_df,x=col,ax=axes[i,j],notch=True,flierprops={"marker": "x"},color="#B4E2B0")

plt.tight\_layout()

#converting target variable to numeric for correlation analysis

df = pd.DataFrame()

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

df[num\_cols] = ckd\_df[num\_cols]

df["target"] = ckd\_df["chronic\_kidney\_disease"].map({'ckd':1,'notckd':0})

df\_cor = df.corr()

mask = np.triu(np.ones\_like(df\_cor))

sns.heatmap(data=df\_cor,mask=mask,annot=True,linewidths=3,fmt='.2f')

plt.tight\_layout

###high positive correlations:

1. blood-glucose and sugar

2. packed\_cell\_volume and haemoglobin

3. red\_blood\_cell\_count and haemoglobin

4. red\_blood\_cell\_count and packed\_cell\_volume

high negative correlations:

1. specific\_gravity and target

2. haemoglobin and target

3. packed\_cell\_volume and target

4. red\_blood\_cell\_count and target

#violin plots and scatter plot to understand the correlation:

def violin(col):

fig = sns.violinplot(ckd\_df, y=col, x="chronic\_kidney\_disease", box=True)

return plt.show()

def scatter(col1, col2):

fig = sns.scatterplot(ckd\_df, x=col1, y=col2, hue="chronic\_kidney\_disease")

return plt.show()

# 1. blood\_glucose and sugar

scatter("blood\_glucose","sugar")

# 2. packed\_cell\_volume and haemoglobin

scatter("packed\_cell\_volume","haemoglobin")

# 3. red\_blood\_cell\_count and haemoglobin

scatter("red\_blood\_cell\_count","haemoglobin")

#4. red\_blood\_cell\_count and packed\_cell\_volume

scatter("red\_blood\_cell\_count","packed\_cell\_volume")

# 1. specific\_gravity and target

violin("specific\_gravity")

# 2. haemoglobin and target

violin("haemoglobin")

# 3. packed\_cell\_volume and target

violin("packed\_cell\_volume")

# 4. red\_blood\_cell\_count and target

violin("red\_blood\_cell\_count")

#one hot encoding

ohe\_data = pd.get\_dummies(ckd\_df,drop\_first=True,prefix\_sep=':',dtype=int,dummy\_na=False)

ohe\_data.isna().sum()

#the missing values have been converted in to 0s for cat columns, converting them back

names={}

for name in ckd\_df.columns:

for ohe in ohe\_data.columns:

if name+':' in ohe and name in cat\_cols:

names[name]=ohe

for i in range(400):

if type(ckd\_df.loc[i,name])!=str:

if math.isnan(ckd\_df.loc[i,name]):

ohe\_data.loc[i,ohe]=ckd\_df.loc[i,name]

ohe\_data.isna().sum()

ohe\_cat\_cols = list(ohe\_data.columns.values)

ohe\_num\_cols = ohe\_cat\_cols[:14]

ohe\_cat\_cols = ohe\_cat\_cols[14:]

ohe\_data.iloc[:,14:]

pipe = make\_pipeline(StandardScaler())

print(pipe)

df = [ohe\_data]

df1 = pd.DataFrame(pipe.fit\_transform(ohe\_data),columns = ohe\_data.columns)

df.append(df1)

df

import missingno as msno

msno.bar(ckd\_df,color="turquoise",sort="ascending")

#KNN imputation

imputer = KNNImputer(weights='distance',n\_neighbors=8)

rrr = [ohe\_data.to\_numpy()]

rrr.append(imputer.fit\_transform(df[1]))

arr = [rrr[0]]

for i in range(1,len(rrr)):

arr.append(pipe[i-1].inverse\_transform(rrr[i]))

imputed\_df=[]

for i in range(len(arr)):

imputed\_df.append(pd.DataFrame(arr[i],columns=ohe\_data.columns))

imputed\_df

ohe\_data = imputed\_df[1].copy()

ohe\_data.dropna().shape #no missing values

msno.bar(ohe\_data,color="aquamarine")

ohe\_data.dropna().shape

# Hypothesis Testing

#plotting categorical variables with target variable

figure, axes = plt.subplots(6, 2,figsize=(50, 100))

figure.suptitle('\nCrossTabs of Categorical Variables with Target Variable', fontsize=70)

for index, col in enumerate(cat\_cols):

i,j = (index // 2), (index % 2)

sns.heatmap(pd.crosstab(ckd\_df[col],ckd\_df['chronic\_kidney\_disease']),

ax=axes[i,j],

square='True',

cbar=False,

annot=True,

annot\_kws={'fontsize':90},

fmt='d')

axes[i,j].set\_xlabel("Disease", fontsize=90)

axes[i,j].set\_ylabel(col,fontsize=90)

axes[i,j].set\_yticklabels(axes[i,j].get\_yticklabels(),fontsize=50)

axes[i,j].set\_xticklabels(["No CKD","CKD"],fontsize=50)

plt.show()

nothing interesting with the categorical variables. lets see if the numerical variable have any effect.

#plotting numerical variables with target variable

figure, axes = plt.subplots(7, 2,figsize=(50, 100))

figure.suptitle('\nBox Plots of Numerical Variables with Target Variable', fontsize=100)

for index, col in enumerate(num\_cols):

i,j = (index // 2), (index % 2)

sns.boxenplot(data=ckd\_df,y=ckd\_df[col],x=ckd\_df['chronic\_kidney\_disease'],

ax=axes[i,j],color='aquamarine')

axes[i,j].set\_xlabel("Disease", fontsize=90)

axes[i,j].set\_ylabel(col,fontsize=90)

axes[i,j].set\_yticklabels(axes[i,j].get\_yticklabels(),fontsize=30)

axes[i,j].set\_xticklabels(["No CKD","CKD"],fontsize=30)

plt.show()

ohe\_data.columns

# Hypothesis 1 - Chi-Square Test for Impact of Specific Gravity on Chronic Kidney Disease

contingency\_table = pd.crosstab(ohe\_data['specific\_gravity'], ohe\_data['chronic\_kidney\_disease:notckd'])

chi2, p\_value, dof, expected = chi2\_contingency(contingency\_table)

if p\_value < 0.05:

print(" Reject the null hypothesis")

else:

print("Fail to reject the null hypothesis")

# Hypothesis 2 - No significant difference in albumin levels between noCKD and CKD groups

albumin\_noCKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 1]['albumin']

albumin\_CKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 0]['albumin']

# Perform Mann-Whitney U test

statistic, p\_value = mannwhitneyu(albumin\_noCKD, albumin\_CKD, alternative='two-sided')

if p\_value < 0.05:

print(" Reject the null hypothesis")

else:

print("Fail to reject the null hypothesis")

# Hypothesis 3 - No significant difference in sugar levels between noCKD and CKD groups

sugar\_noCKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 1]['sugar']

sugar\_CKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 0]['sugar']

# Perform Mann-Whitney U test

statistic, p\_value = mannwhitneyu(sugar\_noCKD, sugar\_CKD, alternative='two-sided')

if p\_value < 0.05:

print(" Reject the null hypothesis")

else:

print("Fail to reject the null hypothesis")

#Hypothesis 4 - No significant difference in blood glucose levels between nockd and ckd

blood\_glucose\_noCKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 1]['blood\_glucose']

blood\_glucose\_CKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 0]['blood\_glucose']

# Perform Mood's Median Test

statistic, p\_value, medians, table = median\_test(blood\_glucose\_noCKD, blood\_glucose\_CKD)

if p\_value < 0.05:

print(" Reject the null hypothesis")

else:

print("Fail to reject the null hypothesis")

#Hypothesis 5 - No significant difference in blood urea levels between nockd and ckd

blood\_urea\_noCKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 1]['blood\_urea']

blood\_urea\_CKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 0]['blood\_urea']

# Perform Mood's Median Test

statistic, p\_value, medians, table = median\_test(blood\_urea\_noCKD, blood\_urea\_CKD)

if p\_value < 0.05:

print(" Reject the null hypothesis")

else:

print("Fail to reject the null hypothesis")

# Hypothesis 6 - No significant difference in serum creatinine levels between noCKD and CKD groups

serum\_noCKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 1]['serum\_creatinine']

serum\_CKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 0]['serum\_creatinine']

# Perform Mann-Whitney U test

statistic, p\_value = mannwhitneyu(serum\_noCKD, serum\_CKD, alternative='two-sided')

if p\_value < 0.05:

print(" Reject the null hypothesis")

else:

print("Fail to reject the null hypothesis")

#Hypothesis 7 - No significant difference in sodium levels between nockd and ckd

sodium\_noCKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 1]['sodium']

sodium\_CKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 0]['sodium']

# Perform Mood's Median Test

statistic, p\_value, medians, table = median\_test(sodium\_noCKD, sodium\_CKD)

if p\_value < 0.05:

print(" Reject the null hypothesis")

else:

print("Fail to reject the null hypothesis")

#Hypothesis 8 - No significant difference in haemoglobin levels between nockd and ckd

haemoglobin\_noCKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 1]['haemoglobin']

haemoglobin\_CKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 0]['haemoglobin']

# Perform two-sample t-test

statistic, p\_value = ttest\_ind(haemoglobin\_noCKD, haemoglobin\_CKD)

if p\_value < 0.05:

print(" Reject the null hypothesis")

else:

print("Fail to reject the null hypothesis")

#Hypothesis 9 - blood\_glucose, sugar and the target variable

# for blood\_glucose and chronic\_kidney\_disease

kw\_statistic\_blood\_glucose, p\_value\_blood\_glucose = kruskal(blood\_glucose\_noCKD, blood\_glucose\_CKD)

# Sugar and chronic\_kidney\_disease

kw\_statistic\_sugar, p\_value\_sugar = kruskal(sugar\_noCKD, sugar\_CKD)

print(f"Kruskal-Wallis - Blood Glucose: H = {kw\_statistic\_blood\_glucose}, p = {p\_value\_blood\_glucose}")

print(f"Kruskal-Wallis - Sugar: H = {kw\_statistic\_sugar}, p = {p\_value\_sugar}")

#Hypothesis 10 - packed\_cell\_volume, red\_blood\_cell\_count and CKD

pcv\_noCKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 1]['packed\_cell\_volume']

pcv\_CKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 0]['packed\_cell\_volume']

rbc\_noCKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd'] == 1]['red\_blood\_cell\_count']

rbc\_CKD = ohe\_data[ohe\_data['chronic\_kidney\_disease:notckd']==0]['red\_blood\_cell\_count']

# Perform two-sample t-test for packed\_cell\_volume and chronic\_kidney\_disease

t\_statistic\_pcv, p\_value\_pcv = ttest\_ind(pcv\_noCKD, pcv\_CKD)

# Perform two-sample t-test for red\_blood\_cell\_count and chronic\_kidney\_disease

t\_statistic\_rbc, p\_value\_rbc = ttest\_ind(rbc\_noCKD, rbc\_CKD)

# Print results

print(f"Two-sample t-test - Packed Cell Volume: t = {t\_statistic\_pcv}, p = {p\_value\_pcv}")

print(f"Two-sample t-test - Red Blood Cell Count: t = {t\_statistic\_rbc}, p = {p\_value\_rbc}")

# Scaling

#scaling using standard scaler

X = ohe\_data.drop("chronic\_kidney\_disease:notckd",axis=1,inplace=False)

y = ohe\_data["chronic\_kidney\_disease:notckd"]

ohe\_scaled\_df = StandardScaler().fit\_transform(ohe\_data)

scaled\_df = StandardScaler().fit\_transform(X)

# Splitting Data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(scaled\_df,y,test\_size=0.3,random\_state=30)

# KNN Classifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, auc, roc\_curve

k\_values = [3, 5, 7, 9, 11, 13, 15, 17, 19, 21]

accuracy\_scores = []

for k in k\_values:

knn\_model = KNeighborsClassifier(n\_neighbors=k) # Create and fit the k-nearest neighbors model

knn\_model.fit(X\_train, y\_train)

y\_pred\_test = knn\_model.predict(X\_test)

knn\_accuracy = accuracy\_score(y\_test,y\_pred\_test) #accuracy

print(f"Results for k = {k}")

print(f"Test Accuracy of KNN is {knn\_accuracy} \n")

print(f"Confusion Matrix :- \n{confusion\_matrix(y\_test, y\_pred\_test)}\n")

print(f"Classification Report :- \n{classification\_report(y\_test, y\_pred\_test)}")

print("=" \* 50)

accuracy\_scores.append(knn\_accuracy)

plt.figure(figsize=(10, 6))

plt.plot(k\_values, accuracy\_scores, marker='o')

plt.xlabel('K-Value')

plt.ylabel('Accuracy')

plt.title('K-Value vs. Accuracy')

plt.xticks(np.arange(1, 21))

plt.grid(True)

plt.show()

#considering k=15 as optimal k-value

knn\_model = KNeighborsClassifier(n\_neighbors=15)

knn\_model.fit(X\_train, y\_train)

y\_pred = knn\_model.predict(X\_test)

knn\_accuracy = accuracy\_score(y\_test,y\_pred)

y\_pred\_probs = knn\_model.predict\_proba(X\_test)[:, 1] # Probabilities for positive class

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_probs)

roc\_auc = auc(fpr, tpr)

confusion = confusion\_matrix(y\_test,y\_pred)

print(f"AUC of ROC curve: {roc\_auc}\n")

plt.figure(figsize=(8, 6))

sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues',

xticklabels=["noCKD","CKD"], yticklabels=["noCKD","CKD"],cbar=False)

plt.title('Confusion Matrix')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.show()

## Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(criterion='entropy',splitter='random')

dt.fit(X\_train,y\_train)

y\_pred = dt.predict(X\_test)

dt\_accuracy = accuracy\_score(y\_test, y\_pred)

y\_pred\_probs = dt.predict\_proba(X\_test)[:,1]

fpr, tpr, \_ = roc\_curve(y\_test,y\_pred\_probs)

roc\_auc = auc(fpr, tpr)

confusion = confusion\_matrix(y\_test, y\_pred)

print(f"Test Accuracy is {dt\_accuracy} \n")

print(f"Confusion Matrix :- \n{confusion}\n")

print(f"Classification Report :- \n {classification\_report(y\_test, dt.predict(X\_test))}")

print(f"AUC of ROC curve: {roc\_auc}\n")

plt.figure(figsize=(10, 6))

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.legend(loc='lower right')

plt.show()

plt.figure(figsize=(8, 6))

sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues',

xticklabels=["noCKD","CKD"], yticklabels=["noCKD","CKD"],cbar=False)

plt.title('Confusion Matrix')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.show()

## Random forest

from sklearn.ensemble import RandomForestClassifier

from matplotlib.legend\_handler import HandlerLine2D

#lets see how default random forest classifier performs

rf = RandomForestClassifier()

rf.fit(X\_train,y\_train)

y\_pred = rf.predict(X\_test)

rf\_accuracy = accuracy\_score(y\_test,y\_pred)

y\_pred\_probs = rf.predict\_proba(X\_test)[:,1]

fpr, tpr, \_ = roc\_curve(y\_test,y\_pred\_probs)

roc\_auc = auc(fpr,tpr)

print(f"The accuracy score is {rf\_accuracy}")

print(f"the ROC AUC is {roc\_auc}")

#finding the optinum number of trees in forest

n\_estimators = [2,3,4,5,10,15,20,25,30,50, 100, 150, 200, 250, 300]

train\_results = []

test\_results = []

for estimator in n\_estimators:

rf = RandomForestClassifier(n\_estimators=estimator, n\_jobs=-1)

rf.fit(X\_train, y\_train)

train\_pred = rf.predict(X\_train)

fpr, tpr, \_ = roc\_curve(y\_train, train\_pred)

roc\_auc = auc(fpr, tpr)

train\_results.append(roc\_auc)

y\_pred = rf.predict(X\_test)

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred)

roc\_auc = auc(fpr, tpr)

test\_results.append(roc\_auc)

line1, = plt.plot(n\_estimators, train\_results, 'b', label="Train AUC")

line2, = plt.plot(n\_estimators, test\_results, 'r', label="Test AUC")

plt.legend(handler\_map={line1: HandlerLine2D(numpoints=2)})

plt.ylabel("AUC score")

plt.xlabel("n\_estimators")

plt.show()

#checking model with recommended parameters

rf = RandomForestClassifier(criterion = 'entropy', max\_depth = 11,

max\_features = 'auto', min\_samples\_leaf = 2,

min\_samples\_split = 3, n\_estimators = 130)

rf.fit(X\_train, y\_train)

# accuracy score, confusion matrix and classification report of random forest

rf.predict(X\_test)

rf\_accuracy = accuracy\_score(y\_test, y\_pred)

y\_pred\_probs = rf.predict\_proba(X\_test)[:, 1] # Probabilities for positive class

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_probs)

roc\_auc = auc(fpr, tpr)

print(f"Training Accuracy is {accuracy\_score(y\_train, rf.predict(X\_train))}")

print(f"Test Accuracy is {rf\_accuracy} \n")

print(f"Confusion Matrix :- \n{confusion\_matrix(y\_test, rf.predict(X\_test))}\n")

print(f"Classification Report :- \n {classification\_report(y\_test, rf.predict(X\_test))}")

print(f"AUC of ROC curve: {roc\_auc}\n")

## AdaBoost/Adaptive Boost Classifier

from sklearn.ensemble import AdaBoostClassifier

#training and fitting model, then printing scores

ada = AdaBoostClassifier(base\_estimator = dt)

ada.fit(X\_train, y\_train)

y\_pred = ada.predict(X\_test)

ada\_accuracy = accuracy\_score(y\_test, y\_pred)

y\_pred\_probs = ada.predict\_proba(X\_test)[:,1]

fpr, tpr, \_ = roc\_curve(y\_test,y\_pred\_probs)

roc\_auc = auc(fpr, tpr)

confusion = confusion\_matrix(y\_test, y\_pred)

print(f"Test Accuracy is {ada\_accuracy} \n")

print(f"Confusion Matrix :- \n{confusion}\n")

print(f"Classification Report :- \n {classification\_report(y\_test, ada.predict(X\_test))}")

print(f"AUC of ROC curve: {roc\_auc}\n")

#plotting ROC

plt.figure(figsize=(10, 6))

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.legend(loc='lower right')

plt.show()

#plotting confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues',

xticklabels=["noCKD","CKD"], yticklabels=["noCKD","CKD"],cbar=False)

plt.title('Confusion Matrix')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.show()

## Gradient Boosting Clasifier

from sklearn.ensemble import GradientBoostingClassifier

#finding the optinum number of n\_estimator

n\_estimators = [2,3,4,5,10,15,20,25,30,50]

train\_results = []

test\_results = []

for estimator in n\_estimators:

gb = GradientBoostingClassifier(n\_estimators=estimator,criterion='squared\_error')

gb.fit(X\_train, y\_train)

train\_pred = gb.predict(X\_train)

fpr, tpr, \_ = roc\_curve(y\_train, train\_pred)

roc\_auc = auc(fpr, tpr)

train\_results.append(roc\_auc)

y\_pred = gb.predict(X\_test)

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred)

roc\_auc = auc(fpr, tpr)

test\_results.append(roc\_auc)

line1, = plt.plot(n\_estimators, train\_results, 'b', label="Train AUC")

line2, = plt.plot(n\_estimators, test\_results, 'r', label="Test AUC")

plt.legend(handler\_map={line1: HandlerLine2D(numpoints=2)})

plt.ylabel("AUC score")

plt.xlabel("n\_estimators")

plt.show()

gb = GradientBoostingClassifier(n\_estimators=25, validation\_fraction=0.2, n\_iter\_no\_change=10,

learning\_rate=0.1, max\_depth=2)

gb.fit(X\_train, y\_train)

y\_pred = gb.predict(X\_test)

gb\_accuracy = accuracy\_score(y\_test, y\_pred)

y\_pred\_probs = gb.predict\_proba(X\_test)[:,1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_probs)

roc\_auc = auc(fpr, tpr)

confusion = confusion\_matrix(y\_test, y\_pred)

print(f"Test Accuracy is {gb\_accuracy} \n")

print(f"Confusion Matrix:\n{confusion}\n")

print(f"Classification Report:\n{classification\_report(y\_test, gb.predict(X\_test))}")

print(f"AUC of ROC curve: {roc\_auc}\n")

#plotting ROC

plt.figure(figsize=(10, 6))

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.legend(loc='lower right')

plt.show()

#plotting confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues',

xticklabels=["noCKD","CKD"], yticklabels=["noCKD","CKD"],cbar=False)

plt.title('Confusion Matrix')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.show()

## Stochastic Gradient Boosting Classifier

#finding the optinum number of n\_estimator

n\_estimators = [2,3,4,5,10,15,20,25,30,50,100,200,300]

train\_results = []

test\_results = []

for estimator in n\_estimators:

sgb = GradientBoostingClassifier(n\_estimators=estimator,criterion='squared\_error')

sgb.fit(X\_train, y\_train)

train\_pred = gb.predict(X\_train)

fpr, tpr, \_ = roc\_curve(y\_train, train\_pred)

roc\_auc = auc(fpr, tpr)

train\_results.append(roc\_auc)

y\_pred = gb.predict(X\_test)

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred)

roc\_auc = auc(fpr, tpr)

test\_results.append(roc\_auc)

line1, = plt.plot(n\_estimators, train\_results, 'b', label="Train AUC")

line2, = plt.plot(n\_estimators, test\_results, 'r', label="Test AUC")

plt.legend(handler\_map={line1: HandlerLine2D(numpoints=2)})

plt.ylabel("AUC score")

plt.xlabel("n\_estimators")

plt.show()

sgb = GradientBoostingClassifier(max\_depth=2, subsample=1, max\_features=1, n\_estimators=25, random\_state=42)

sgb.fit(X\_train, y\_train)

y\_pred = sgb.predict(X\_test)

sgb\_accuracy = accuracy\_score(y\_test, y\_pred)

y\_pred\_probs = sgb.predict\_proba(X\_test)[:,1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_probs)

roc\_auc = auc(fpr, tpr)

confusion = confusion\_matrix(y\_test, y\_pred)

print(f"Test Accuracy is {sgb\_accuracy} \n")

print(f"Confusion Matrix:\n{confusion}\n")

print(f"Classification Report:\n{classification\_report(y\_test, sgb.predict(X\_test))}")

print(f"AUC of ROC curve: {roc\_auc}\n")

#plotting ROC

plt.figure(figsize=(10, 6))

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.legend(loc='lower right')

plt.show()

#plotting confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues',

xticklabels=["noCKD","CKD"], yticklabels=["noCKD","CKD"],cbar=False)

plt.title('Confusion Matrix')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.show()

## XgBoost Classifier

from xgboost import XGBClassifier

xgb = XGBClassifier(objective = 'binary:logistic', learning\_rate = 0.5, max\_depth = 4, n\_estimators = 200)

xgb.fit(X\_train, y\_train)

y\_pred = xgb.predict(X\_test)

xgb\_accuracy = accuracy\_score(y\_test, y\_pred)

y\_pred\_probs = xgb.predict\_proba(X\_test)[:,1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_probs)

roc\_auc = auc(fpr, tpr)

confusion = confusion\_matrix(y\_test, y\_pred)

print(f"Test Accuracy is {xgb\_accuracy} \n")

print(f"Confusion Matrix:\n{confusion}\n")

print(f"Classification Report:\n{classification\_report(y\_test, xgb.predict(X\_test))}")

print(f"AUC of ROC curve: {roc\_auc}\n")

#plotting ROC

plt.figure(figsize=(10, 6))

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.legend(loc='lower right')

plt.show()

#plotting confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues',

xticklabels=["noCKD","CKD"], yticklabels=["noCKD","CKD"],cbar=False)

plt.title('Confusion Matrix')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.show()

## CatBoost Classifier

from catboost import CatBoostClassifier

cat = CatBoostClassifier(iterations=7,verbose=0)

cat.fit(X\_train, y\_train)

y\_pred = cat.predict(X\_test)

cat\_accuracy = accuracy\_score(y\_test, y\_pred)

y\_pred\_probs = cat.predict\_proba(X\_test)[:,1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_probs)

roc\_auc = auc(fpr, tpr)

confusion = confusion\_matrix(y\_test, y\_pred)

print(f"Test Accuracy is {cat\_accuracy} \n")

print(f"Confusion Matrix:\n{confusion}\n")

print(f"Classification Report:\n{classification\_report(y\_test, cat.predict(X\_test))}")

print(f"AUC of ROC curve: {roc\_auc}\n")

#plotting ROC

plt.figure(figsize=(10, 6))

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.legend(loc='lower right')

plt.show()

#plotting confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues',

xticklabels=["noCKD","CKD"], yticklabels=["noCKD","CKD"],cbar=False)

plt.title('Confusion Matrix')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.show()

## Extra Trees Classifier

from sklearn.ensemble import ExtraTreesClassifier

from sklearn.model\_selection import cross\_val\_predict

et = ExtraTreesClassifier(n\_estimators=100, max\_depth=5, random\_state=30)

# Implement cross-validation to control overfitting

y\_pred\_cv = cross\_val\_predict(et, X\_train, y\_train, cv=5)

# Calculate accuracy score

et\_accuracy = accuracy\_score(y\_train, y\_pred\_cv)

et.fit(X\_train, y\_train)

y\_pred = et.predict(X\_test)

y\_pred\_probs = et.predict\_proba(X\_test)[:,1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_probs)

roc\_auc = auc(fpr, tpr)

confusion = confusion\_matrix(y\_test, y\_pred)

print(f"Test Accuracy is {et\_accuracy} \n")

print(f"Confusion Matrix:\n{confusion}\n")

print(f"Classification Report:\n{classification\_report(y\_test, et.predict(X\_test))}")

print(f"AUC of ROC curve: {roc\_auc}\n")

#plotting ROC

plt.figure(figsize=(10, 6))

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.legend(loc='lower right')

plt.show()

#plotting confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues',

xticklabels=["noCKD","CKD"], yticklabels=["noCKD","CKD"],cbar=False)

plt.title('Confusion Matrix')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.show()

## LightGBM Classifier

from lightgbm import LGBMClassifier

lgbm = LGBMClassifier(learning\_rate = 2, n\_estimators=100,max\_depth=3)

# Implement cross-validation to control overfitting

y\_pred\_cv = cross\_val\_predict(lgbm, X\_train, y\_train, cv=5)

# Calculate accuracy score

lgbm\_accuracy = accuracy\_score(y\_train, y\_pred\_cv)

lgbm.fit(X\_train, y\_train)

y\_pred = lgbm.predict(X\_test)

y\_pred\_probs = lgbm.predict\_proba(X\_test)[:,1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_probs)

roc\_auc = auc(fpr, tpr)

confusion = confusion\_matrix(y\_test, y\_pred)

print(f"Test Accuracy is {lgbm\_accuracy} \n")

print(f"Confusion Matrix:\n{confusion}\n")

print(f"Classification Report:\n{classification\_report(y\_test, lgbm.predict(X\_test))}")

print(f"AUC of ROC curve: {roc\_auc}\n")

#plotting confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues',

xticklabels=["noCKD","CKD"], yticklabels=["noCKD","CKD"],cbar=False)

plt.title('Confusion Matrix')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.show()

## Performance Comparison

performance = pd.DataFrame({

'Model' : [ 'KNN', 'Decision Tree Classifier', 'Ada Boost Classifier',

'Gradient Boosting Classifier', 'Stochastic Gradient Boosting', 'XgBoost', 'Cat Boost', 'Extra Trees Classifier', 'LGBM Classifier'],

'Score' : [knn\_accuracy, dt\_accuracy, ada\_accuracy, gb\_accuracy, sgb\_accuracy, xgb\_accuracy, cat\_accuracy, et\_accuracy, lgbm\_accuracy]

})

performance = performance.sort\_values(by='Score', ascending=True).reset\_index()

plt.figure(figsize=(20, 15))

sns.barplot(data=performance, x='Model', y='Score', palette=sns.color\_palette("Blues", len(performance)))

plt.title('\nAccuracy of Various Models\n', fontsize=50)

plt.xticks(rotation=45, ha='right', fontsize=20)

plt.yticks(fontsize=20)

plt.xlabel('Model',fontsize=25)

plt.ylabel('Score',fontsize=25)

for index, row in performance.iterrows():

plt.text(index, row['Score'], round(row['Score'], 2), ha='center', va='bottom', fontsize=20)

plt.tight\_layout()

plt.show()