# Advanced Machine Learning Approaches for Chronic Kidney Disease Prediction: A Comparative and Comprehensive Study with an Ensemble Machine Learning Model

**Sagar Deep Saha[1], Vandana Kumari[2], Swarup Nandi[3]**
[1]P.G Student, [2]P.G Student, [3]Assistant Professor
[1,2,3] Department of Information Technology, Tripura University, Suryamaninagar, Tripura, India

**Abstract**
Chronic Kidney Disease (CKD) is a major health problem affecting millions of people all around the world. It is also a major burden on the healthcare system. Early detection of CKD and accurate outcome prediction are crucial. In this study, we are finding out if machine learning (ML) algorithms are efficient for the prediction of CKD using data from diagnostics and clinical. Our models, including k-Nearest Neighbors (KNN), Decision Tree (DT), Grid Search CV (DTC), Random Forest (RF), XGBoost (XGB), Logistic Regression (LR), Support Vector Machine (SVM), Gradient Boosting Classifier (GBC) and our own custom hybrid model called Ensemble Model (ENM) were used for systematic evaluation. The ENM, which combines RF and GBC, outperforms all other models according to all evaluation metrics—accuracy, precision, recall, and F1-score. A preprocessing pipeline was developed to clean, normalize, and select features for our data. The results of the research highlight how useful the ENM could be for the prediction of CKD. This research shows valuable comparative strengths of ML and presents a new one that would improve its prediction for CKD. The results add to the growing use of AI in healthcare, highlighting its potential for improving medical diagnoses.

**Keywords:** Chronic Kidney Disease (CKD), Machine Learning (ML), prediction models, data pre-processing, Ensemble Model (ENM), early diagnosis.

## 1. Introduction

CKD has become a major global health problem that largely affects millions of people and severely impairs their quality of life, progressing to end-stage renal disease if left untreated. Early detection and accurate prediction are major elements in providing timely interventions, improving outcomes, and slowing the progression of the disease. The power tools achieved in the field of ML address challenging medical issues such as predicting CKD. Data-pattern identification is the forte of ML algorithms, giving a

precise and reliable diagnosis. Such data-driven methods can be utilized by healthcare providers for better CKD diagnosis and prediction and, subsequently, better treatment strategies.

In light of this paper, ML models for CKD prediction were developed: k-Nearest Neighbors (KNN), Decision Tree (DT), Grid Search CV (DTC), Random Forest (RF), XGBoost (XGB), Logistic Regression (LR), Support Vector Machine (SVM), Gradient Boosting Classifier (GBC), and an Ensemble Model (ENM). Strong efforts have been made to explore a highly powerful model while simultaneously describing the strengths and weaknesses of every model.

It analyzes the structured dataset, which includes critical health signs like glucose levels and blood pressure. A sound preprocessing pipeline includes data cleaning, feature engineering, and scaling of the data before ML application. Performance metrics such as accuracy, precision, recall, and F1-score give an all-around framework for evaluating performance.

This research contributes to the advancement of ML in healthcare by giving relative insights into which model to use, thus helping researchers and practitioners design effective ML-based solutions to address CKD and other medical challenges.

## 2. Motivation

The management of Chronic Kidney Disease is among the biggest challenges in health around the world; it leads to expensive treatments like dialysis and transplantation. Slowing progression or improving outcomes, however, begins with early diagnosis, and it is precisely the early diagnosis stage where traditional approaches fail. Powerful tools in ML can be employed to analyze complicated data and thereby improve diagnostic accuracy. This article evaluates multiple techniques of ML using ensemble models and explores their applications to other chronic diseases.

## 3. Objectives

This research will evaluate the performance of various machine learning algorithms to predict CKD and identify the two best-performing models, which are the Random Forest Classifier (RF) and the Gradient Boosting Classifier (GBC). Based on this, it aims to design a new ensemble model (ENM) based on the strength of these models and test it against individual algorithms. The main objectives are:

- Development of a sound preprocessing pipeline for cleaning, normalizing, and preparing data.
- Predictive performance evaluation by metrics such as accuracy, precision, recall, and F1-score.
- Systematic comparison of strengths and limitations of algorithms used.
- The practical impact of ML-driven early diagnosis in CKD.

This study drives innovation for data-driven healthcare, leading toward early CKD intervention.

## 4. Literature survey

In this work, several approaches for the prediction of CKD using ML have been discussed. Various algorithms are practiced and their accuracy and reliability are evaluated. ML has been an essential tool in medicine wherein data analysis and early diagnosis improve patient outcomes. In CKD prediction, ML approaches have found utility in classifying the data and thereby aid in consequent timely interventions.

| Ref No. | Author Name | Year | Method | Key Findings |
|---|---|---|---|---|
| [1] | VarshaBansa, Dr. RituSindhu | 2024 | Supervised ML classifiers | Predictive analytics based on ML is efficient in the identification of CKD. |
| [2] | DibabaAdebaDebal, TilahunMelakSitote | 2022 | Random Forest with Recursive Feature Elimination | RF with cross-validation performs best in multiclass and binary CKD classification. |
| [3] | Jing Xiao et al. | 2019 | Logistic Regression, Statistical Models, Neural Networks | Logistic regression performed best with AUC = 0.873. |
| [4] | AfiaFarjana et al. | 2024 | LightGBM | LightGBM attained maximum accuracy (99%) for prediction of CKD. |
| [5] | Baswaraj D et al. | 2024 | Random Forest with Recursive Feature Elimination | Enhanced performance in the prediction of CKD stages. |
| [6] | Nitasha Khan et al. | 2024 | Random Forest, SVM | Verified non-invasive CKD diagnostic models on F1-score and sensitivity. |
| [7] | Hira Khalid et al. | 2023 | Hybrid Model (Gaussian Naïve Bayes, Gradient Boosting, Decision Trees) | Attained 100% accuracy on UCI CKD data. |
| [8] | Gazi Mohammed Ifraz et al. | 2024 | Logistic Regression | Logistic regression model attained 97% accuracy in predicting CKD. |
| [9] | SusmithaMandava et al. | 2024 | Artificial Neural Networks (ANNs) | ANNs reached a maximum accuracy of 100% among the ML approaches. |
| [10] | MadhusreeSankar Roy et al. | 2021 | Extra Trees Classifier | Outperformed other models at 99.36% accuracy. |
| [11] | VirenJadhav et al. | 2023 | K-Nearest Neighbors (KNN), Support Vector Machine (SVM) | Projected to develop CKD at an early stage with KNN and SVM for early treatment in medicine. |

| Ref No. | Author Name | Year | Method | Key Findings |
|---|---|---|---|---|
| [12] | E. Tejasree et al. | 2023 | ML Framework (Preprocessing, Transformation, ML Classifiers) | Early-stage CKD prediction system provided. |
| [13] | VineetaGulati et al. | 2021 | Real-life Dataset Analysis | Real-world datasets improved the model's accuracy in predicting CKD. |
| [14] | Bhoomika CM et al. | 2024 | Bagging Ensemble (RF, XGBoost, AdaBoost) | Obtained 97% accuracy with ensemble approaches. |

Collectively, these studies indicate that ML methods have improved to predict CKD: several algorithms will be successful and indicate further potential for improvement and optimization.

## 5. System requirements

**Hardware**

- **Processor:** Intel Core i5 or better multi-core processor.
- **Memory (RAM):** Min 8 GB, but 16 GB or more is recommended to speed up big data processing and big data model training.
- **Storage:** Min 512GB in storage (SSD is recommended) for fast reads/writes on datasets and models.
- **Graphics Processing Unit (GPU):** NVIDIA and at least with CUDA support, such as NVIDIA GTX 1650 or better, to speed up machine learning model training and evaluation.

**Software**

- **Operating System:** Windows 10 for machine learning purposes
- **Development Environment:** Jupyter Notebook or Anaconda.
- **ML Libraries, Visualization, and Data Processing Tools:**

For this assignment, here is a brief overview of the machine-learning libraries and modules used:

- **pandas (pd):** General tabular data manipulation
- **numpy (np):** Numerical operations and array manipulations
- **matplotlib (plt):** Static, animated, and interactive visualizations
- **seaborn (sns):** High-level statistical graphics built on top of matplotlib
- **plotly:** Interactive and dynamic visualizations
- **warnings:** Handling or silencing warning messages
- **sklearn. ensemble:** Extra Trees, Random Forest, Gradient Boosting classifiers

- **sk-learn. preprocessing:** Label Encoder and Standard Scaler for preprocessing
- **sk-learn.impute:** Simple Imputer and Iterative Imputer for imputation
- **cv2 (OpenCV):** everything with images
- **gdown:** Downloads files from Google Drive
- **statsmodels.api:** Statistical models and statistical tests are implemented
- **variance_inflation_factor:** Feature multicollinearity measurement
- **sk-learn.model_selection:** Splits dataset into training set using train_test_split
- **sk-learn.neighbors:** k-Nearest Neighbors
- **sk-learn.metrics:** evaluation metrics like accuracy and F1-score
- **sk-learn.inspection:** Feature importance via permutation_importance
- **VotingClassifier:** Combining multiple classifiers into a single ensemble model. The packages combined are those used in preparing data, visualization, feature engineering, building, assessment, and interpretation for the project.

**Programming Languages**

- **Python:** Primary implementation language

The combined hardware, software, and programming languages would therefore be able to deliver a robust and effective research environment for running machine learning-based research in predicting Chronic Kidney Disease.

## 6. Methodology

CKD is systematically predicted with the help of machine learning. It involves preprocessing of data or basically cleaning and normalization and doing missing values handling. Feature engineering and selection are used in order to optimize the input variables for model training. Various algorithms like LR, KNN, RF, XGB, and GBC, which all are implemented and tried. The prime process of tuning does Grid Search Cross-Validation on hyperparameter tuning. The system is based on proper accuracy, precision, recall, and F1-score to validate the correctness of any model. In the end, the developed ENM gets a few best performing algorithms mixed into an improved Ensemble Model.

**Comprehensive Workflow for CKD Prediction using Machine Learning**

Figure 1 Workflow for the prediction of CKD by structured machine learning technique. First, a CKD dataset is collected, and then it undergoes exploratory data analysis and cleaning to prepare the dataset for further analysis in case there are inconsistencies or missing values.

This would be the preprocessing of data involving missing values handling, selection of the most relevant features to predict, normalization or standardization of the data, and splitting prepared data into subsets for training and testing with cross-validation for robust model evaluation.

The predictive models are trained using the following different machine learning classification techniques: KNN (k-Nearest Neighbors), DT (Decision Tree), DTC (Decision Tree Classifier), RF (Random Forest), XGB (Extreme Gradient Boosting), LR (Logistic Regression), SVM (Support Vector Machine), and GBC (Gradient Boosting Classifier). After comparing the performances of these models, the best two are selected for developing a custom Ensemble Model known as ENM based on their cumulative strengths.

Finally, the models are tested for reliability through the use of key performance metrics: accuracy, precision, recall, F1-score, and area under the curve (AUC), to ensure the reliability of disease prediction. The outcome of this workflow results in a strong prediction of being positive or negative for CKD.

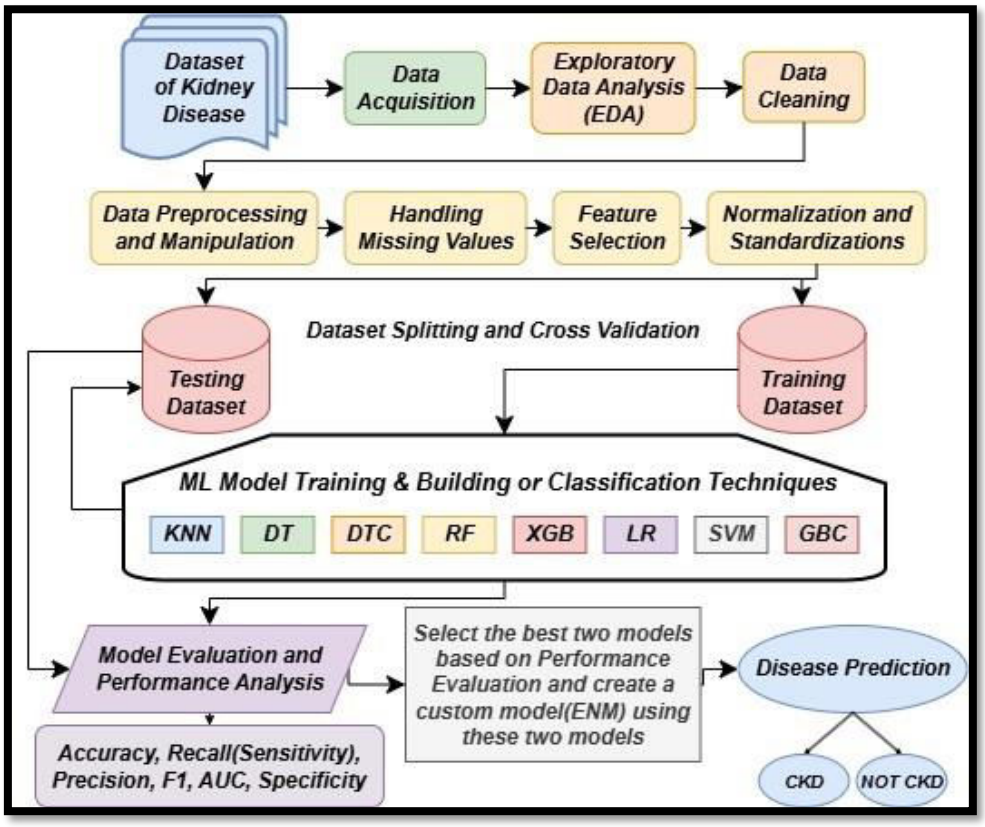



**Figure 1:** Workflow Diagram for Chronic Kidney Disease Prediction using Machine Learning

### Overview of Chronic Kidney Disease Dataset

Figure 2 is a preview of the CKD dataset loaded into a pandas DataFrame. The dataset has 400 rows and 25 columns, which are various features like age, blood_pressure (bp),

specific_gravity (sg), albumin (al), and sugar (su), along with diagnostic results like red_blood_cells (rbc), packed_cell_volume (pcv), and classification labels (ckd or notckd).

It consists of mixed feature types including both categorical and numerical features along with missing values in some columns. This has to be preprocessed as well. The classification column indicates whether the patient has CKD or not; the table also portrays diversity and complexity, which thus requires preprocessing appropriately to be predicted accurately.

| id | age | bp | sg | al | su | rbc | pc | pcc | ba | bgr | ... | pcv | wc | rc | htn | dm | cad | appet | pe | ane | classification |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 48.0 | 80.0 | 1.020 | 1.0 | 0.0 | NaN | normal | notpresent | notpresent | 121.0 | ... | 44 | 7800 | 5.2 | yes | yes | no | good | no | no | ckd |
| 1 | 7.0 | 50.0 | 1.020 | 4.0 | 0.0 | NaN | normal | notpresent | notpresent | NaN | ... | 38 | 6000 | NaN | no | no | no | good | no | no | ckd |
| 2 | 62.0 | 80.0 | 1.010 | 2.0 | 3.0 | normal | normal | notpresent | notpresent | 423.0 | ... | 31 | 7500 | NaN | no | yes | no | poor | no | yes | ckd |
| 3 | 48.0 | 70.0 | 1.005 | 4.0 | 0.0 | normal | abnormal | present | notpresent | 117.0 | ... | 32 | 6700 | 3.9 | yes | no | no | poor | yes | yes | ckd |
| 4 | 51.0 | 80.0 | 1.010 | 2.0 | 0.0 | normal | normal | notpresent | notpresent | 106.0 | ... | 35 | 7300 | 4.6 | no | no | no | good | no | no | ckd |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 395 | 55.0 | 80.0 | 1.020 | 0.0 | 0.0 | normal | normal | notpresent | notpresent | 140.0 | ... | 47 | 6700 | 4.9 | no | no | no | good | no | no | notckd |
| 396 | 42.0 | 70.0 | 1.025 | 0.0 | 0.0 | normal | normal | notpresent | notpresent | 75.0 | ... | 54 | 7800 | 6.2 | no | no | no | good | no | no | notckd |
| 397 | 12.0 | 80.0 | 1.020 | 0.0 | 0.0 | normal | normal | notpresent | notpresent | 100.0 | ... | 49 | 6600 | 5.4 | no | no | no | good | no | no | notckd |
| 398 | 17.0 | 60.0 | 1.025 | 0.0 | 0.0 | normal | normal | notpresent | notpresent | 114.0 | ... | 51 | 7200 | 5.9 | no | no | no | good | no | no | notckd |
| 399 | 58.0 | 80.0 | 1.025 | 0.0 | 0.0 | normal | normal | notpresent | notpresent | 131.0 | ... | 53 | 6800 | 6.1 | no | no | no | good | no | no | notckd |

400 rows × 25 columns

**Figure 2:** Representation of the Chronic Kidney Disease Dataset

| Feature | Description | Type | Example |
|---|---|---|---|
| Age | The age of the patient in years. | Numerical | 48 years |
| Blood Pressure (bp) | The patient's blood pressure, typically measured in mm Hg. | Numerical | 80 (diastolic pressure) |
| Specific Gravity (sg) | Measure of urine concentration, indicating kidney's ability to concentrate urine. | Categorical (1.005-1.025) | 1.020 |
| Albumin (al) | Presence of albumin in urine, indicating kidney damage. | Categorical (0-5) | 2 |
| Sugar (su) | Presence of sugar in urine, possibly indicating diabetes or kidney disease. | Categorical (0-5) | 3 |
| Red Blood Cells (rbc) | Presence of red blood cells in urine, abnormal levels may indicate kidney disease. | Categorical (normal, abnormal) | abnormal |
| Pus Cell (pc) | Presence of pus cells in urine, indicating infection or inflammation in kidneys. | Categorical (normal, abnormal) | normal |
| Pus Cell Clumps (pcc) | Clumps of pus cells in urine, suggesting a more severe infection. | Categorical (present, not present) | not present |
| Bacteria (ba) | Presence of bacteria in urine, possibly indicating a urinary tract infection. | Categorical (present, not present) | present |
| Blood Glucose Random (bgr) | Random measurement of blood glucose levels, indicating blood sugar control. | Numerical (mg/dL) | 121 mg/dL |
| Blood Urea (bu) | Level of urea in blood, an indicator of kidney function. | Numerical (mg/dL) | 36 mg/dL |
| Serum Creatinine (sc) | Level of creatinine in blood, a marker of kidney function. | Numerical (mg/dL) | 1.2 mg/dL |
| Sodium (sod) | Level of sodium in blood, affecting kidney function. | Numerical (mEq/L) | 140 mEq/L |
| Potassium (pot) | Level of potassium in blood, important for kidney function. | Numerical (mEq/L) | 4.5 mEq/L |
| Hemoglobin (hemo) | Amount of hemoglobin in blood, can be low in kidney disease. | Numerical (g/dL) | 13.5 g/dL |
| Packed Cell Volume (pcv) | Volume percentage of red blood cells in blood, related to kidney function. | Numerical (Percentage) | 41% |
| White Blood Cell Count (wc) | Number of white blood cells in blood, indicating infection or inflammation. | Numerical (cells/cumm) | 8400 cells/cumm |
| Red Blood Cell Count (rc) | Number of red blood cells in blood, related to overall blood health. | Numerical (millions/cumm) | 5.2 millions/cumm |
| Hypertension (htn) | Indicates whether the patient has hypertension, often associated with CKD. | Categorical (yes, no) | yes |
| Diabetes Mellitus (dm) | Indicates whether the patient has diabetes mellitus, which is a risk factor for CKD. | Categorical (yes, no) | yes |
| Coronary Artery Disease (cad) | Indicates whether the patient has coronary artery disease, another risk factor for CKD. | Categorical (yes, no) | no |
| Appetite | Indicates the patient's appetite, which can be affected by CKD. | Categorical (good, poor) | good |
| Pedal Edema | Indicates whether the patient has swelling in the lower extremities, a common symptom in CKD. | Categorical (yes, no) | no |
| Anemia | Indicates whether the patient has anemia, which can be a complication of CKD. | Categorical (yes, no) | no |
| Class | Indicates whether the patient has CKD or not (the target variable). | Categorical (ckd, notckd) | ckd |

**Figure 3:** Understanding about the Kidney Disease Dataset Features

| | age | bp | sg | al | su | bgr | bu | sc | sod | pot | hemo |
|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 391.000000 | 388.000000 | 353.000000 | 354.000000 | 351.000000 | 356.000000 | 381.000000 | 383.000000 | 313.000000 | 312.000000 | 348.000000 |
| mean | 51.483376 | 76.469072 | 1.017408 | 1.016949 | 0.450142 | 148.036517 | 57.425722 | 3.072454 | 137.528754 | 4.627244 | 12.526437 |
| std | 17.169714 | 13.683637 | 0.005717 | 1.352679 | 1.099191 | 79.281714 | 50.503006 | 5.741126 | 10.408752 | 3.193904 | 2.912587 |
| min | 2.000000 | 50.000000 | 1.005000 | 0.000000 | 0.000000 | 22.000000 | 1.500000 | 0.400000 | 4.500000 | 2.500000 | 3.100000 |
| 25% | 42.000000 | 70.000000 | 1.010000 | 0.000000 | 0.000000 | 99.000000 | 27.000000 | 0.900000 | 135.000000 | 3.800000 | 10.300000 |
| 50% | 55.000000 | 80.000000 | 1.020000 | 0.000000 | 0.000000 | 121.000000 | 42.000000 | 1.300000 | 138.000000 | 4.400000 | 12.650000 |
| 75% | 64.500000 | 80.000000 | 1.020000 | 2.000000 | 0.000000 | 163.000000 | 66.000000 | 2.800000 | 142.000000 | 4.900000 | 15.000000 |
| max | 90.000000 | 180.000000 | 1.025000 | 5.000000 | 5.000000 | 490.000000 | 391.000000 | 76.000000 | 163.000000 | 47.000000 | 17.800000 |

**Figure 4:** Representation of the Dataset Description and Key Observations

**Feature Distribution Analysis and Class Comparison**

In Figure 5 comparison of the distribution of several numerical features between the two classes: CKD and Non-CKD. Density plots help to show what the features look like and whether each represents significant differences in the two groups under study. For example:

- Distributions of blood urea, serum creatinine, and packed cell volume features are very well defined between the two classes. Then, they are good discriminators.
- The two distributions share similarities in some features, for instance, sodium and potassium, thus contributing minimally towards differentiating the classes.

These inform feature selection and modeling decisions.

This graph illustrates the difference in numeric features between CKD and non-CKD patients in terms of density plots. The prominent features, such as serum creatinine, blood urea, and packed cell volume have clear-cut differences and are therefore imperative for predicting CKD. Features such as sodium and potassium contain overlapping distributions with reduced predictive power. It is helpful to find these differences in feature choice and improve models' accuracy and dependability. The graph also finds outliers and skewness and influences data preprocessing as well as the performance of models.

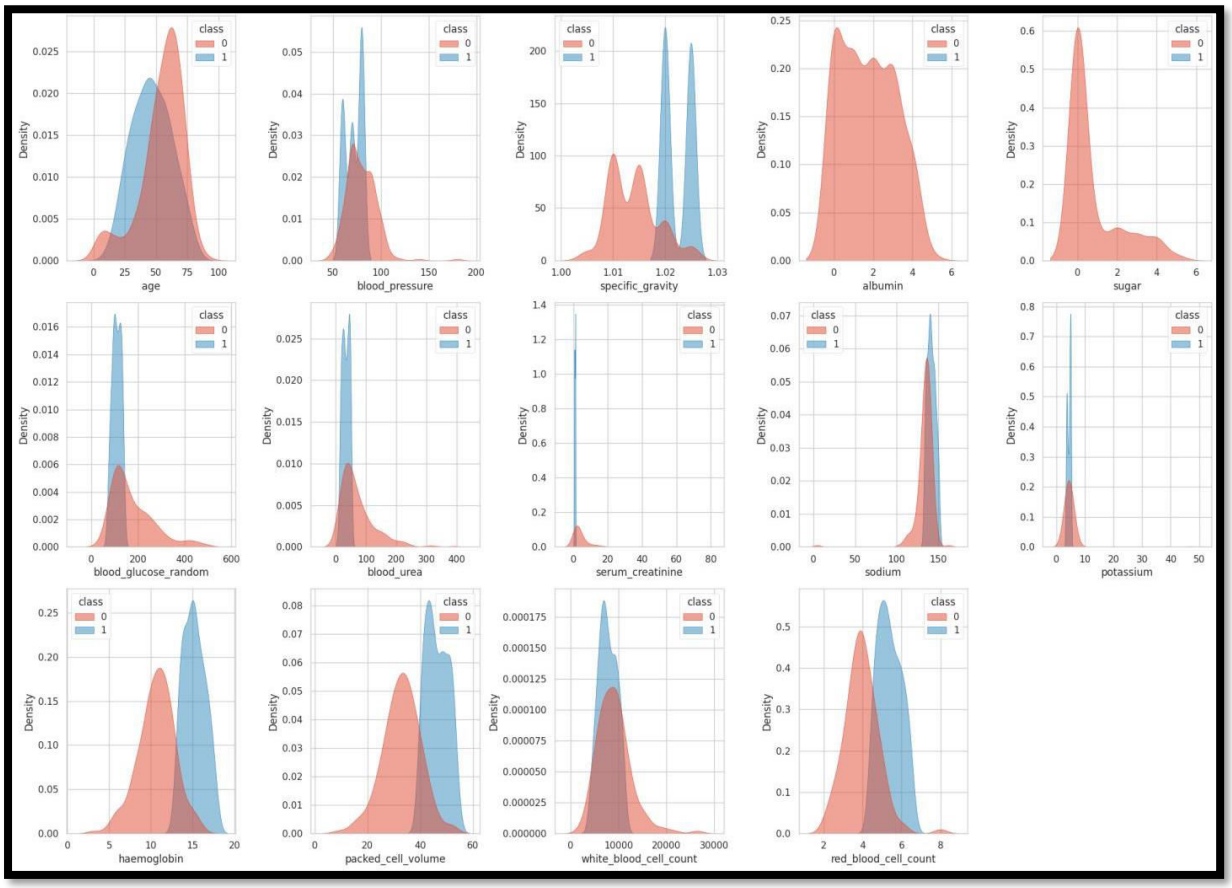



**Figure 5:** Distribution of Numeric Features by Disease Class

**Categorical Feature Analysis and Class Insights**

Figure 6 represents the frequency distribution of most of the categorical variables in CKD and Non-CKD patients. Some interesting observations are as follows:

- RBC and pus cells: The percentage of abnormalities is higher in CKD patients compared to the control group.
- Hypertension, diabetes_mellitus, and anemia: These are more common in CKD patients, and thus would be potentially relevant in the overall picture of disease prediction.
- Appetite: Poor appetite is strongly associated with CKD patients compared to Non-CKD individuals.

This visualization represents how category features help to distinguish one class from another.

This bar graph contrasts categorical variables such as red blood cell count (RBC), pus cells, hypertension, diabetes mellitus, and anemia in CKD and non-CKD patients. It shows that RBC abnormality, hypertension, and loss of appetite are comparatively more common in CKD patients, which reinforces their relative significance as diagnoses. The presence of pus cells and the diabetes status between the two groups are also quite different and therefore effective predictors. These results are of utmost

importance to incorporate into feature encoding and improve classification models for early CKD detection.

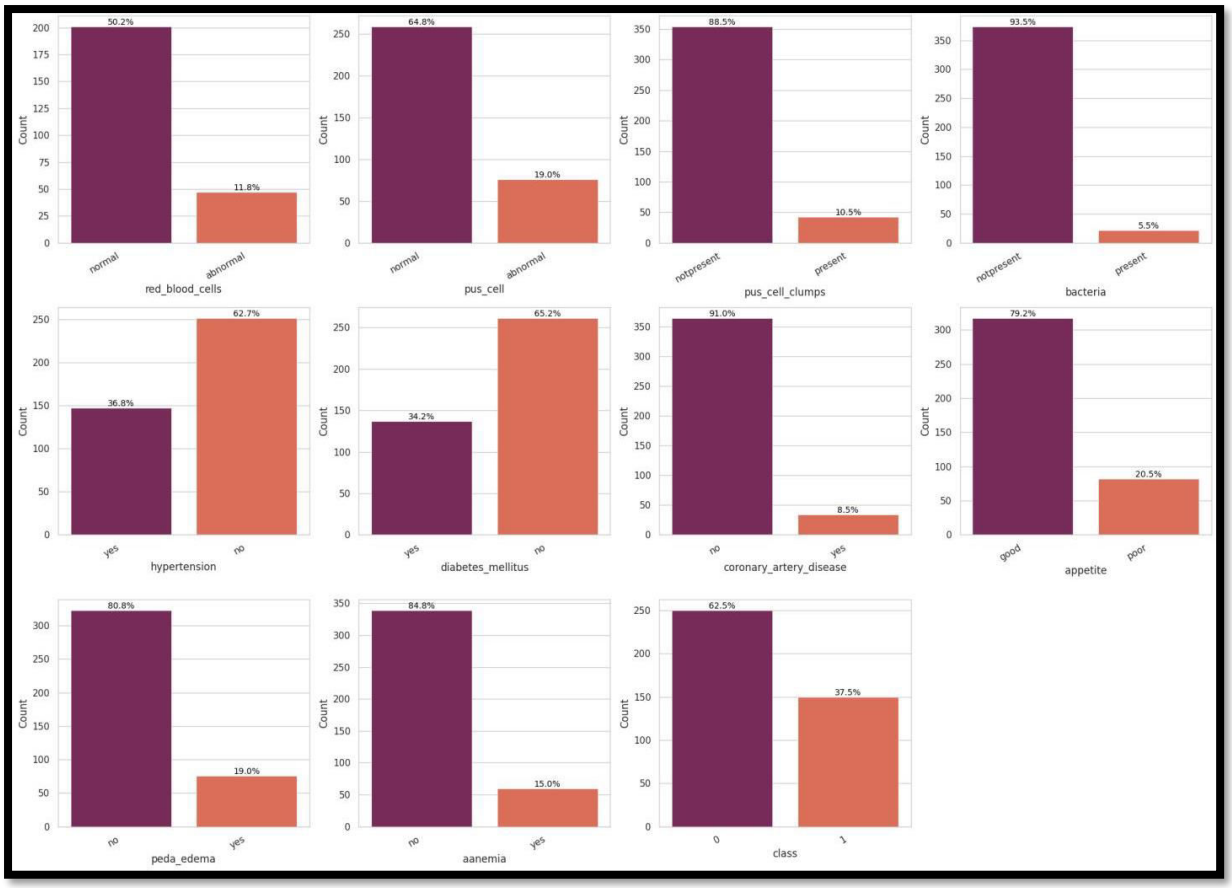



**Figure 6:** Distribution of Categorical Features by Class

**Outcome Variable Analysis and Distribution**

Figure 7 the class distribution is presented with both a bar chart (left) and a pie chart (right).

- Left Bar Chart: It shows the number of samples in each class. The total dataset consists of 250 CKD cases and 150 non-CKD cases, so it is biased toward CKD cases.
- Right Pie Chart: The proportion of CKD and non-CKD cases is depicted, where CKD accounts for 62.5% of the dataset, and non-CKD accounts for 37.5%.

The image clearly shows a class imbalance, which may skew model training, thus requiring some resampling strategy, cost-sensitive learning, or AUC-ROC and F1-score in the performance measures.

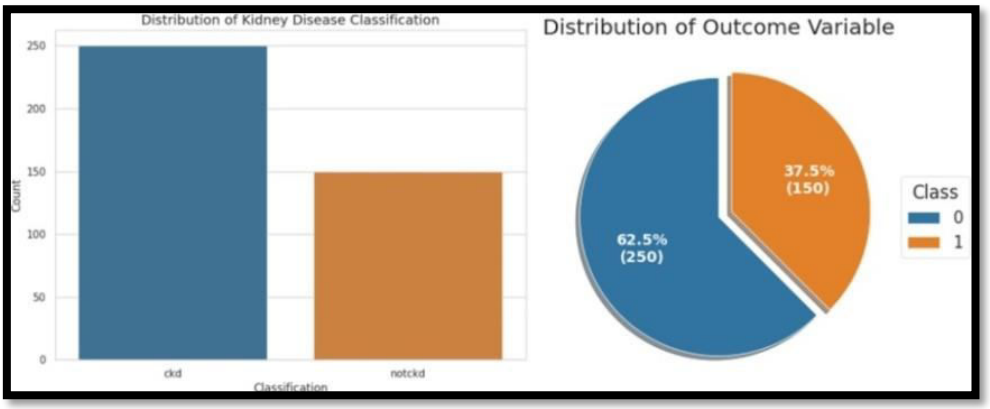



**Figure 7:** Distribution of Outcome Variable for Chronic Kidney Disease Classification

**Feature Selection and Significance Analysis**

Figure 8 Relative importance's of the features for predicting Chronic Kidney Disease from the model.

- **Major Features**
  - Red Blood Cells: Score is 0.161.
  - Specific Gravity: The score is also 0.139.
  - Albumin, Packed Cell Volume and Hypertension all rank in a row as major predictors.
- **Minor Features**
  - Contributions of Variables such as Bacteria, Pus Cell Clumps, and Coronary Artery Disease towards the predictability of the model are negligible.

This makes the feature selection process efficient in terms of focusing on high-impact variables that may improve the model's performance at the cost of computational overhead. Visualization underlines the role of clinical biomarkers in the classification of CKD.

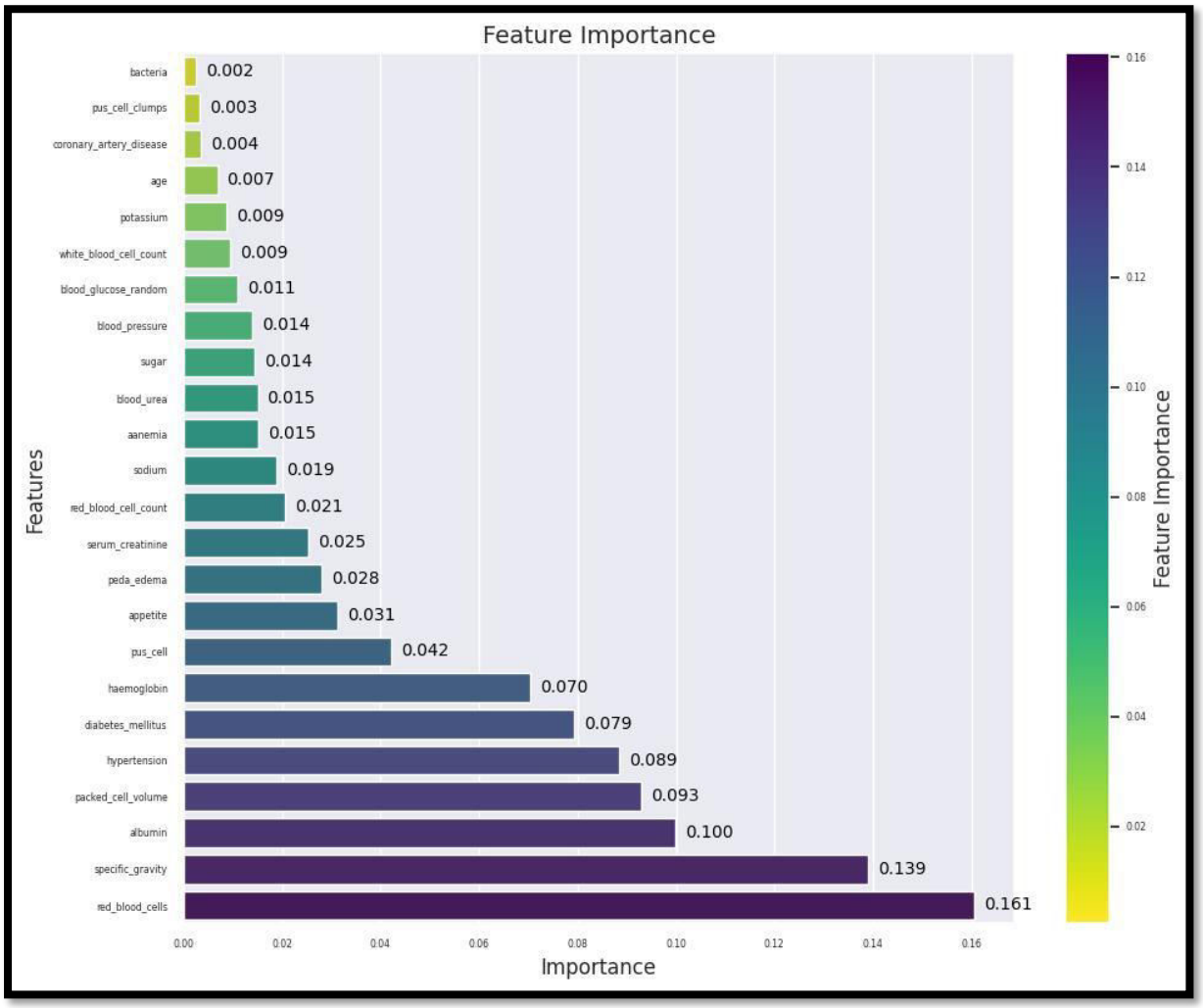



**Figure 8:** Feature Importance in Predicting Chronic Kidney Disease

**Correlation Analysis for Feature Interdependencies**

Figure 9 Correlation Matrix - Heatmap for Numeric Features This is the heat map for the correlation matrix for the numeric features of the dataset. Color intensity reflects the strength of the relationship between feature pairs:

- **High Positive Correlations**
  - Packed Cell Volume and Haemoglobin, 0.9 very highly correlated.
  - Specific Gravity and Albumin, 0.73 very highly correlated.
- **Negative Correlations**
  - Specific Gravity and Blood Urea, -0.47 inversely related.
  - Haemoglobin and Serum Creatinine, -0.63 very negative.
- **Outcome Variable (Class)**
  - Specific Gravity shows a very high positive correlation with the target variable with a coefficient of correlation of 0.73 and Albumin 0.77.
  - Haemoglobin and Packed Cell Volume are also highly correlated with outcome.

The interdependencies among the features highlighted through such an analysis will guide feature engineering and selection strategies for predictive modeling.
This plots a correlation heatmap of the numerical feature correlations of the dataset features. Albumin and specific gravity, and hemoglobin and packed cell volume, both exhibit strongly positive correlations which indicate their relation to each other in diagnosing CKD. Hemoglobin has a negative correlation with serum creatinine, validating their use as markers of the clinical assessment of kidney function.
The heatmap helps in the identification of redundant or highly correlated features, which can be removed or redefined to improve model efficiency. The low correlation between some features also points towards their poor predictive ability, so feature selection and dimensionality reduction towards improved model performance is simpler.

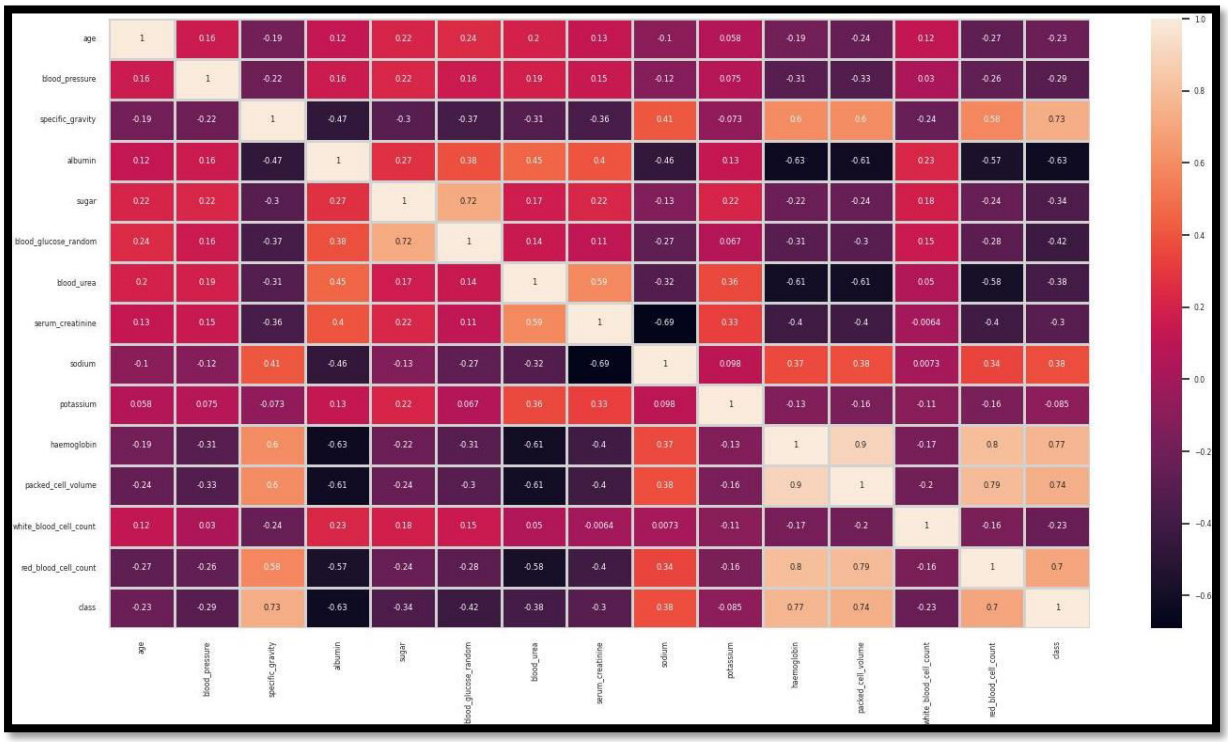



**Figure 9:** Heatmap of Correlations among Numeric Features

**Categorical Features and Their Impact on Outcome**

Figure 10 plots dot plots of the association between other categorical features and the outcome classes, Class 0 and Class 1. Each subplot corresponds to a specific categorical feature, plotted against the age of the individuals for both class groups. The red and blue dots represent Class 1 (disease present) and Class 0 (disease absent), respectively.
Important observations:

- Hypertension: Most of the patients with hypertension fall in Class 1.
- Diabetes Mellitus: Diabetes mellitus is strongly related to Class 1.

- Peda Edema and Anemia: The class is more likely to be of Class 1 in the case of conditions.
- Coronary Artery Disease: This is slightly higher in Class 1 as compared to Class 0.
- Presence of Bacteria: No class-dependent trend is noted for this characteristic.

This visualization efficiently helps in describing the contribution of categorical variables made toward distinguishing one class from the other, giving very good insights into distribution.

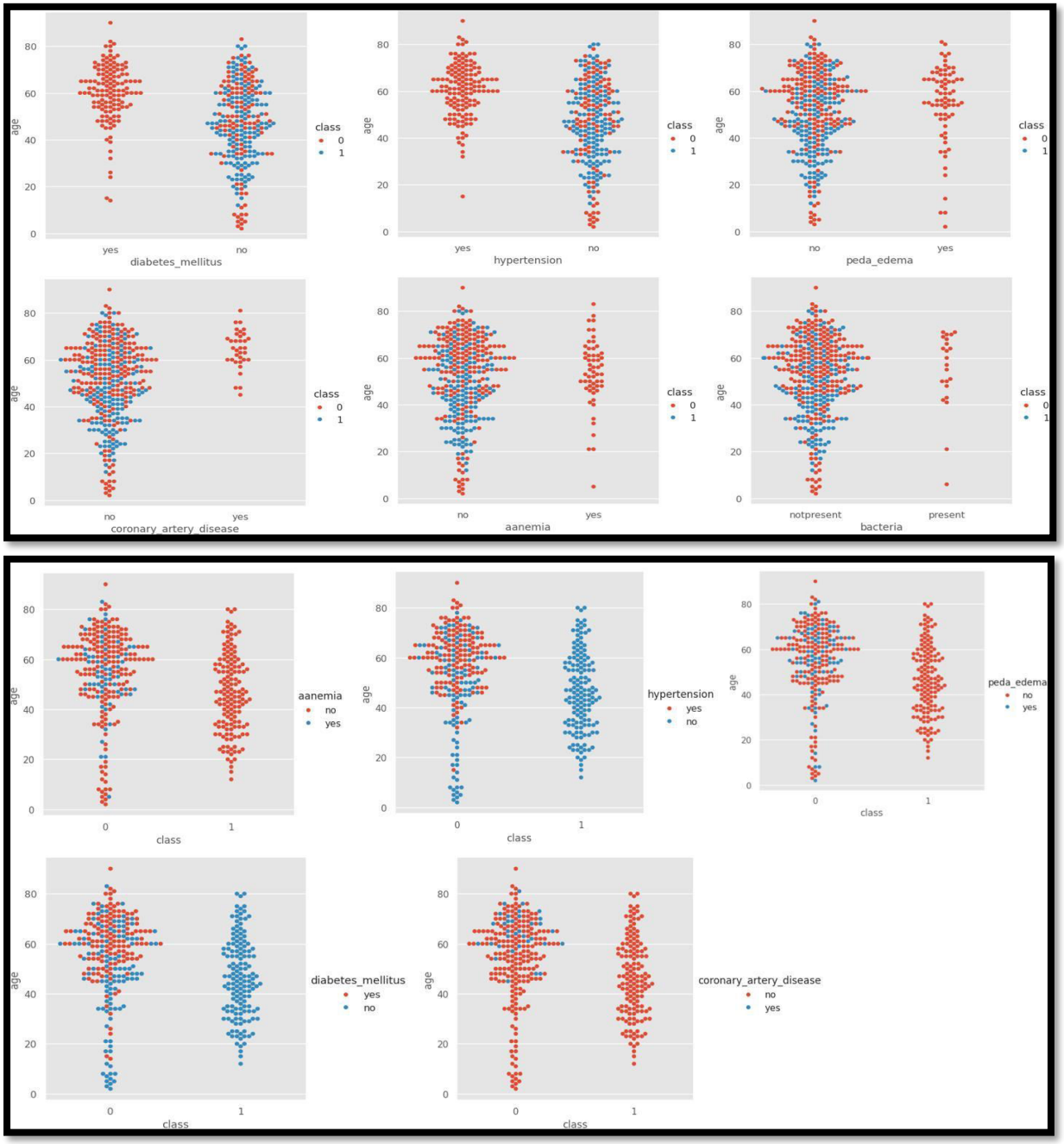
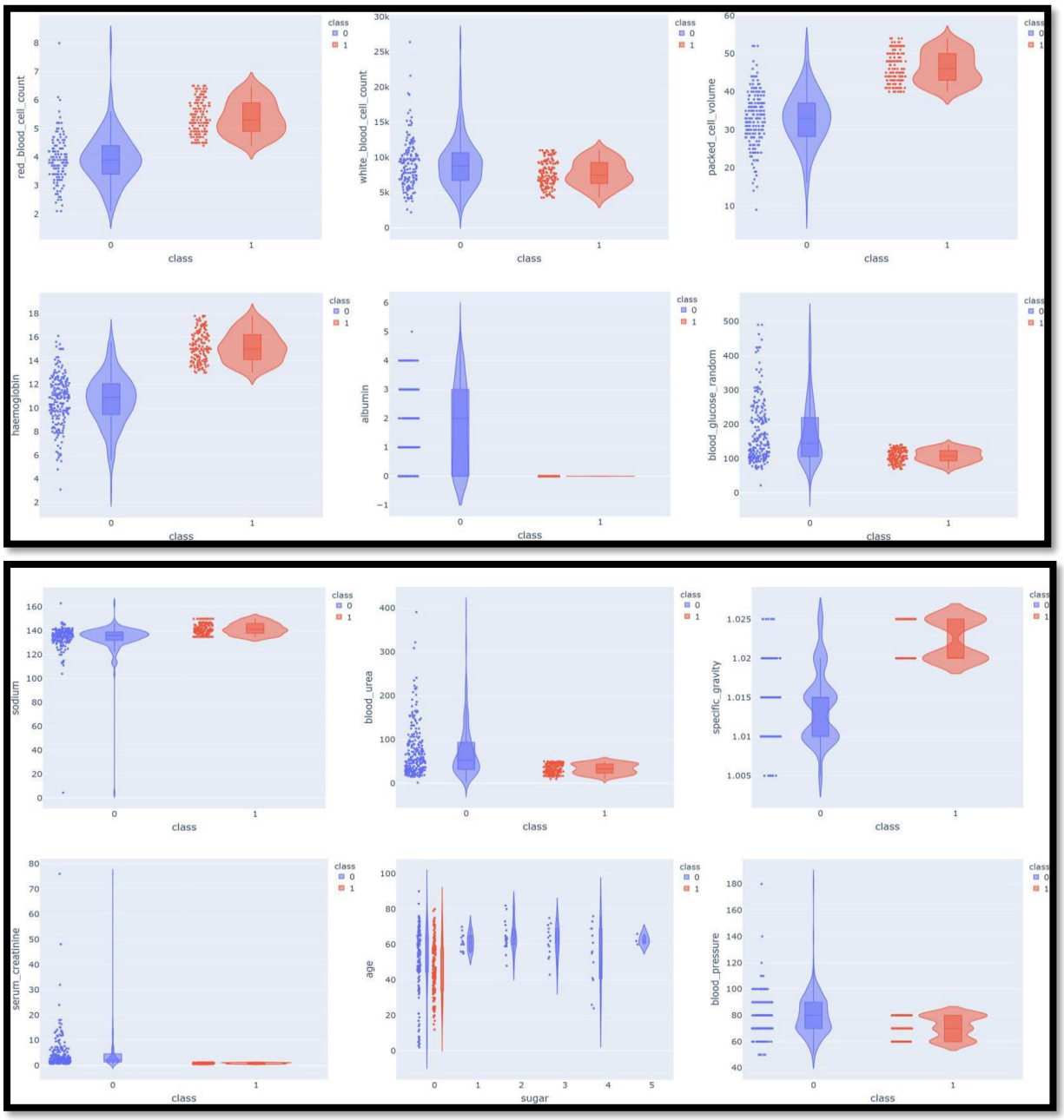



**Figure 10:** Dot Plot Representation of Class Distribution by Categorical Features

**Comprehensive Distribution Analysis of Numerical Variables**

Figure 11 Employ violin plots combined with box plots as a double-layer plot to visualize numerical feature distributions on both classes of images, specifically, Class 0 and Class 1. In the violin plot, there's information regarding the shape and density. Added above would be a plot overlay including that particular statistical summary on medians, quartiles, and the like of outliers taken from the box plot.

This figure provides a comprehensive view of the behavior of numerical features showing the important differences between classes and the detailed statistical context.



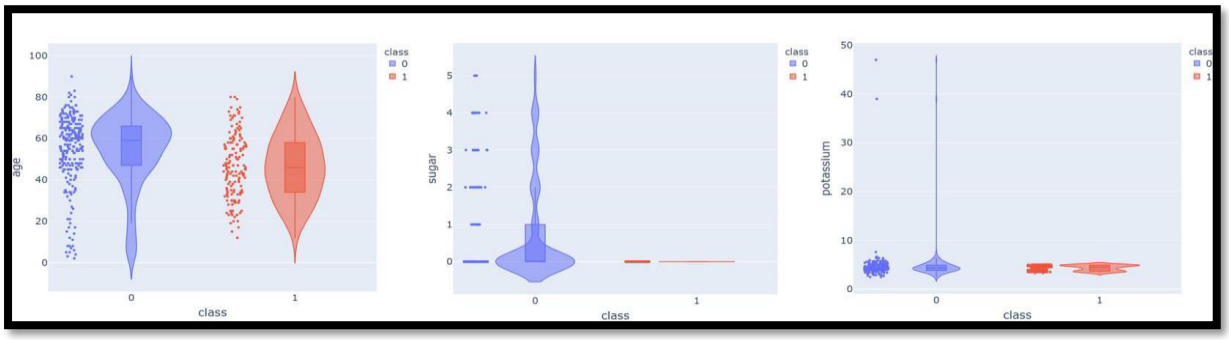
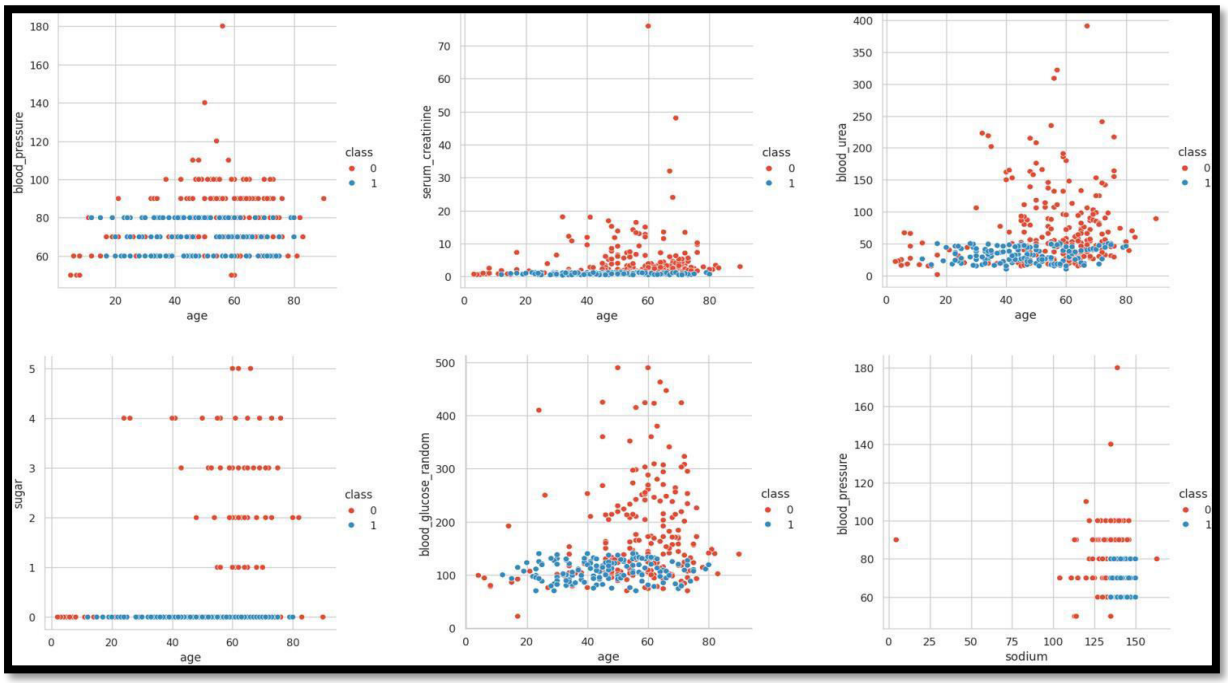


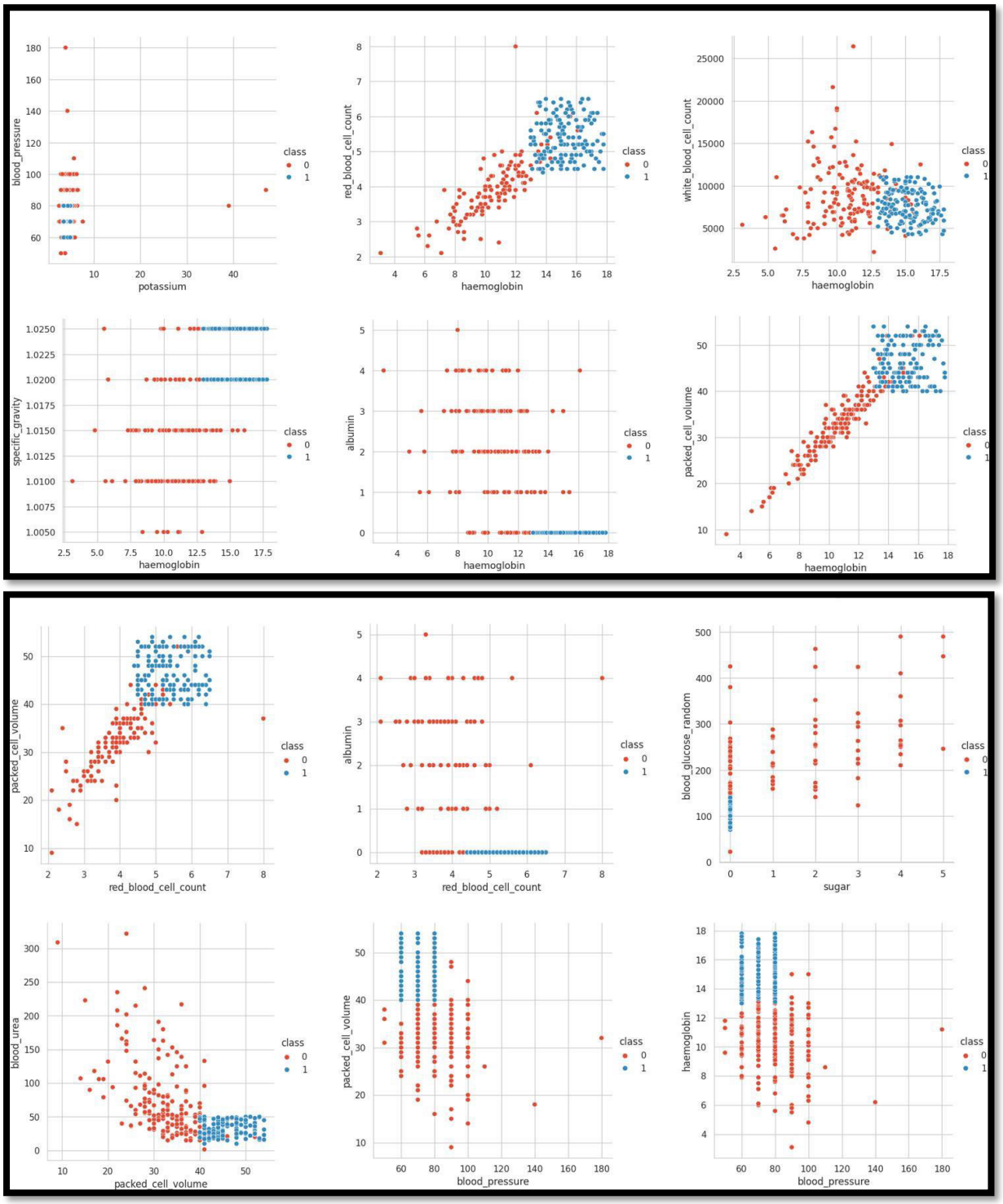

**Figure 11:** Violin and Box Plot Comparison of Numerical Features across Classes

**Violin and Scatter Plot Analysis Highlighting Distribution and Correlations Across Different Classes**

Figure 12 is a complete depiction of hematological and biochemical parameters by various classes. The graphs are violin and scatter graphs for the various features, i.e., red blood cell count, white blood cell count, packed_cell_volume, hemoglobin, albumin, blood_glucose_random, blood_pressure, specific gravity, serum_creatinine, sodium, potassium, sugar level, and age.

Violin plots illustrate the distribution, density, and spread of each parameter across each class, with relationships between other parameters highlighted by the scatter plots. The colors differentiate between classes (0 and 1) and allow for easy comparison. The plots illustrate distinctive patterns and correlations that could be helpful in class-specific feature detection, facilitating data-driven analysis and interpretation.




blood_pressure
potassium
red_blood_cell_count
haemoglobin
white_blood_cell_count
haemoglobin
specific_gravity
haemoglobin
albumin
haemoglobin
packed_cell_volume
haemoglobin
packed_cell_volume
red_blood_cell_count
albumin
red_blood_cell_count
blood_glucose_random
sugar
blood_urea
packed_cell_volume
packed_cell_volume
blood_pressure
haemoglobin
blood_pressure
class
0
1

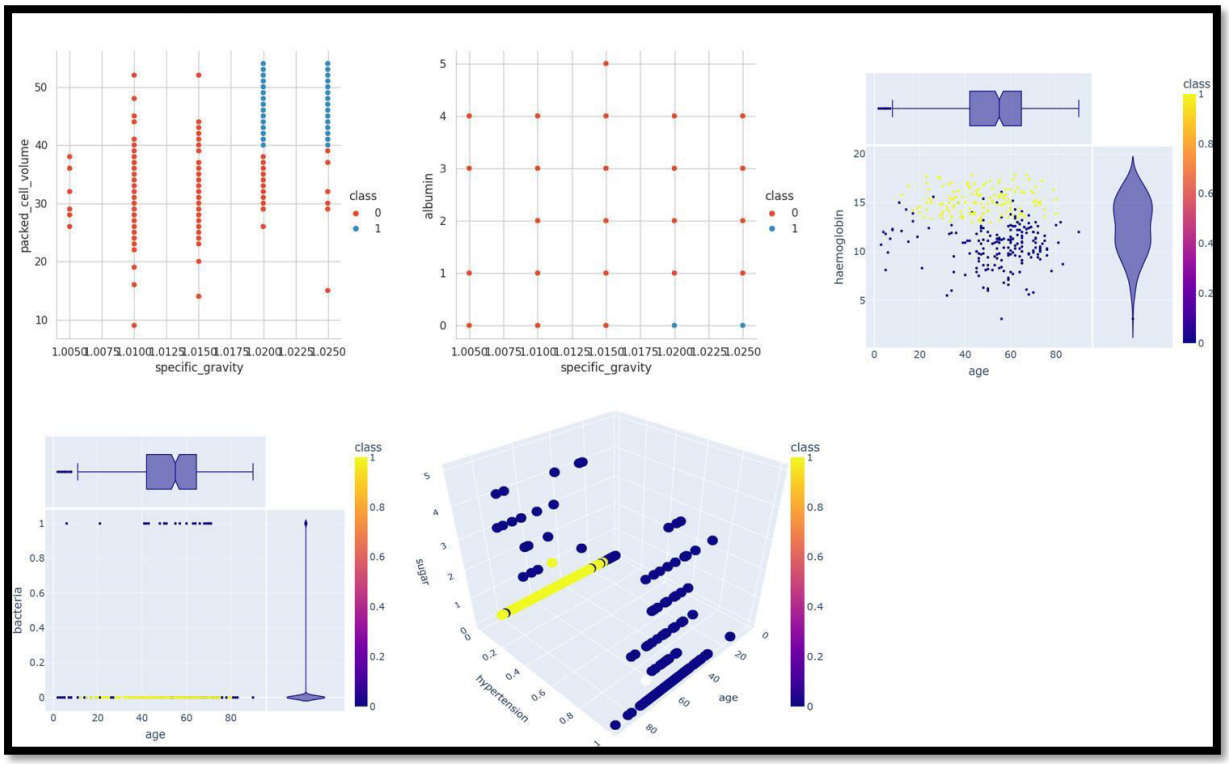



**Figure 12:** Comprehensive Visualization of Hematological and Biochemical Parameters by Class

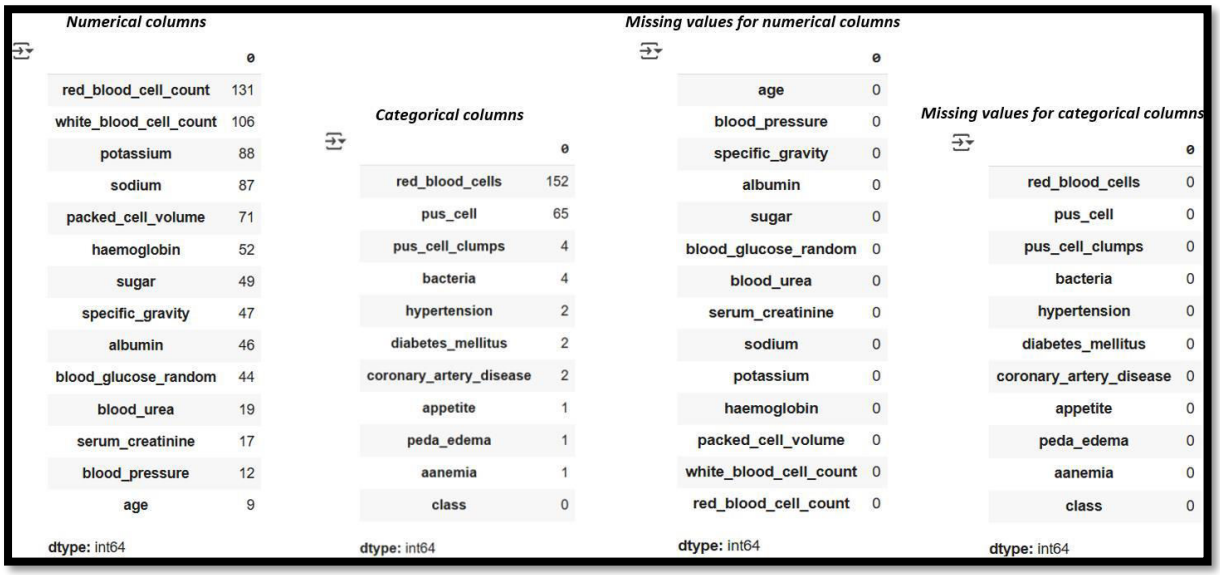
**Missing Value Imputation**

Figure 13 displays a thorough categorization of missing data over both numerical and categorical columns in the data set, which displays the meticulous precautions exercised in preprocessing to handle missing values efficiently. The chart consists of two overarching categories:

The left side of the figure shows the original number of missing values for all columns, highlighting key numeric columns such as "red blood cell count" (131 missing) and "potassium" (88 missing), and categorical columns such as "red blood cells" (152 missing), with a keen emphasis on data gaps before imputation.

The right side of the figure displays results after applying imputation techniques, where the absence of missing values in all columns shows successful data processing. Numerical and categorical columns are now complete, as is evident from columns with "o" missing entries.

This picture depicts the importance of imputation of missing values in machine learning processes. Proper imputation of missing values during preprocessing ensures data quality and is the foundation of reliable model building and analysis.



*Numerical columns*

| | 0 |
|---|---|
| red_blood_cell_count | 131 |
| white_blood_cell_count | 106 |
| potassium | 88 |
| sodium | 87 |
| packed_cell_volume | 71 |
| haemoglobin | 52 |
| sugar | 49 |
| specific_gravity | 47 |
| albumin | 46 |
| blood_glucose_random | 44 |
| blood_urea | 19 |
| serum_creatinine | 17 |
| blood_pressure | 12 |
| age | 9 |

dtype: int64

*Categorical columns*

| | 0 |
|---|---|
| red_blood_cells | 152 |
| pus_cell | 65 |
| pus_cell_clumps | 4 |
| bacteria | 4 |
| hypertension | 2 |
| diabetes_mellitus | 2 |
| coronary_artery_disease | 2 |
| appetite | 1 |
| peda_edema | 1 |
| aanemia | 1 |
| class | 0 |

dtype: int64

*Missing values for numerical columns*

| | 0 |
|---|---|
| age | 0 |
| blood_pressure | 0 |
| specific_gravity | 0 |
| albumin | 0 |
| sugar | 0 |
| blood_glucose_random | 0 |
| blood_urea | 0 |
| serum_creatinine | 0 |
| sodium | 0 |
| potassium | 0 |
| haemoglobin | 0 |
| packed_cell_volume | 0 |
| white_blood_cell_count | 0 |
| red_blood_cell_count | 0 |

dtype: int64

*Missing values for categorical columns*

| | 0 |
|---|---|
| red_blood_cells | 0 |
| pus_cell | 0 |
| pus_cell_clumps | 0 |
| bacteria | 0 |
| hypertension | 0 |
| diabetes_mellitus | 0 |
| coronary_artery_disease | 0 |
| appetite | 0 |
| peda_edema | 0 |
| aanemia | 0 |
| class | 0 |

dtype: int64

**Figure 13:** Overview of Missing Data and Completion Status

## Feature Selection Using VIF

In this figure 14, VIF values are shown to establish multicollinearity between features. It can be seen that most variables have low VIF values, which signifies minimum correlation and, hence, removes redundancy from the dataset to be modeled.

*Variance Inflation Factor (VIF)*

| | Column | VIF |
|---|---|---|
| 14 | const | 47134.704658 |
| 11 | packed_cell_volume | 2.646588 |
| 10 | haemoglobin | 2.642671 |
| 6 | blood_urea | 2.117688 |
| 7 | serum_creatinine | 2.113107 |
| 8 | sodium | 1.676145 |
| 4 | sugar | 1.552265 |
| 5 | blood_glucose_random | 1.536479 |
| 2 | specific_gravity | 1.525247 |
| 13 | red_blood_cell_count | 1.523607 |
| 3 | albumin | 1.462135 |
| 9 | potassium | 1.225934 |
| 1 | blood_pressure | 1.136647 |
| 0 | age | 1.111428 |
| 12 | white_blood_cell_count | 1.071832 |

**Figure 14:** Variance Inflation Factor (VIF) Analysis

## Data Cleaning and Pre-Processing

The following figure 15 compares the dataset before and after pre-processing. In the "Before Pre-Processing" stage, there were many missing values in columns and non-uniform data types. All missing values were handled and data types were standardized after carrying out the above-mentioned pre-processing steps, which eventually gave a clean and complete dataset for further analysis.

*Before pre-processing*

```
<class 'pandas.core.frame.DataFrame'>
Index: 400 entries, 0 to 399
Data columns (total 25 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   age                     391 non-null    float64
 1   blood_pressure          388 non-null    float64
 2   specific_gravity        353 non-null    float64
 3   albumin                 354 non-null    float64
 4   sugar                   351 non-null    float64
 5   red_blood_cells         248 non-null    object
 6   pus_cell                335 non-null    object
 7   pus_cell_clumps         396 non-null    object
 8   bacteria                396 non-null    object
 9   blood_glucose_random    356 non-null    float64
 10  blood_urea              381 non-null    float64
 11  serum_creatinine        383 non-null    float64
 12  sodium                  313 non-null    float64
 13  potassium               312 non-null    float64
 14  haemoglobin             348 non-null    float64
 15  packed_cell_volume      330 non-null    object
 16  white_blood_cell_count  295 non-null    object
 17  red_blood_cell_count    270 non-null    object
 18  hypertension            398 non-null    object
 19  diabetes_mellitus       398 non-null    object
 20  coronary_artery_disease 398 non-null    object
 21  appetite                399 non-null    object
 22  peda_edema              399 non-null    object
 23  aanemia                 399 non-null    object
 24  class                   400 non-null    object
dtypes: float64(11), object(14)
memory usage: 81.2+ KB
```

*After pre-processing*

```
<class 'pandas.core.frame.DataFrame'>
Index: 400 entries, 0 to 399
Data columns (total 25 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   age                     400 non-null    float64
 1   blood_pressure          400 non-null    float64
 2   specific_gravity        400 non-null    float64
 3   albumin                 400 non-null    float64
 4   sugar                   400 non-null    float64
 5   red_blood_cells         400 non-null    int64
 6   pus_cell                400 non-null    int64
 7   pus_cell_clumps         400 non-null    int64
 8   bacteria                400 non-null    int64
 9   blood_glucose_random    400 non-null    float64
 10  blood_urea              400 non-null    float64
 11  serum_creatinine        400 non-null    float64
 12  sodium                  400 non-null    float64
 13  potassium               400 non-null    float64
 14  haemoglobin             400 non-null    float64
 15  packed_cell_volume      400 non-null    float64
 16  white_blood_cell_count  400 non-null    float64
 17  red_blood_cell_count    400 non-null    float64
 18  hypertension            400 non-null    int64
 19  diabetes_mellitus       400 non-null    int64
 20  coronary_artery_disease 400 non-null    int64
 21  appetite                400 non-null    int64
 22  peda_edema              400 non-null    int64
 23  aanemia                 400 non-null    int64
 24  class                   400 non-null    int64
dtypes: float64(14), int64(11)
memory usage: 97.4 KB
```

**Figure 15:** Impact of Pre-Processing on Dataset Quality

## 7. Model building and analysis

This paper reviews the application and performance of several machine learning algorithms in the classification of patients into the presence of Chronic Kidney Disease (CKD). Comparisons of several algorithms including LR, KNN, DT, RF, GBC, and XGB are considered. A novel ensemble model known as ENM is also developed that combines models to improve its performance.

Accuracy, precision, recall, F1-score, and AUC are metrics used in measuring performance. Using the cross-validation technique and the hyper parameter tuning will ensure proper model optimization. Visualization is done using ROC curves, confusion matrices, and feature importance charts that are useful for improving the prediction of CKD.

**k-Nearest Neighbors (KNN):**A non-parametric instance-based learning algorithm that assigns a label to an instance according to the majority class of the K nearest neighbors in feature space.

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2}$$

**Decision Tree (DT)**: A decision tree is a non-parametric model which partitions the data into subsets depending upon the values of input features. It builds a tree-like model where every node is a feature, every edge is a decision rule, and every leaf is a class label.

- Gini Index:

$$Gini = 1 - \sum_{i=1}^{C} p_i^2$$

- Entropy:

$$Entropy = -\sum_{i=1}^{C} p_i \log(p_i)$$

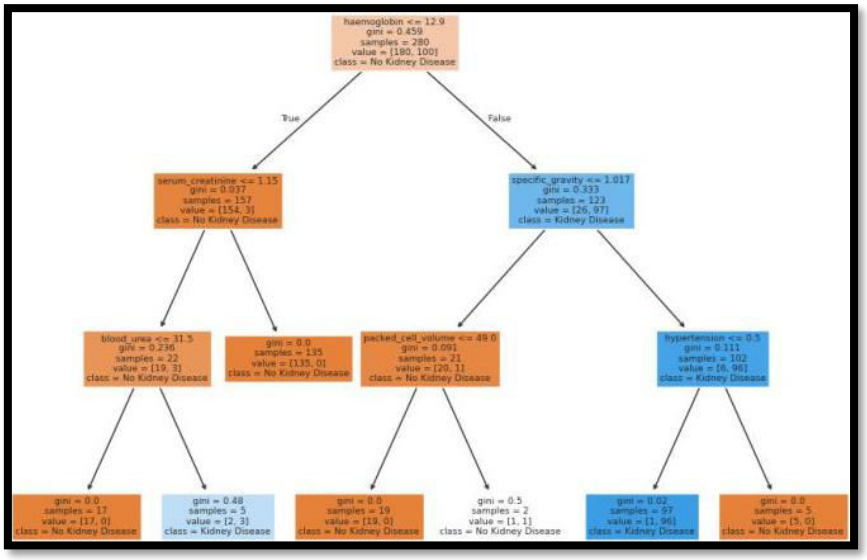



**Figure 16:** Decision making in binary class of CKD

**Grid Search Cross-Validation (DTC):**Grid Search CV is another technique utilized to hyper parameter tune a machine learning model systematically by attempting them all at once in a given parameter grid. It utilizes cross-validation to estimate the performance using each set of hyper parameters and selects the best.

$$Error = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, \hat{y}_i\right)$$

Where:

- $N$is the number of cross-validation folds.
- $L(y_i, \hat{y}_i)$is the loss function for true$y_i$and predicted$\hat{y}_i$.

**Random Forest (RF):** RF is an ensemble learning algorithm with a mixture of several decision trees. Each is trained on a random subset of data and features; the prediction, by averaging (regression), or majority voting (classification), is made.

$$f(x) = \frac{1}{T}\sum_{i=1}^{T} h_i\,(x)$$

Where:

- $T$ is the number of trees.
- $h_i(x)$ is the prediction of the $i$-th tree for the input $x$.

**XGBoost (XGB):** XGBoost is a highly optimized version of gradient boosting for decision trees. It is a method that trains models sequentially where each model tries to correct the errors of the model before it.

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

Where:

- $F_{m-1}(x)$ is the prediction at iteration $m-1$.
- $h_m(x)$ is the new weak learner (decision tree).
- $\gamma_m$ is the learning rate.

**Logistic Regression (LR):** Logistic regression is a linear model used for binary classification. It finds the probability of an input presented to it being a particular class by using the logistic function.

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-(w^T x + b)}}$$

**Support Vector Machine (SVM):** SVM is the supervised learning model that finds out the optimal hyperplane which distinguishes different classes in a higher dimensional space to the maximum level.

$$f(x) = w^T x + b$$

Where:

- $w$ is the weight vector.
- $x$ is the input vector.
- $b$ is the bias term.

The optimization maximizes the margin $M$ by minimizing: $\| w \|^2$

**Gradient Boosting Classifier (GBC):** Gradient boosting is an ensemble learning approach where a series of models are built sequentially to improve the prediction of earlier models' errors using the gradient of the loss function.

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

**Ensemble Model (ENM):** The **Ensemble Model (ENM)** is a hybrid machine-learning algorithm designed to enhance predictive accuracy and robustness by leveraging the strengths of multiple classifiers. The ENM used in this study combines the **Random Forest Classifier (RF)** and the **Gradient Boosting Classifier (GBC)** to benefit from both ensemble approaches.

**Random Forest Classifier (RF)** predicts using an ensemble of decision trees that reduce variance and increase generalization and handles high-dimensional data efficiently with minimal overfitting. The **Gradient Boosting Classifier (GBC)** builds decision trees in sequence, with each new tree trying to correct the errors made by the previous tree to detect complex patterns and improve accuracy.

The **Ensemble Model (ENM)** combines the outcomes of RF and GBC using the **majority voting** scheme. The models are trained using the same data while being trained and make the most occurring prediction at the time of prediction. The model gives greater priority to the RF prediction when predictions are different from each other.

When both models provide probability estimates, the ENM calculates the mean **average probability of the positive class** to obtain **Area under the Curve (AUC).** The approach ensures accurate and sound classification, particularly in binary cases, resulting in a robust predictive model.

**1. Ensemble Prediction (Majority Voting)**

The final prediction of the ENM ($P_{ENM}$) is determined as follows:

$$P_{ENM}(x) = \begin{cases} P_{RF}(x) & \text{if } P_{RF}(x) = P_{GBC}(x) \\ P_{RF}(x) & \text{if } P_{RF}(x) \neq P_{GBC}(x) \end{cases}$$

Where:

- $P_{RF}(x)$: Prediction from the **Random Forest Classifier (RF)** for input $x$.
- $P_{GBC}(x)$: Prediction from the **Gradient Boosting Classifier (GBC)** for input $x$.
- The model prioritizes **RF** in case of disagreement due to its robustness.

**2. Average Probability (If Available)**

If both classifiers provide probability estimates, the **average probability** is calculated as:

$$\text{Avg\_Proba}(x) = \frac{P_{RF\_proba}(x) + P_{GBC\_proba}(x)}{2}$$

Where:

- $P_{RF\_proba}(x)$: Probability of the positive class from **RF**.
- $P_{GBC\_proba}(x)$: Probability of the positive class from **GBC**.

**Training:** The training of a learning machine model to pattern recognition ability from labeled data input, and it adjusts parameters so as to minimize its errors.
**Testing:** It is called the phase of evaluation in which the applied trained model undergoes testing over unseen data that tests how well a model has generalized.
**True positive (TP):** TP is the state where actual as well as forecasted values both are positive.
**True negative (TN):** TN is the situation where both the true value of the data point and the prediction are negative.
**False positive (FP):** FP stands for the cases in which the actual value of the data point is negative while the predicted is positive.
**False negative (FN):** FN is experienced when the actual value of the data point is positive whereas the predicted one is negative.

**Recall / Sensitivity / True Positive Rate (TPR):** Recall, also known as Sensitivity or True Positive Rate, is used to measure the percentage of true positive instances that are correctly predicted as positive by the model.

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

**False Positive Rate (FPR):** False Positive Rate is the rate of true negatives that are predicted wrong by the model as positive.

$$FPR = \frac{FP}{FP + TN}$$

**Specificity:** Specificity measures the proportion of actual negatives that the model correctly identifies as negative. It is the complement of False Positive Rate: Specificity=1–FPR.

$$Specificity = \frac{TN}{TN + FP}$$

**Accuracy:** Accuracy quantifies the general precision of the model by determining the percentage of correct predictions (both true positives and true negatives) to total predictions.

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

**Precision:** Precision, also known as Positive Predictive Value, measures the proportion of predicted positive cases that are positive.

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

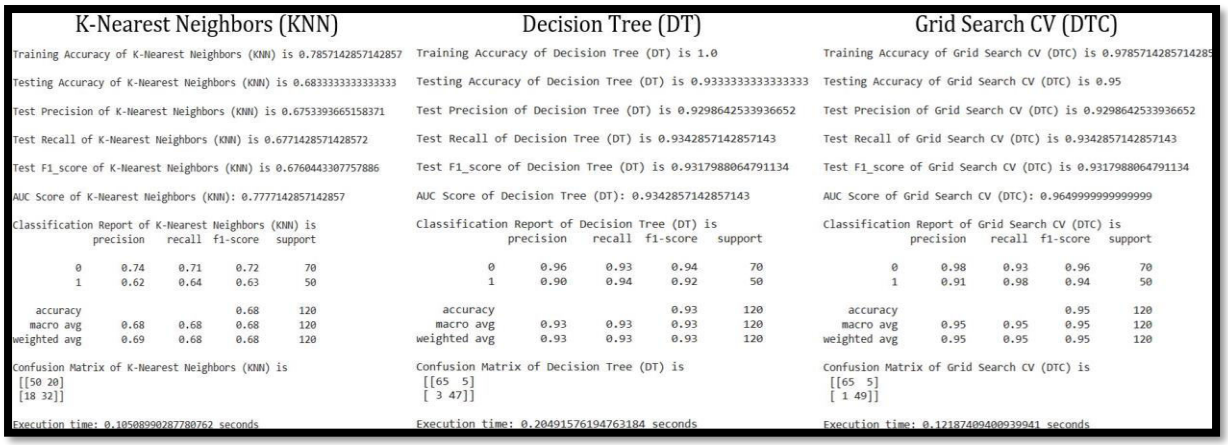
**F1 Score:** Precision and Recall's harmonic mean is the F1 Score. It offers one metric that strikes a balance between the two metrics, especially useful when they are in tension.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

**Area under the Curve (AUC):** AUC represents the area under the ROC curve, summarizing the model's performance across all classification thresholds.

- Continuous: $AUC = \int_0^1 T\,PR\,d(FPR)$
- Discrete form: $AUC = \sum_{i=1}^{n-1} \frac{(FPR_{i+1} - FPR_i)(TPR_i + TPR_{i+1})}{2}$

**Learning curves:** Learning curves are graphical representations that show how a machine learning model's performance improves over time or with an increasing amount of training data. They plot the training error and validation error (or accuracy) against the size of the training dataset or the number of training iterations.



K-Nearest Neighbors (KNN)

Training Accuracy of K-Nearest Neighbors (KNN) is 0.7857142857142857

Testing Accuracy of K-Nearest Neighbors (KNN) is 0.6833333333333333

Test Precision of K-Nearest Neighbors (KNN) is 0.6753393665158371

Test Recall of K-Nearest Neighbors (KNN) is 0.6771428571428572

Test F1_score of K-Nearest Neighbors (KNN) is 0.6760443307757886

AUC Score of K-Nearest Neighbors (KNN): 0.7777142857142857

Classification Report of K-Nearest Neighbors (KNN) is

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.74 | 0.71 | 0.72 | 70 |
| 1 | 0.62 | 0.64 | 0.63 | 50 |
| accuracy | | | 0.68 | 120 |
| macro avg | 0.68 | 0.68 | 0.68 | 120 |
| weighted avg | 0.69 | 0.68 | 0.68 | 120 |

Confusion Matrix of K-Nearest Neighbors (KNN) is

[[50 20]
[18 32]]

Execution time: 0.10508990287780762 seconds

Decision Tree (DT)

Training Accuracy of Decision Tree (DT) is 1.0

Testing Accuracy of Decision Tree (DT) is 0.9333333333333333

Test Precision of Decision Tree (DT) is 0.9298642533936652

Test Recall of Decision Tree (DT) is 0.9342857142857143

Test F1_score of Decision Tree (DT) is 0.9317988064791134

AUC Score of Decision Tree (DT): 0.9342857142857143

Classification Report of Decision Tree (DT) is

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.96 | 0.93 | 0.94 | 70 |
| 1 | 0.90 | 0.94 | 0.92 | 50 |
| accuracy | | | 0.93 | 120 |
| macro avg | 0.93 | 0.93 | 0.93 | 120 |
| weighted avg | 0.93 | 0.93 | 0.93 | 120 |

Confusion Matrix of Decision Tree (DT) is

[[65 5]
[ 3 47]]

Execution time: 0.20491576194763184 seconds

Grid Search CV (DTC)

Training Accuracy of Grid Search CV (DTC) is 0.9785714285714285

Testing Accuracy of Grid Search CV (DTC) is 0.95

Test Precision of Grid Search CV (DTC) is 0.9298642533936652

Test Recall of Grid Search CV (DTC) is 0.9342857142857143

Test F1_score of Grid Search CV (DTC) is 0.9317988064791134

AUC Score of Grid Search CV (DTC): 0.9649999999999999

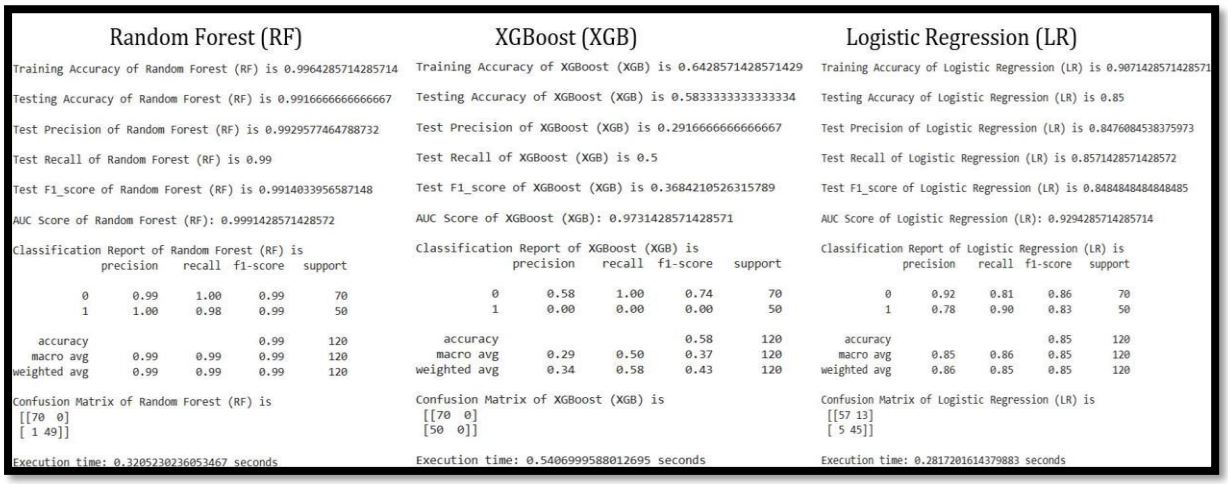
Classification Report of Grid Search CV (DTC) is

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.98 | 0.93 | 0.96 | 70 |
| 1 | 0.91 | 0.98 | 0.94 | 50 |
| accuracy | | | 0.95 | 120 |
| macro avg | 0.95 | 0.95 | 0.95 | 120 |
| weighted avg | 0.95 | 0.95 | 0.95 | 120 |

Confusion Matrix of Grid Search CV (DTC) is

[[65 5]
[ 1 49]]

Execution time: 0.12187409400939941 seconds



Random Forest (RF)

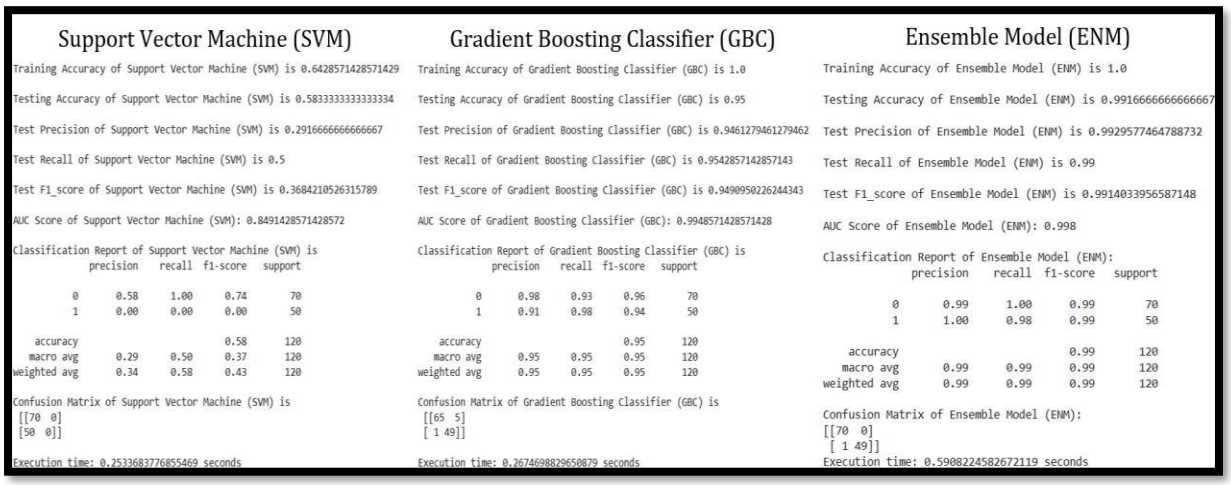
```
Training Accuracy of Random Forest (RF) is 0.9964285714285714

Testing Accuracy of Random Forest (RF) is 0.9916666666666667

Test Precision of Random Forest (RF) is 0.9929577464788732

Test Recall of Random Forest (RF) is 0.99

Test F1_score of Random Forest (RF) is 0.9914033956587148

AUC Score of Random Forest (RF): 0.9991428571428572

Classification Report of Random Forest (RF) is
              precision    recall  f1-score   support

           0       0.99      1.00      0.99        70
           1       1.00      0.98      0.99        50

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120

Confusion Matrix of Random Forest (RF) is
 [[70  0]
 [ 1 49]]

Execution time: 0.3205230236053467 seconds
```

XGBoost (XGB)

```
Training Accuracy of XGBoost (XGB) is 0.6428571428571429

Testing Accuracy of XGBoost (XGB) is 0.5833333333333334

Test Precision of XGBoost (XGB) is 0.2916666666666667

Test Recall of XGBoost (XGB) is 0.5

Test F1_score of XGBoost (XGB) is 0.3684210526315789

AUC Score of XGBoost (XGB): 0.9731428571428571

Classification Report of XGBoost (XGB) is
              precision    recall  f1-score   support

           0       0.58      1.00      0.74        70
           1       0.00      0.00      0.00        50

    accuracy                           0.58       120
   macro avg       0.29      0.50      0.37       120
weighted avg       0.34      0.58      0.43       120

Confusion Matrix of XGBoost (XGB) is
 [[70  0]
 [50  0]]

Execution time: 0.5406999588012695 seconds
```

Logistic Regression (LR)

```
Training Accuracy of Logistic Regression (LR) is 0.9071428571428571

Testing Accuracy of Logistic Regression (LR) is 0.85

Test Precision of Logistic Regression (LR) is 0.8476084538375973

Test Recall of Logistic Regression (LR) is 0.8571428571428572

Test F1_score of Logistic Regression (LR) is 0.8484848484848485

AUC Score of Logistic Regression (LR): 0.9294285714285714

Classification Report of Logistic Regression (LR) is
              precision    recall  f1-score   support

           0       0.92      0.81      0.86        70
           1       0.78      0.90      0.83        50

    accuracy                           0.85       120
   macro avg       0.85      0.86      0.85       120
weighted avg       0.86      0.85      0.85       120

Confusion Matrix of Logistic Regression (LR) is
 [[57 13]
 [ 5 45]]

Execution time: 0.2817201614379883 seconds
```



Support Vector Machine (SVM)

```
Training Accuracy of Support Vector Machine (SVM) is 0.6428571428571429

Testing Accuracy of Support Vector Machine (SVM) is 0.5833333333333334

Test Precision of Support Vector Machine (SVM) is 0.2916666666666667

Test Recall of Support Vector Machine (SVM) is 0.5

Test F1_score of Support Vector Machine (SVM) is 0.3684210526315789

AUC Score of Support Vector Machine (SVM): 0.8491428571428572

Classification Report of Support Vector Machine (SVM) is
              precision    recall  f1-score   support

           0       0.58      1.00      0.74        70
           1       0.00      0.00      0.00        50

    accuracy                           0.58       120
   macro avg       0.29      0.50      0.37       120
weighted avg       0.34      0.58      0.43       120

Confusion Matrix of Support Vector Machine (SVM) is
 [[70  0]
 [50  0]]

Execution time: 0.2533683776855469 seconds
```

Gradient Boosting Classifier (GBC)

```
Training Accuracy of Gradient Boosting Classifier (GBC) is 1.0

Testing Accuracy of Gradient Boosting Classifier (GBC) is 0.95

Test Precision of Gradient Boosting Classifier (GBC) is 0.9461279461279462

Test Recall of Gradient Boosting Classifier (GBC) is 0.9542857142857143

Test F1_score of Gradient Boosting Classifier (GBC) is 0.9490950226244343

AUC Score of Gradient Boosting Classifier (GBC): 0.9948571428571428

Classification Report of Gradient Boosting Classifier (GBC) is
              precision    recall  f1-score   support

           0       0.98      0.93      0.96        70
           1       0.91      0.98      0.94        50

    accuracy                           0.95       120
   macro avg       0.95      0.95      0.95       120
weighted avg       0.95      0.95      0.95       120

Confusion Matrix of Gradient Boosting Classifier (GBC) is
 [[65  5]
 [ 1 49]]

Execution time: 0.2674698829650879 seconds
```

Ensemble Model (ENM)

```
Training Accuracy of Ensemble Model (ENM) is 1.0

Testing Accuracy of Ensemble Model (ENM) is 0.9916666666666667

Test Precision of Ensemble Model (ENM) is 0.9929577464788732

Test Recall of Ensemble Model (ENM) is 0.99

Test F1_score of Ensemble Model (ENM) is 0.9914033956587148

AUC Score of Ensemble Model (ENM): 0.998

Classification Report of Ensemble Model (ENM):
              precision    recall  f1-score   support

           0       0.99      1.00      0.99        70
           1       1.00      0.98      0.99        50

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120

Confusion Matrix of Ensemble Model (ENM):
[[70  0]
 [ 1 49]]
Execution time: 0.5908224582672119 seconds
```

**Figure 17:** Advanced Model Analysis of KNN, DT, DTC, RF, XGB, LR, SVM, GBC, and ENM

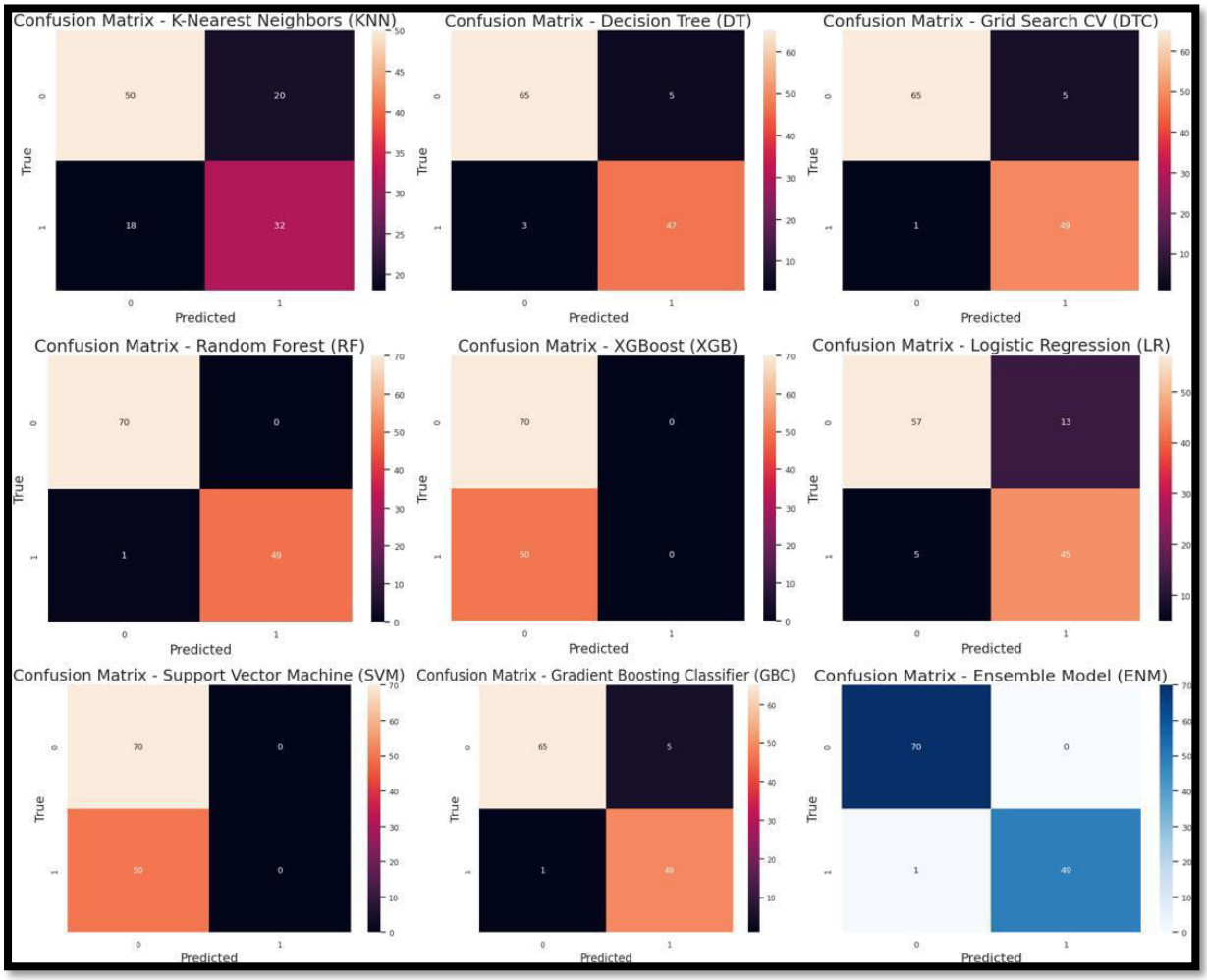



**Figure 18:** Confusion Matrix of k-Nearest Neighbors (KNN), Decision Tree (DT), Grid Search CV (DTC), Random Forest (RF), XGBoost (XGB), Logistic Regression (LR), Support Vector Machine (SVM), Gradient Boosting Classifier (GBC), Ensemble Model (ENM)

**Feature Importance Analysis Across Machine Learning Models**

Figure 19 presents the comparative feature importance rankings obtained using various machine learning models used in this study to predict kidney disease. The models are k-Nearest Neighbors (KNN), Decision Tree Classifier (DT), Grid Search CV (DTC), Random Forest (RF), Support Vector Machine (SVM), Gradient Boosting Machine (GBC), XGBoost (XGB), Logistic Regression (LR), and an Ensemble Model (ENM).

Every subplot highlights which features are most important individually for the particular model to use in making those predictions. One sees that frequently hemoglobin and packed cell volume, serum creatinine, emerge as being among the most important individuals in multiple models for predicting kidney disease.

The analysis points out the diverse rankings produced by different algorithms for features, indicating the variations in feature interaction in different paradigms of modeling. Some models such as Permutation Importance use the relative contribution of features, and intrinsic measures are used especially for tree-based models such as Random Forest and XGBoost. The above comparison allows for insights into the ranking of important biomarkers and further prioritizes the factors to be used within clinical interpretation.

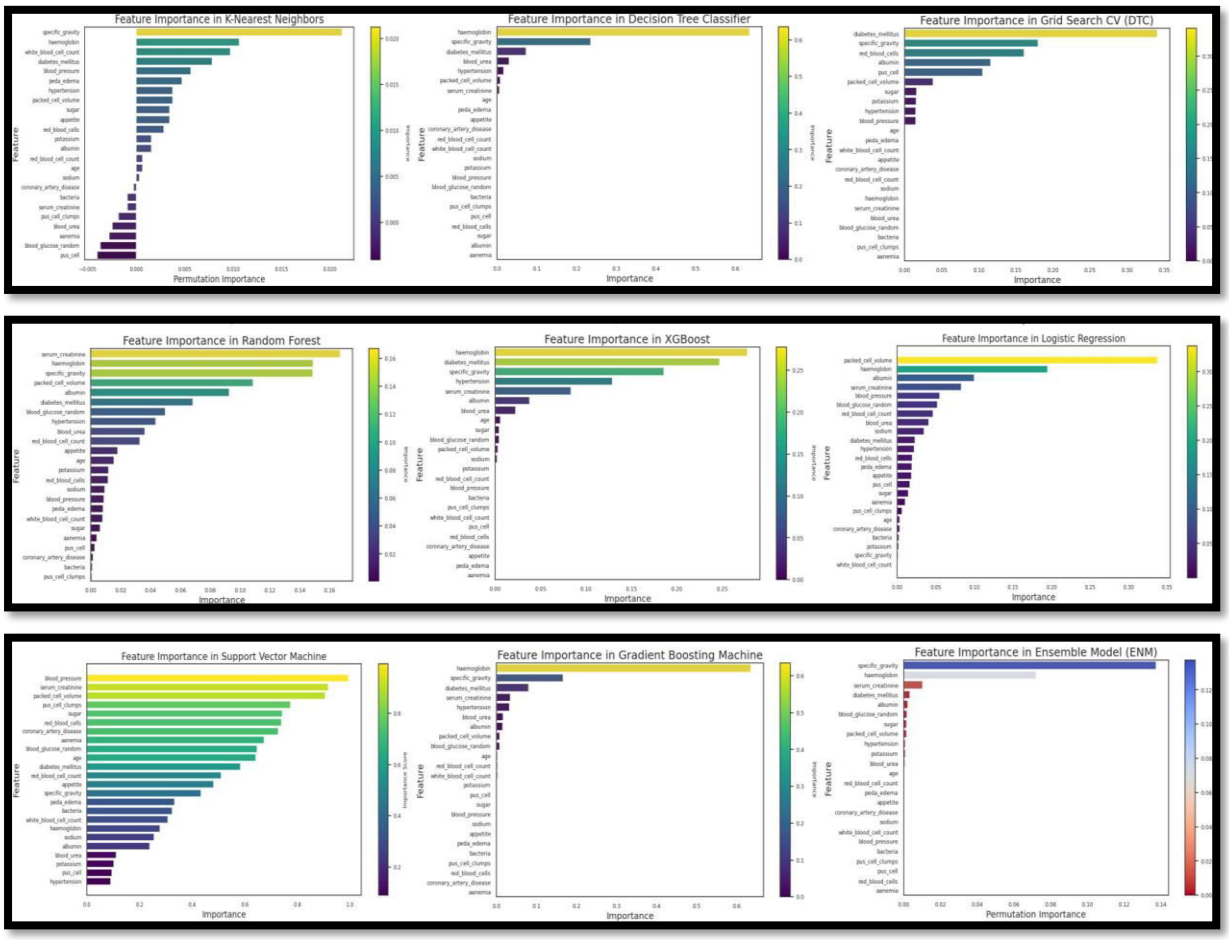



**Figure 19:** Feature Importance of k-Nearest Neighbors, Decision Tree, Grid Search CV, Random Forest, XGBoost, Logistic Regression, Support Vector Machine, Gradient Boosting Classifier, Ensemble Model

**Comprehensive Evaluation of Predictive Models for Kidney Disease**

Figure 20 provides an extensive review of predictive models used in kidney disease prediction. It contains four broad components, each dealing with a central aspect of model assessment and performance comparison.

- **ROC and Precision-Recall Analysis:** The initial two plots illustrate the ROC (Receiver Operating Characteristic) and Precision-Recall curves, which play a critical role in performance measurement for classification. The ROC curve illustrates the trade-off between specificity (false positive rate) and sensitivity

(true positive rate), and the Precision-Recall curve illustrates the trade-off between precision and recall. The curves allow models with higher discriminatory power to be distinguished. AUC (Area Under the Curve) and AUPRC (Area Under the Precision-Recall Curve) are performance measures used to quantify model performance, and RF (Random Forest), XGB (XGBoost), GBC (Gradient Boosting Classifier), and ENM (Ensemble Model) all recorded near-perfect AUC values of 1.00, indicating great prediction performance.

- **AU-ROC-Based Model Comparison:** The bottom-left figure employs a box plot to display AU-ROC score distributions between models and their stability and variability. RF, ENM, XGB, and GBC all have reliably high AU-ROC scores and are thus stable in kidney disease prediction. SVM and KNN are less stable have lower scores and necessitate strict model selection.

- **Learning Curve Analysis for Generalization:** The bottom-right plot shows the learning curves for each model's generalization to new data. Models with training and validation curves that diverge less, such as RF, ENM, and GBC, show good generalization. KNN and SVM show larger gaps, which indicate overfitting or underfitting.
  This bar chart is a key figure in the comparison of the best-performing prediction model, with insights into every algorithm's learning style, pitfalls, and capabilities regarding kidney disease prognosis and diagnosis.

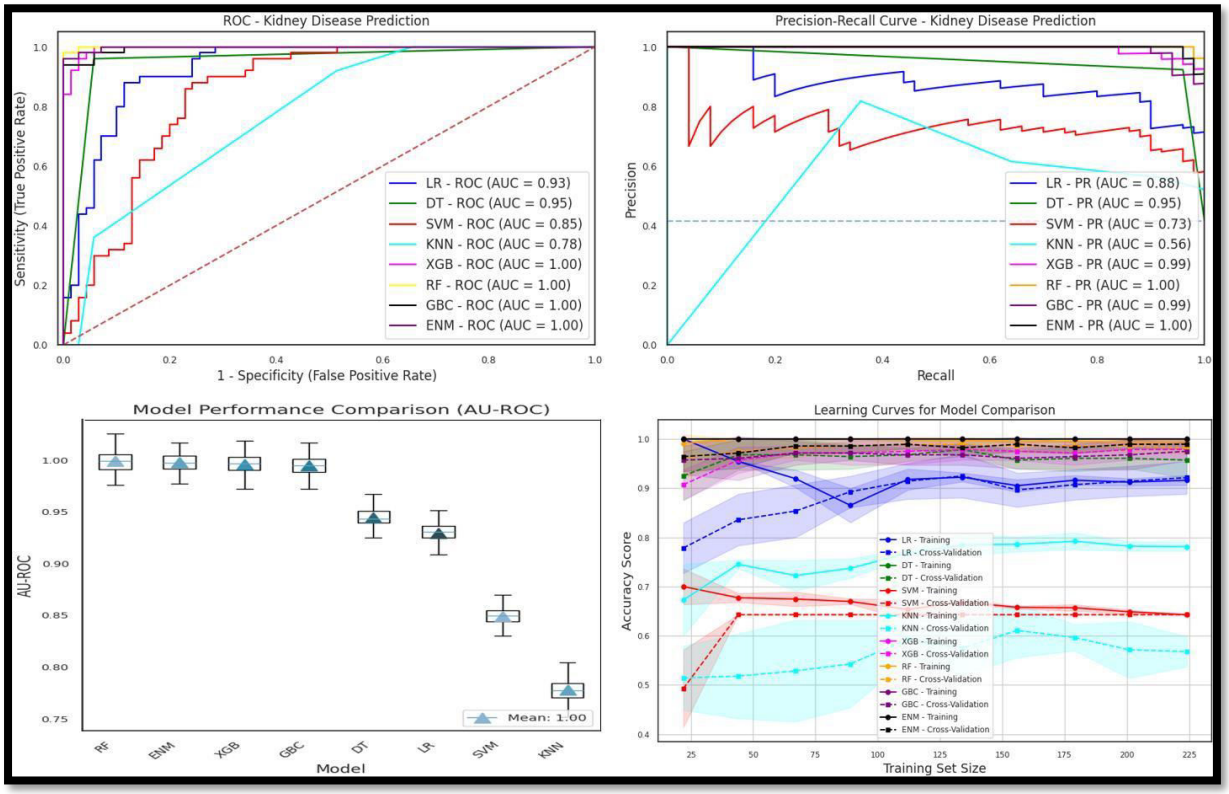


**Figure 20:** Comprehensive Model Evaluation of Model Performance and Generalization Analysis for Kidney Disease Prediction

## 8. Results and discussion

This section conducts tests of in-depth machine learning algorithms involving outcome predictions from patients with Chronic Kidney Disease. All the models that will be discussed comprise KNN, DT, DTC, RF, XGB, LR, SVM, GBC, and Ensemble Model (ENM). All the chosen models are measured on parameters that comprise accuracy, precision, recall, F1-score, and AUC.

All the evaluation metrics showed that the proposed ensemble model (ENM), by combining the strengths of both RF and GBC, outperformed each of the individual models. It showed higher accuracy, better precision, and recall, which made it more reliable and robust in CKD prediction. Additionally, the ROC and Precision-Recall curves illustrated the balance the models offer between sensitivity and specificity, making the ENM highly suitable for early diagnosis.

Most importantly, there were the confusion matrices that reflected the errors of classification, false positives, and negatives, significant parameters for the improvement of diagnostic tools. Ensemble methods were successful in dealing with imbalanced datasets and capturing complex patterns, although computational complexity and interpretability were the challenges identified.

Thus, results reflect the good potential of ENM in the early detection of CKD and represent a valid tool for diagnosis. Such findings would open avenues to future research by validation of large datasets and expansion of the approach towards other medical conditions.

| **Nameofthetechnique** | **TrainingandTestingAccuracy** | **Accuracy** | **Precision** | **Sensitivity(recall)** | **Specificity** | **F1-Score** | **AUC** | **Support** | **Execution time (seconds)** |
|---|---|---|---|---|---|---|---|---|---|
| K-Nearest Neighbors(KNN) | Training:0.78 Testing:0.68 | 0.77 | 0.74for 0, 0.62for1 | 0.71for 0, 0.64for1 | 0.02 | 0.72for 0, 0.63for1 | 0.77 | 70foro, 50for1 | 0.1050899 |
| Decision Tree(DT) | Training:1.0 Testing:0.93 | 0.95 | 0.96 for 0, 0.90 for1 | 0.93 foro, 0.94 for1 | 0.03 | 0.94for 0, 0.92for1 | 0.93 | 70foro, 50for1 | 0.2049157 |
| Grid Search CV(DTC) | Training:0.97 Testing:0.95 | 0.58 | 0.98for 0, 0.91for1 | 0.93for 0, 0.98for1 | 0.03 | 0.96for 0, 0.94for1 | 0.96 | 70foro, 50for1 | 0.1218740 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Random Forest (RF) | Training:0.99 Testing: 0.99 | 0.99 | 0.99for 0, 1.00for1 | 1.00for 0, 0.98for1 | 0.04 | 0.99for 0, 0.99for1 | 0.99 | 70foro, 50for1 | 0. 3205230 |
| XGBoost (XGB) | Training:0.64 Testing:0.58 | 0.58 | 0.58for 0, 0.00for1 | 1.00for 0, 0.00for1 | 0.41 | 0.74for 0, 0.00for1 | 0.97 | 70foro, 50for1 | 0.5406999 |
| Logistic Regression (LR) | Training:0.90 Testing:0.85 | 0.85 | 0.92 for 0, 0.78 for1 | 0.81 foro, 0.90 for1 | 0.15 | 0.86for 0, 0.83for1 | 0.92 | 70foro, 50for1 | 0.2817201 |
| Support Vector Machine(SVM) | Training:0.64 Testing:0.58 | 0.58 | 0.58for 0, 0.00for1 | 1.00for 0, 0.00for1 | 0.22 | 0.74for 0, 0.00for1 | 0.84 | 70foro, 50for1 | 0.2533683 |
| Gradient Boosting Classifier(GBC) | Training:1.0 Testing:0.95 | 0.95 | 0.98for 0, 0.91for1 | 0.93for 0, 0.98for1 | 0.03 | 0.96for 0, 0.94for1 | 0.99 | 70foro, 50for1 | 0.2674698 |
| Ensemble Model (ENM) | Training:1.0 Testing:0.99 | 0.99 | 0.99for 0, 1.00for1 | 1.00for 0, 0.98for1 | 0.04 | 0.99for 0, 0.99for1 | 0.99 | 70foro, 50for1 | 0.5908224 |

**Table 1:** Comparison of Training and Testing Accuracy, Precision, Recall, Specificity, F1-Score, AUC and Support for various models in table format

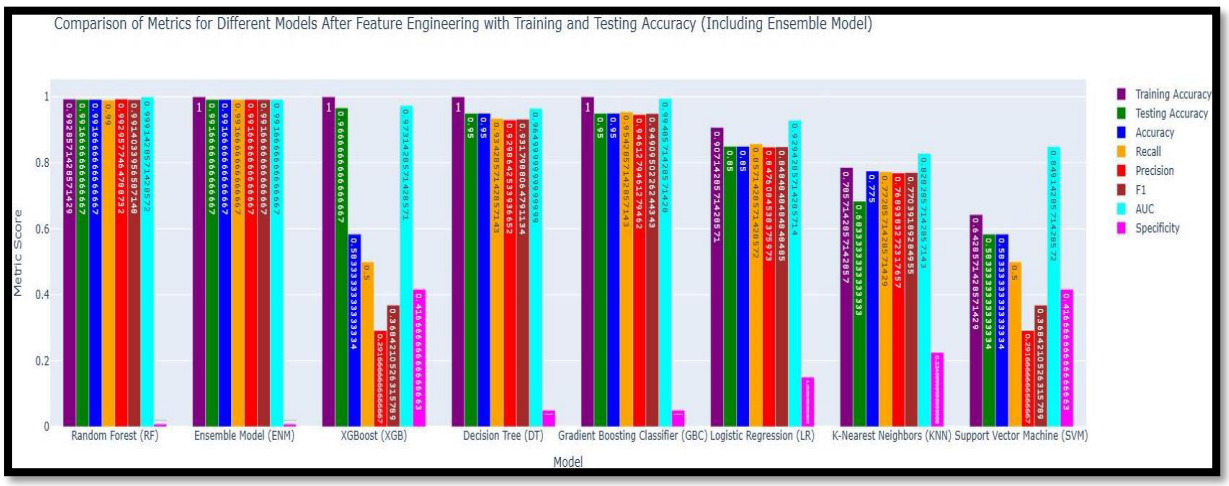



**Figure 21:** Comparison of Metrics for Different Models After Feature Engineering with Training and Testing Accuracy

## 9. Conclusion

This work describes the prediction of the existence of CKD based on clinical and diagnostic data through machine learning algorithms. Models are compared with KNN, DT, DTC, RF, GBC, LR, KNN, and SVM. The result of the comparison shows that the best two algorithms to predict the presence of CKD from the above models are RF and GBC.

Besides, as a result, analysis was proposed to develop the new Ensemble Model named ENM with strength from both RF and GBC. The comparison results were there regarding superior performance in accuracy, precision, recall, and F1-score in comparison with individual algorithms. ENM proved even more robust and reliable regarding the pattern complexity dealing with imbalanced data sets and was thus robust.

This also explains preprocessing steps such as data cleaning and feature selection, which have greatly improved the model performance. Furthermore, analysis of ROC curves, confusion matrices, and feature importance shows that the proposed methods are capable of reliable discrimination between cases of CKD.

This may generally indicate potential ML-based solutions for improving earlier diagnosis and early intervention in patients with CKD. Future studies on the ENM should be applied to larger diverse datasets and establish its adaptability to other disease conditions. A new door toward innovative and effectual AI-based healthcare tools will, therefore, emerge.

## 10. Future scope

This study lays the foundation for future machine learning developments within healthcare applications and therefore CKD prediction, the important future directions are:

- **Validation on Larger Datasets:** Testing the Ensemble Model on larger multi-center datasets with more diversified demographics would enhance its generalizability.
- **Real-time implementation:** Developing user-friendly applications will ease their seamless integration into clinical workflow to enable early diagnosis of CKD.
- **Feature Optimization:** Selecting the more advanced features may enhance model performance but minimize computation complexity.
- **Broader Applicability:** The application of ENM to other chronic diseases will create wider relevance to medical diagnostics.
- **Explainable AI:** Coupling the use of explainable AI techniques will boost clinician trust and acceptance of AI tools.
- **Wearable Integration:** It will ensure continuous monitoring and the early detection of CKD by incorporating data from wearable devices.

These innovations can propel novel, AI-based solutions toward improved healthcare outcomes.

**References:**

1. Bansa, V. and Sindhu, R. (2024). A comparative study on chronic kidney disease using different machine learning techniques. J. Electrical Systems, 20-1os: 72–82.
2. Debal, D.A. and Sitote, T.M. (2022). Chronic kidney disease prediction using machine learning techniques. Journal of Big Data, 9:109.
3. Xiao, J., Ding, R., Xu, X., Guan, H., Feng, X., Sun, T., Zhu, S. and Ye, Z. (2019). Comparison and development of machine learning tools in the prediction of chronic kidney disease progression. J. Transl. Med., 17:119.
4. Farjana, A., Liza, F.T., Das, M.C. and Hasan, M. (2024). Predicting chronic kidney disease using machine learning algorithms. ResearchGate.
5. Baswaraj, D., Chatrapathy, K., Prasad, M.L., Pughazendi, N., Kiran, A., Partheeban, N. and Reddy, P.C.S. (2024). Chronic kidney disease risk prediction using machine learning techniques. Journal of Information Technology Management.
6. Khan, N., Raza, M.A., Mirjat, N.H., Balouch, N., Abbas, G., Yousef, A. and Touti, E. (2024). Unveiling the predictive power: a comprehensive study of machine learning models for anticipating chronic kidney disease. Frontiers in Artificial Intelligence.
7. Khalid, H., Khan, A., Khan, M.Z., Mehmood, G. and Qureshi, M.S. (2023). Machine learning hybrid model for the prediction of chronic kidney disease. Computational Intelligence and Neuroscience.
8. Ifraz, G.M., Rashid, M.H., Tazin, T., Bourouis, S. and Khan, M.M. (2024). Comparative analysis for prediction of kidney disease using intelligent machine learning methods. AI in Healthcare.
9. Mandava, S., Vinta, S.R., Ghosh, H. and Rahat, I.S. (2024). A comparative analysis of machine learning and deep learning approaches for prediction of chronic kidney disease progression. EAI Endorsed Transactions on Internet of Things.
10. Roy, M.S., Ghosh, R., Goswami, D. and Karthik, R. (2021). Comparative analysis of machine learning methods to detect chronic kidney disease. Journal of Physics: Conference Series, 1911:012005.
11. Jadhav, V., Baste, V., Andhare, P. and Gaurav, A. (2023). Comparative analysis of kidney disease prediction using machine learning and deep learning. JETIR, 10(5).
12. Tejasree, E. and Ramana, K.V. (2023). Comparing machine learning approaches for chronic kidney disease prediction. IJCRT, 11(9).

13. Gulati, V. and Raheja, N. (2021). Comparative analysis of machine learning techniques based on chronic kidney disease dataset. IOP Conf. Series: Materials Science and Engineering, 1131:012010.
14. Bhoomika, C.M. and Harish, B.G. (2024). Chronic kidney disease prediction using machine learning. International Research Journal of Modernization in Engineering Technology and Science, 6(7).