Chronic Kidney Disease

Artificial Intelligence Group Project Report

Group Members:
Abdul Zahid Shaik - ashaik@pdx.edu - 914429560
Nida Mariam Shaikh Aslam - nidama@pdx.edu - 958577752
Snehil Shrivastava - snehils@pdx.edu - 906881230
Mohammed Taariq Mansurie - mansurie@pdx.edu - 967032383

CS 541: Artificial Intelligence
Portland State University
Spring 2024

Individual Contributions:

Nida Mariam Sheikh Aslam: I worked on the data preparation and preprocessing steps which included, addressing issues such as missing values(using Iterative Imputer and random imputation) and inconsistencies in the formatting of the categorical variables(using Label Encoder to convert categorical values to numeric values). I also contributed to the visualization of the distribution of the Numerical Variables, Categorical Variables, the Correlation Matrix and missing values. I implemented a custom function of the KNN algorithm and did the evaluation to identify the effectiveness of the model in predicting the disease. I also contributed to the documentation.

Snehil Shrivastava: In my contributions, I focused on several key areas including data preprocessing and model evaluation. I played a significant role in preparing the data by implementing techniques such as StandardScaler for feature scaling and train-test split. I also worked on the model evaluation phase, implementing functions to train, predict, and evaluate classifiers. These functions handle the classifier, model_name, X_train, Y_train, X_test, Y_test, producing metrics like confusion matrix, accuracy, precision, recall, and F1 score in a dictionary format. This process ensured a thorough assessment of model performance in predicting disease outcomes. Moreover, I implemented a custom Logistic Regression function and conducted evaluations to assess its predictive accuracy for disease outcomes. I also worked on documentation.

Mohammed Taariq Amin Mansurie : I developed and evaluated the Gaussian Naive Bayes algorithm for diagnosing chronic kidney disease due to its simplicity and efficiency in handling medical data with assumptions of feature independence and Gaussian distributions. Implemented alongside classifiers like Decision Tree, KNN, and Logistic Regression, it demonstrated approximately 95% accuracy on both training and new data, balancing precision and recall. This highlighted its robustness and suitability for healthcare applications. Additionally, I explored models such as SVM, Random Forest, and boosting techniques like XGBoost, which did not perform as expected. I shortlisted the most effective algorithms based on their performance and documented the report, demonstrating their strengths and applications in diagnosing chronic kidney disease.

Abdul Zahid Shaik :
In this project, I implemented a Decision Tree classifier from scratch, focusing on creating a model capable of accurately predicting Chronic Kidney Disease (CKD) from the dataset.Achieved good performance with a **training accuracy of 71%** and a **test accuracy of 55%**, indicating the model's strong predictive capability and good generalization on unseen data. I also compared solving with sklearn to see the results which gave highest accuracy than my current method, then to scratch code I tried making changes to the model like using ensemble methods, making changes to data to get more accuracy. I compared its performance with other models, including K-Nearest Neighbors (KNN), Logistic Regression, and Gaussian Naive Bayes. In addition to the model implementation and evaluation, I conducted research on future improvements and extensions of our project like Improving Model Robustness, Expanding Data Sources, Advanced Machine Learning Techniques, Clinical Validation etc.

Table Of Contents:

I. Abstract..4
II. Background...4
a. Objective...4
b. Dataset..4
III. Implementation...4
a. Data Loading and preparation..4
b. Exploring Data Analysis..5
c. Missing Data Imputation...8
d. Feature Encoding..10
e. Model Development..10
f. Model Evaluation...10
IV. Models..10
a. Decision Tree Classifier:...10
b. Logistic Regression...11
c. KNN...11
d. Naive Bayesian:...12
V. Experimental Results:..13
a. Decision Tree Classifier:...13
b. Logistic Regression:..14
c. KNN:..15
d. Naive Bayesian:...16
VI. Conclusion:...18
VII. Future Work:..19
VIII. References:...20
IX. Code Link:..20

I. Abstract

In this project, we worked on the classification of Chronic Kidney Disease (CKD) using various machine learning techniques. The primary objective was to develop and evaluate different predictive models to accurately classify CKD based on the patient data. We compared models including K-Nearest Neighbors (KNN), Gaussian Naive Bayes, Logistic Regression, and Decision Tree.

The dataset that we used consists of patient attributes with both numerical and categorical features. The data preprocessing included data visualization and handling missing values through imputation, encoding categorical variables, and standardizing numerical features. Various different plots were used to visualize the data distribution, missing values and model performance, this aided us to get some insights on the dataset and to evaluate the performance of each model. Then the dataset was split into training and testing sets. The models were then trained and evaluated based on metrics such as accuracy, precision, recall, F1 score, and confusion matrices. The KNN, SVM and Gaussian Naive Bayes algorithms were implemented from scratch in order to get a better understanding of their working.

This work highlights the importance of AI methods in medical diagnostics and their potential to provide reliable predictive models for diseases like CKD, ultimately helping in early diagnosis and better patient treatment.

II. Background

a. Objective

The main objective of this project is to develop and evaluate different Artificial Intelligence models to accurately predict the classification of Chronic Kidney Diseases(CKD). Through data preprocessing and implementing various classification algorithms we aim to compare the performance of different models and identify the model that performs the best. Ultimately, aiding in early diagnosis and improved patient management.

b. Dataset

The Chronic Kidney Disease(CKD) dataset contains the data of 400 patients, and contains 24 features, which are used to predict the presence of CKD. These features include, blood pressure, blood glucose levels, red blood cell count, etc., which are all used as indicators of Kidney function and health. The dataset contains the target variable, i.e.,the class ,hence, can be used for supervised learning tasks. The dataset can be found at - Dataset.

III. Implementation

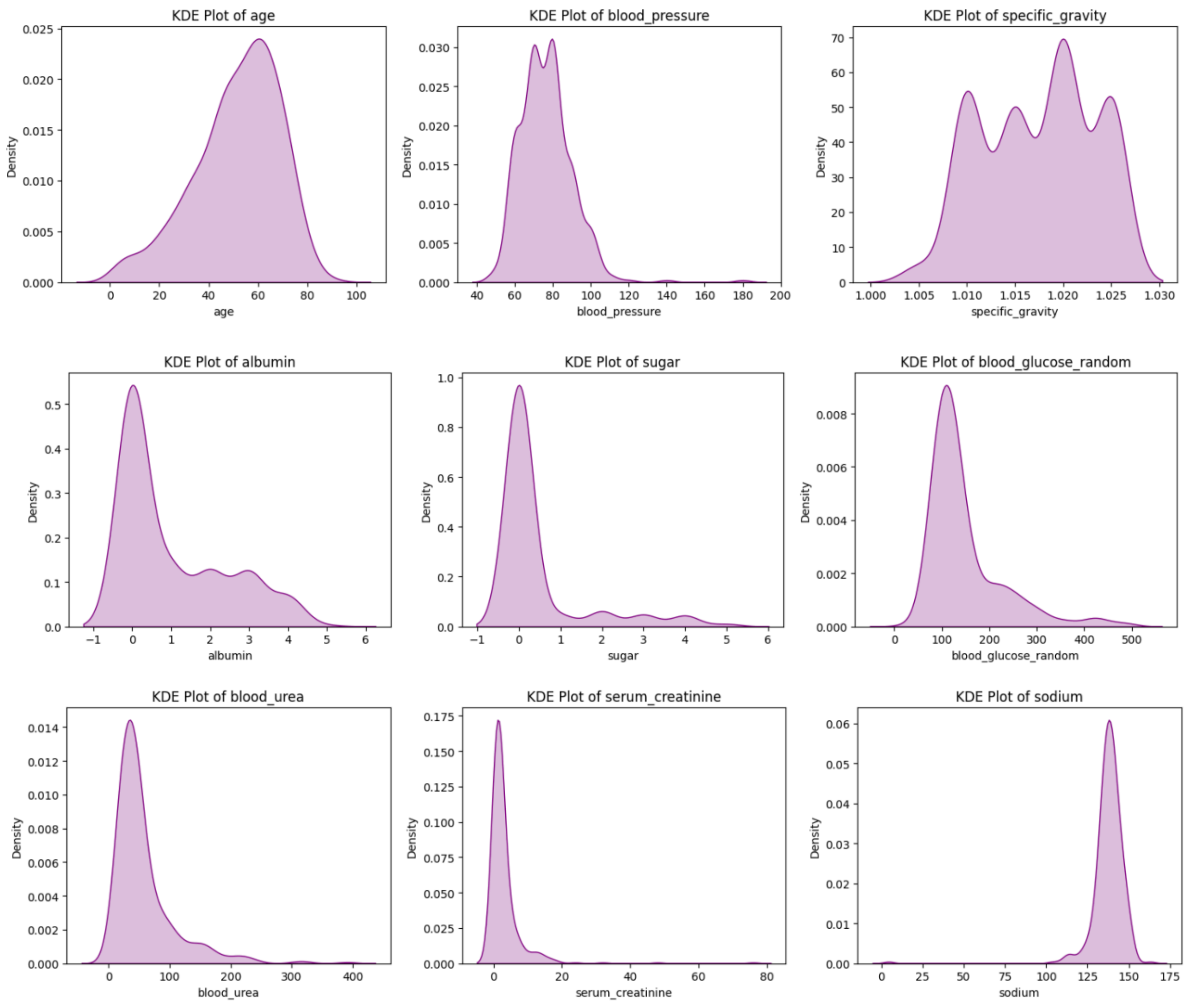
a. Data Loading and preparation

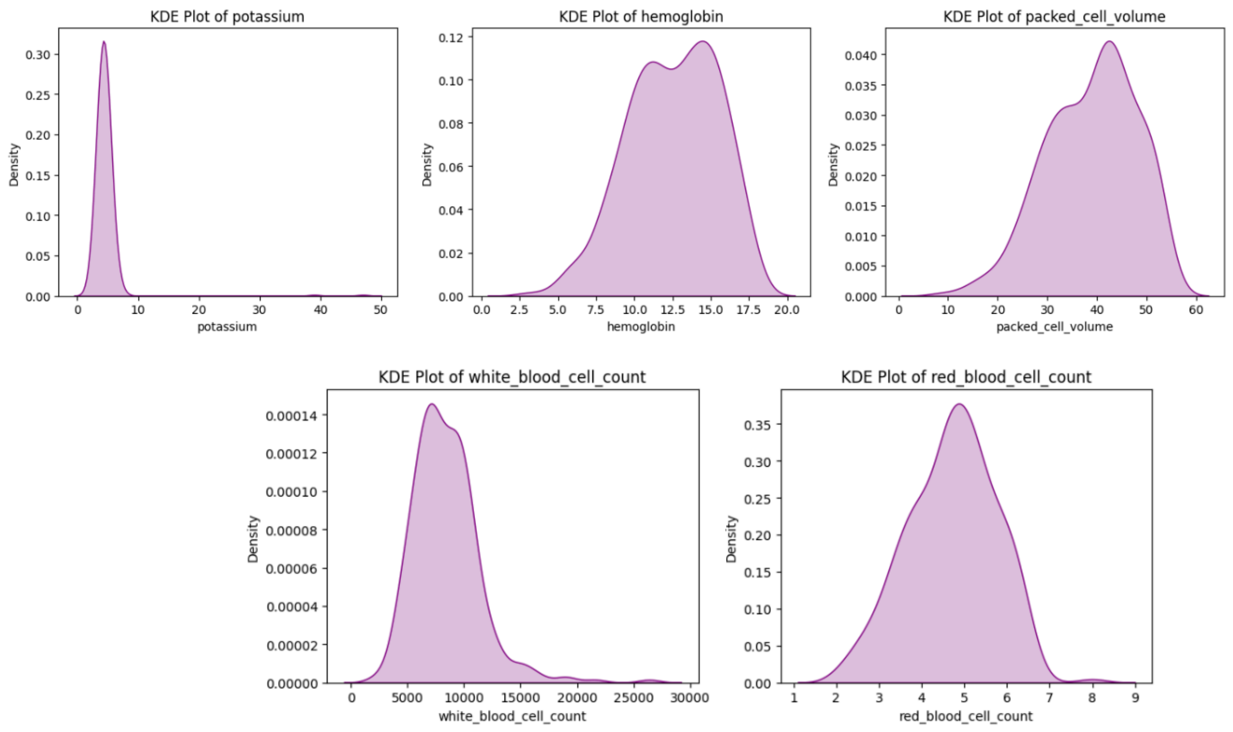
We started by loading the dataset into a Pandas Dataframe by using the ‘read_csv’ method from the pandas library. Then we checked the dataset for any missing values as well as for duplicate rows. For the ease of understanding and using the dataset we renamed the columns to elaborate on what they represent. This aided us in identifying the columns that had numerical values but had object datatype, namely *packed_cell_volume, white_blood_cell_count and red_blood_cell_count*, they were converted to numerical datatype.

After identifying all the unique possible values in each column of the dataset, we separated the numerical columns and categorical columns. Later we cleaned up the categorical columns to remove non-standard values.

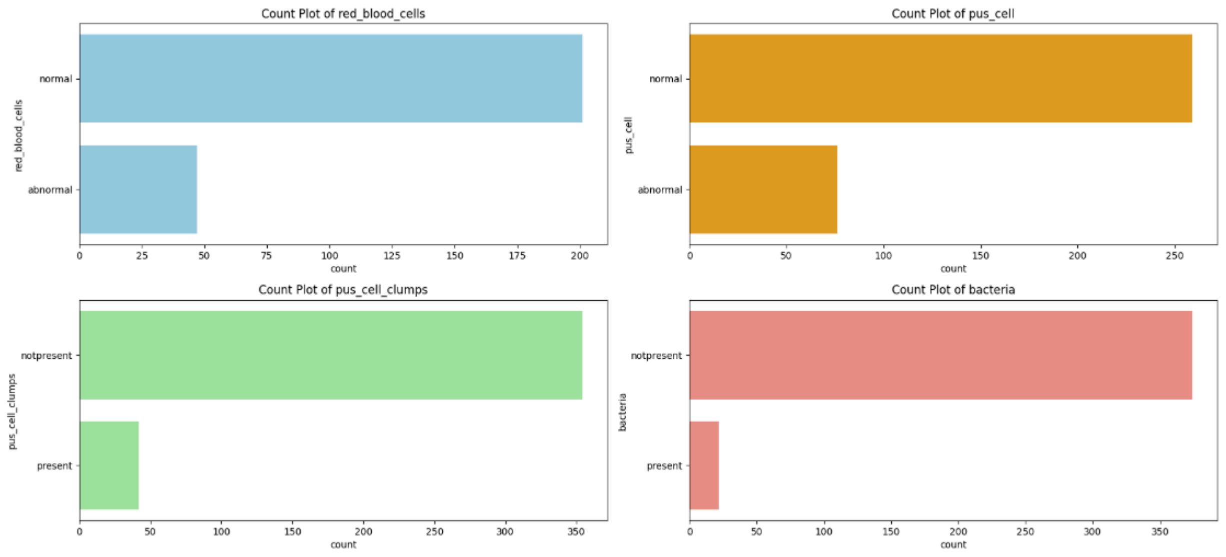
b. Exploring Data Analysis

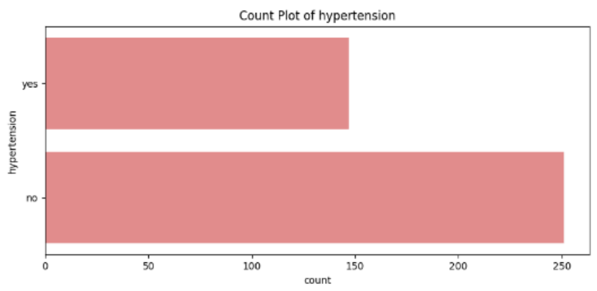
We visualize the distribution of the data to identify patterns and understand what factors affect the final disease diagnosis. The distribution of the numeric columns was done using Kernel Density Estimation (KDE) plots as follows:

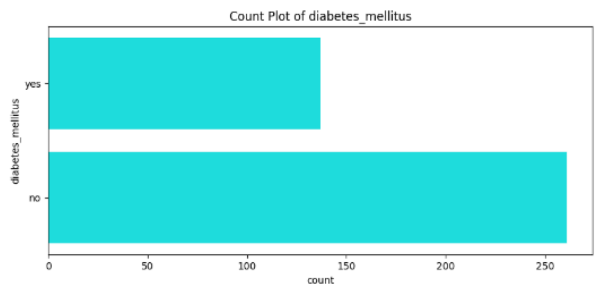


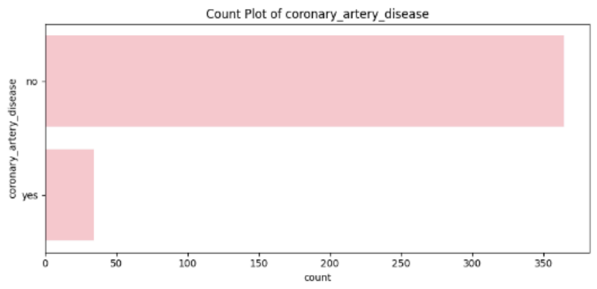


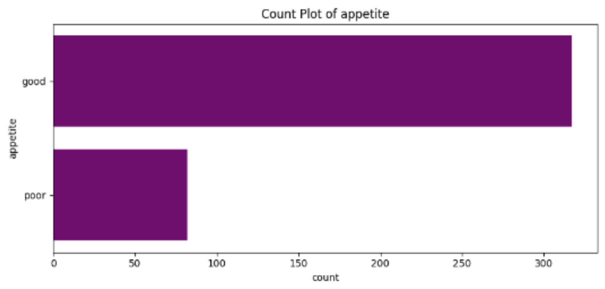
Then we visualized the Categorical Variables using count plots as follows:

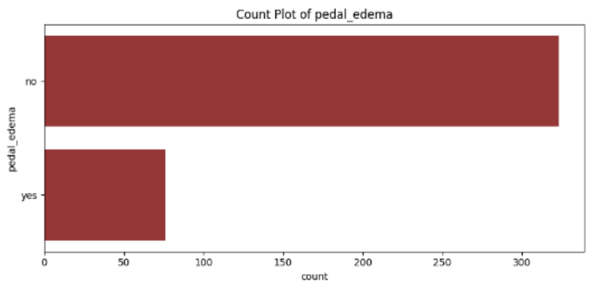


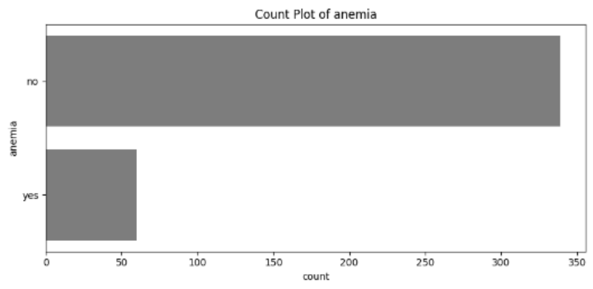




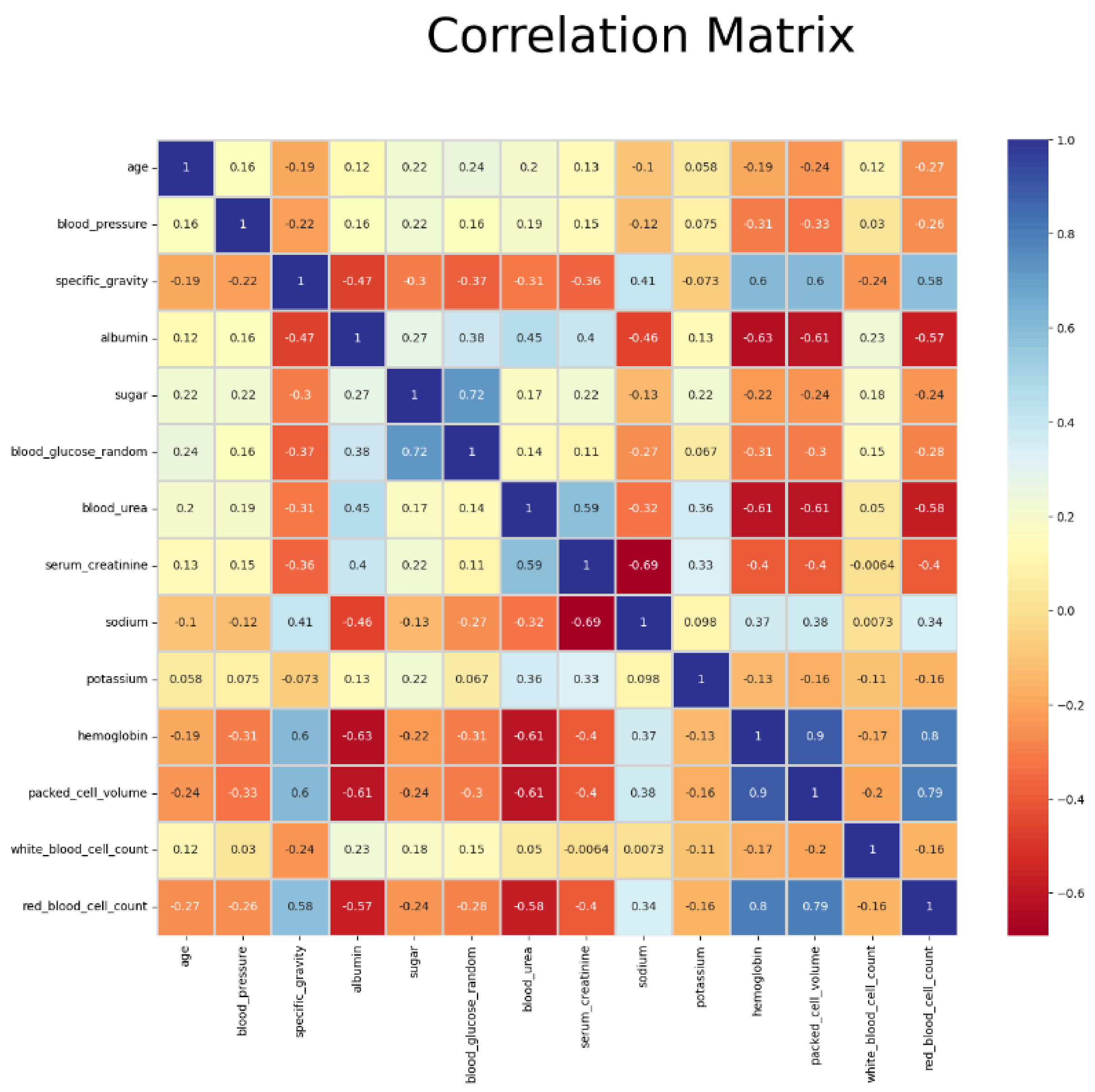








Using the correlation matrix we tried to identify and visualize the linear relationship between each pair of values in the dataset.



	age	blood_pressure	specific_gravity	albumin	sugar	blood_glucose_random	blood_urea	serum_creatinine	sodium	potassium	hemoglobin	packed_cell_volume	white_blood_cell_count	red_blood_cell_count
age	1	0.16	-0.19	0.12	0.22	0.24	0.2	0.13	-0.1	0.058	-0.19	-0.24	0.12	-0.27
blood_pressure	0.16	1	-0.22	0.16	0.22	0.16	0.19	0.15	-0.12	0.075	-0.31	-0.33	0.03	-0.26
specific_gravity	-0.19	-0.22	1	-0.47	-0.3	-0.37	-0.31	-0.36	0.41	-0.073	0.6	0.6	-0.24	0.58
albumin	0.12	0.16	-0.47	1	0.27	0.38	0.45	0.4	-0.46	0.13	-0.63	-0.61	0.23	-0.57
sugar	0.22	0.22	-0.3	0.27	1	0.72	0.17	0.22	-0.13	0.22	-0.22	-0.24	0.18	-0.24
blood_glucose_random	0.24	0.16	-0.37	0.38	0.72	1	0.14	0.11	-0.27	0.067	-0.31	-0.3	0.15	-0.28
blood_urea	0.2	0.19	-0.31	0.45	0.17	0.14	1	0.59	-0.32	0.36	-0.61	-0.61	0.05	-0.58
serum_creatinine	0.13	0.15	-0.36	0.4	0.22	0.11	0.59	1	-0.69	0.33	-0.4	-0.4	-0.0064	-0.4
sodium	-0.1	-0.12	0.41	-0.46	-0.13	-0.27	-0.32	-0.69	1	0.098	0.37	0.38	0.0073	0.34
potassium	0.058	0.075	-0.073	0.13	0.22	0.067	0.36	0.33	0.098	1	-0.13	-0.16	-0.11	-0.16
hemoglobin	-0.19	-0.31	0.6	-0.63	-0.22	-0.31	-0.61	-0.4	0.37	-0.13	1	0.9	-0.17	0.8
packed_cell_volume	-0.24	-0.33	0.6	-0.61	-0.24	-0.3	-0.61	-0.4	0.38	-0.16	0.9	1	-0.2	0.79
white_blood_cell_count	0.12	0.03	-0.24	0.23	0.18	0.15	0.05	-0.0064	0.0073	-0.11	-0.17	-0.2	1	-0.16
red_blood_cell_count	-0.27	-0.26	0.58	-0.57	-0.24	-0.28	-0.58	-0.4	0.34	-0.16	0.8	0.79	-0.16	1

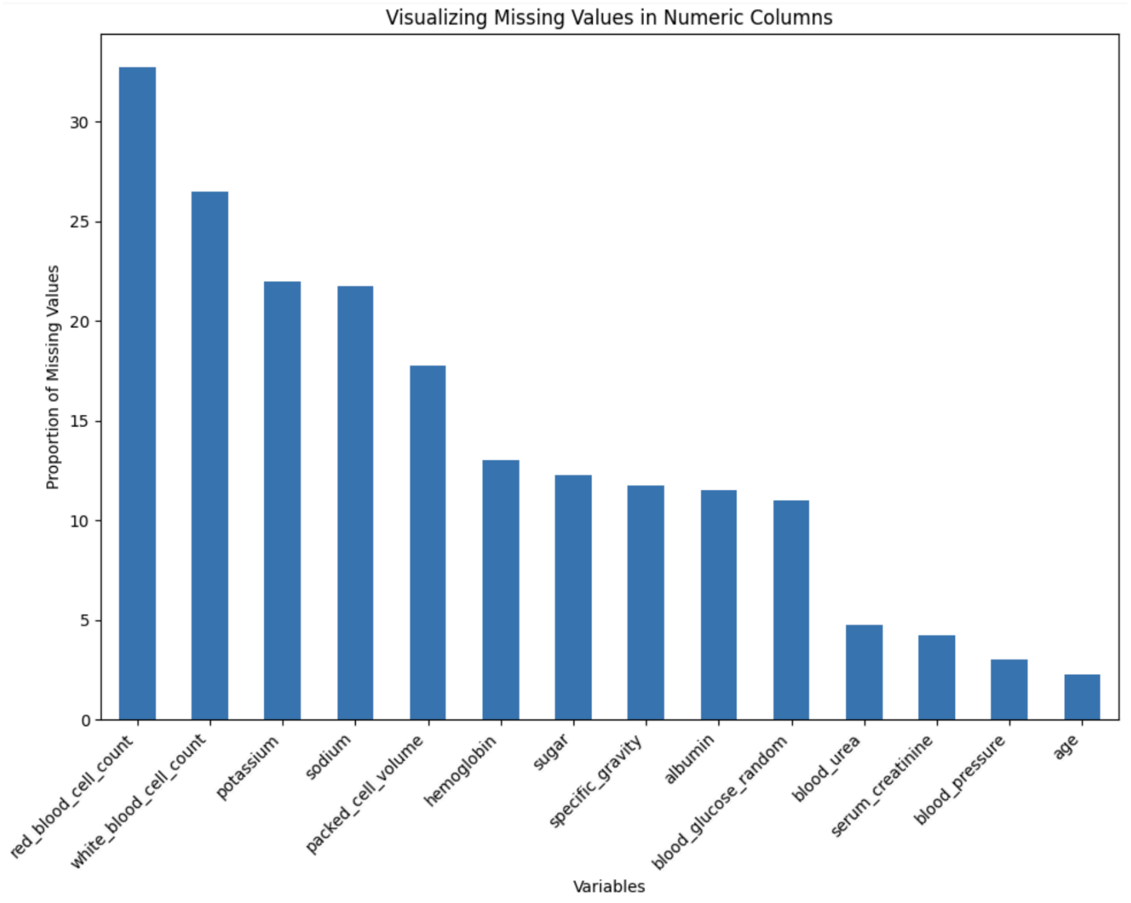
c. Missing Data Imputation

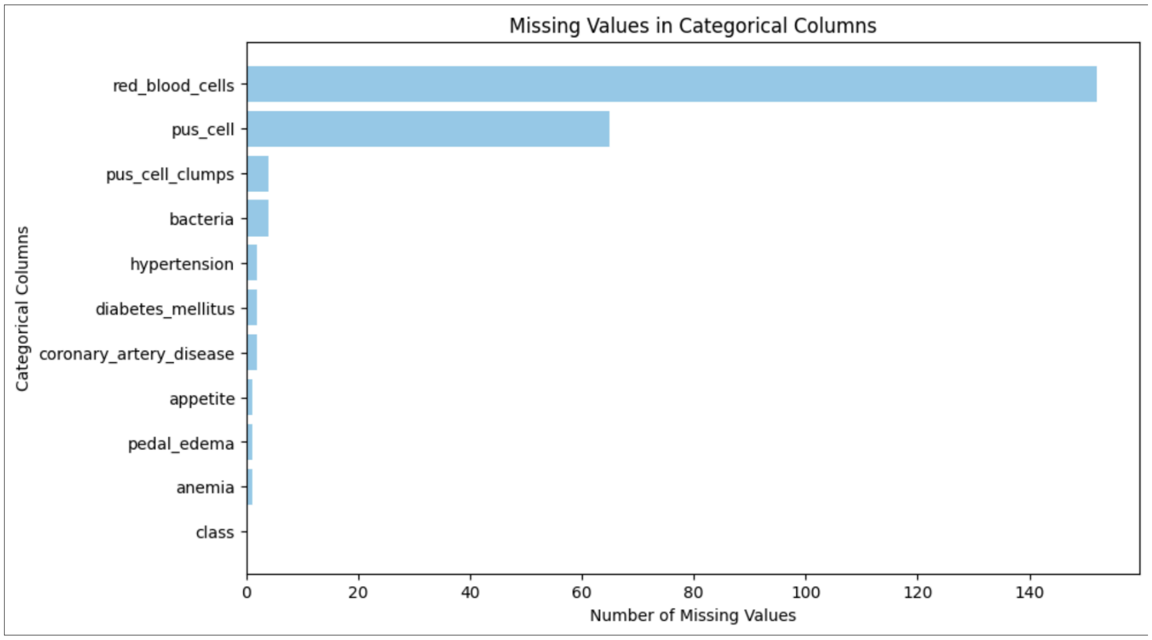
1. Identifying Missing Values:

First we are determining the extent of missing values in both numerical and categorical columns.

2. Visualizing Missing Values:

Providing a clear visual representation of the proportion and distribution of missing values in the dataset.





3. Handling Missing Values:

- Iterative Imputation for Numeric Columns: We are using scikit-learn's IterativeImputer to fill missing values in the numeric columns. The IterativeImputer leverages the relationship of that column with other features to predict what the missing value could be.

- Random Value Imputation for Specific Features: For each column, missing values are replaced with randomly selected non-missing values from the same column. This approach preserves the variability in the input data.

- Mode Imputation for Categorical Columns: Filled the missing values with the most frequent value (mode) of each categorical column.

Through a combination of iterative imputation, random value imputation, and mode imputation, we effectively handled missing values in the dataset. This ensured that the dataset retained its integrity and variability, enhancing the predictive power of our subsequent models.

d. Feature Encoding

We utilized the 'LabelEncoder' to transform categorical values into numerical ones, making them suitable for models. This conversion enhances the models' performance and accuracy.

e. Model Development

Four classification algorithms—Decision Tree Classifier, K-Nearest Neighbors (KNN), Logistic Regression, and Gaussian Naive Bayes—were trained on a dataset split into 50% training and 50% testing sets for chronic kidney disease diagnosis. Each model was tailored to leverage distinct learning approaches: decision rules (Decision Tree), instance similarity (KNN), linear relationships (Logistic Regression), and feature independence assumptions (Gaussian Naive Bayes). These algorithms were prepared to predict disease presence based on medical features.

f. Model Evaluation

The trained models underwent evaluation on unseen testing data to assess their performance using metrics such as training accuracy for model fit, test accuracy for predictive ability on new data, and precision, recall, and F1 score for classification effectiveness. Gaussian Naive Bayes emerged as the top performer with the highest test accuracy (97.5%) and recall (97.97%), crucial for accurate identification of positive cases in medical diagnostics. This evaluation process facilitated the selection of the most effective model for precise chronic kidney disease diagnosis based on dataset characteristics and performance metrics.

IV. Models

a. Decision Tree Classifier:

The Decision Tree classifier recursively partitions the feature space into regions based on the values of features. It is capable of handling both numerical and categorical data and is particularly effective for classification tasks.

Model Construction:

- **DecisionTreeClassifier** Class: Implemented from scratch without using scikit-learn.
- **Initialization (__init__ method):** Sets parameters such as max_depth (maximum depth of the tree) and min_samples_split (minimum samples required to split a node).
- **Node Splitting Methods** (_entropy and _gini): Compute the entropy and Gini impurity to measure the quality of splits.
- **Finding the Best Split** (_find_best_split method): Iteratively searches for the best feature and value to split the data based on minimizing impurity.
- **Building the Tree** (_build_tree method): Recursively constructs the decision tree by splitting nodes until a stopping criterion is met (either maximum depth or pure nodes).
- **Fitting the Model (fit method):** Uses the training data to build the decision tree structure.
- **Predicting** (_predict_instance and predict methods): Traverses the decision tree to predict the class label for new instances.
- Evaluation Metrics:

- **evaluate_model Function**: Evaluates the model's performance using various metrics calculated on both training and test datasets.
- **Metrics**: Training accuracy, test accuracy, precision, recall, F1-score, confusion matrix, and classification report.

b. Logistic Regression

Logistic Regression is a fundamental technique used for binary classification tasks. In this case we have used Logistic Regression to predict the presence of CKD based on medical data.

To better understand the logistic Regression model, we have written a custom function as follows:

- Initialization: We have first initialized the LogisticRegression with a learning rate and number of iterations.
- Sigmoid Function: The ‘Sigmoid’ method is defined to compute the sigmoid of the input.
- Training: The ‘fit’ method is defined which uses gradient descent to optimize the weight and bias. It computes the linear combination of weights and features, applies the sigmoid function, computes the gradients, and updates the weights and bias.
- Prediction: The ‘predict’ method computes the linear combination, applies the sigmoid function, and classifies the results as 0 or 1 based on the threshold of 0.5.
- Evaluation: Lastly we are using the ‘evaluate_model’ function which is used to train, predict and evaluate the model.

c. KNN

The K-Nearest Neighbors(KNN) is a simple instance based learning algorithm that is used for classification. In this case we have used KNN to classify instances of CKD based on various medical attributes.

To better understand the KNN model, we have written a custom function as follows:

- Initialization: We first instantiate the KNN class with the specified number of neighbors ‘k’ which represents the number of neighbors that are considered when making the predictions.
- Training: Since KNN is a lazy learning algorithm that does not have an explicit learning phase, the ‘fit’ method simply stores the training data.
- Prediction:
    - For each test instance the ‘predict’ method finds the Euclidean distance between the test instance and all training instances.
    - Then the ‘k’ nearest neighbors are identified by sorting the distances.
    - Based on the most frequent class in these ‘k’ nearest neighbors, that class is assigned as the predicted class for our test instance.

d. Naive Bayesian:

Gaussian Naive Bayes (GNB) is a variant of the Naive Bayes algorithm, specifically designed for continuous data that follows a Gaussian (normal) distribution. Naive Bayes is a probabilistic classifier based on Bayes' Theorem, which assumes independence between predictors.

The custom Gaussian Naive Bayes classifier in the provided code is designed to classify instances of chronic kidney disease (CKD) based on a set of medical features. This model can be used in healthcare applications to aid in the early diagnosis and treatment of CKD by predicting the presence or absence of the disease.
It has quite a few applications such as:

- Spam detection
- Document classification
- Sentiment analysis
- Medical diagnosis.

For better understanding of the Gaussian Naive Bayes, we have written a custom function as follows:

- Initialization: The custom GaussianNaiveBayes class is initialized without any parameters. Initialization occurs implicitly when an instance of the class is created.

- Training: The training process involves fitting the model to the training data, which entails calculating the mean, variance, and prior probability for each class.
    - Unique Classes Identification: Identify unique classes in the target variable y.
    - Parameters Calculation: For each class:
        i. Calculate the mean and variance of each feature.
        ii. Compute the prior probability of the class.

- Prediction: Prediction involves calculating the posterior probability for each class and selecting the class with the highest posterior probability.

- So, for each feature, we are calculating the likelihood using the Gaussian probability density function. These are the likelihood calculations.
- For each class, we are computing the posterior probability by summing the log-likelihoods and adding the log-prior.
- Lastly, select the class with the highest posterior probability.
- Additionally, the model can predict the class probabilities.

V. Experimental Results:

a. Decision Tree Classifier:

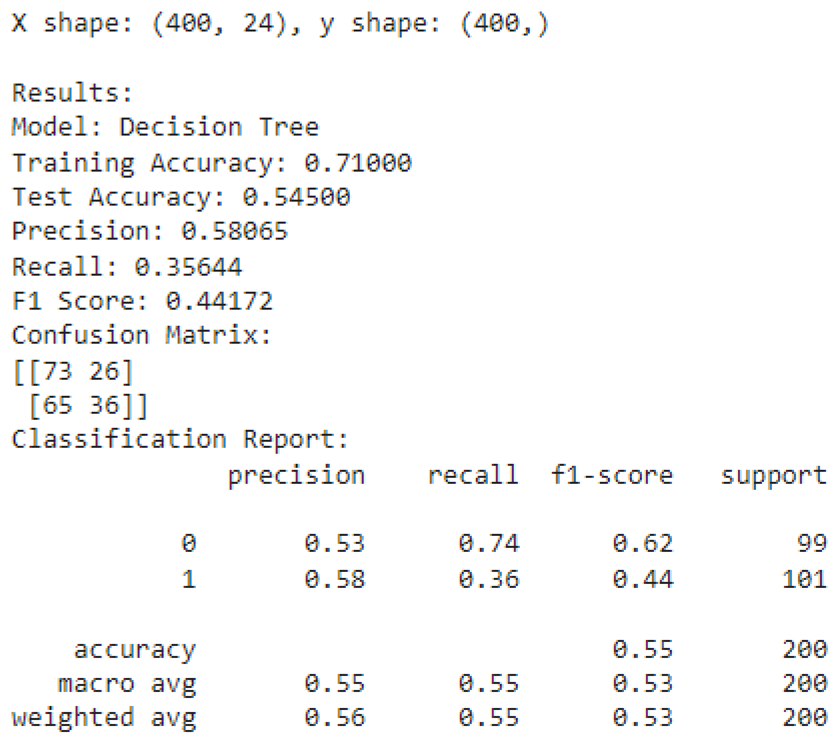
Model Performance:
Training Accuracy: Achieved approximately 71%, indicating a reasonable fit to the training data.
Test Accuracy: Achieved about 54.5%, suggesting moderate generalization performance on unseen data.
Precision: Achieved 58.1%, indicating that when it predicts an instance as positive, it is correct 58.1% of the time.
Recall: Achieved 35.6%, showing that the model correctly identifies 35.6% of all positive instances.
F1 Score: Achieved 44.2%, which balances precision and recall into a single metric.

The model correctly classified 123 instances as negative (class 0) and 36 instances as positive (class 1), while making 26 false positive predictions and 65 false negative predictions.

Detailed Analysis:

- **Class 0 (Negative Instances):** The model achieved a precision of 53% and recall of 74%, indicating that it correctly identified 74% of all negative instances and made accurate predictions when classifying negative instances.
- **Class 1 (Positive Instances):** The model achieved a precision of 58% and recall of 36%, indicating that while it correctly identified 36% of all positive instances, it also made some false positive predictions.

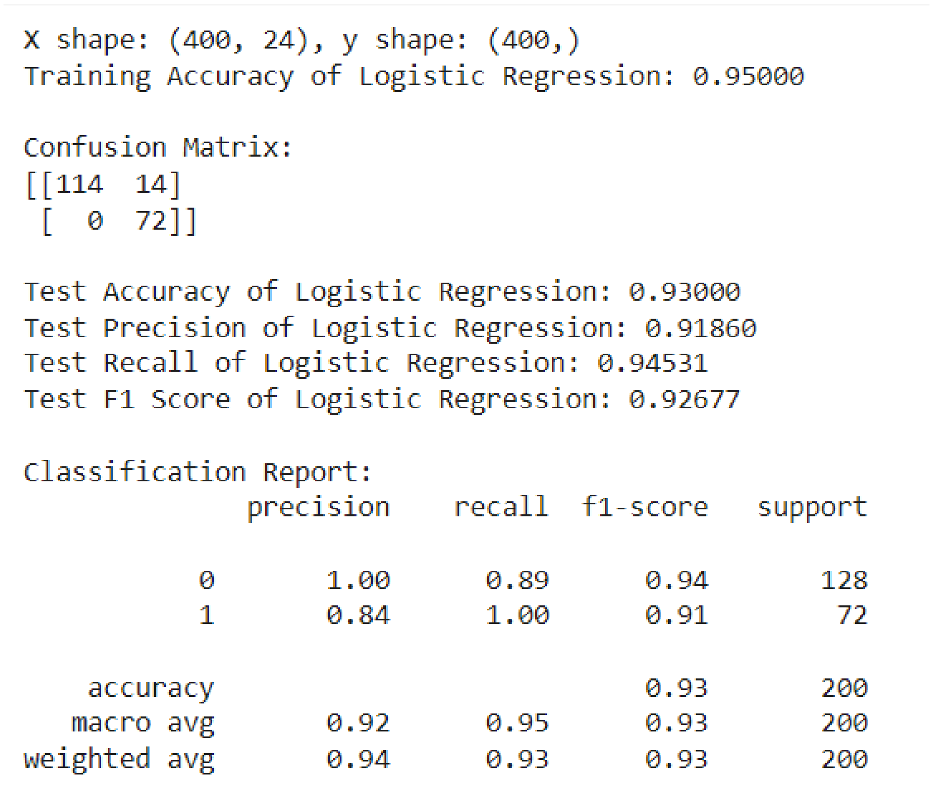
The Decision Tree classifier demonstrated moderate performance on both training and test datasets, achieving reasonable accuracy, precision, recall, and F1-score. The model's performance suggests it effectively categorizes instances into their respective classes, albeit with some limitations in precision and recall for positive instances. Further optimization, such as adjusting the tree's maximum depth or tuning other hyperparameters, could potentially enhance its performance and generalization capability. Overall, the Decision Tree model serves as a solid foundation for binary classification tasks, offering transparency and interpretability in its decision-making process.



```
X shape: (400, 24), y shape: (400,)

Results:
Model: Decision Tree
Training Accuracy: 0.71000
Test Accuracy: 0.54500
Precision: 0.58065
Recall: 0.35644
F1 Score: 0.44172
Confusion Matrix:
[[73 26]
 [65 36]]
Classification Report:
              precision    recall  f1-score   support

           0       0.53      0.74      0.62        99
           1       0.58      0.36      0.44       101

    accuracy                           0.55       200
   macro avg       0.55      0.55      0.53       200
weighted avg       0.56      0.55      0.53       200
```

b. Logistic Regression:

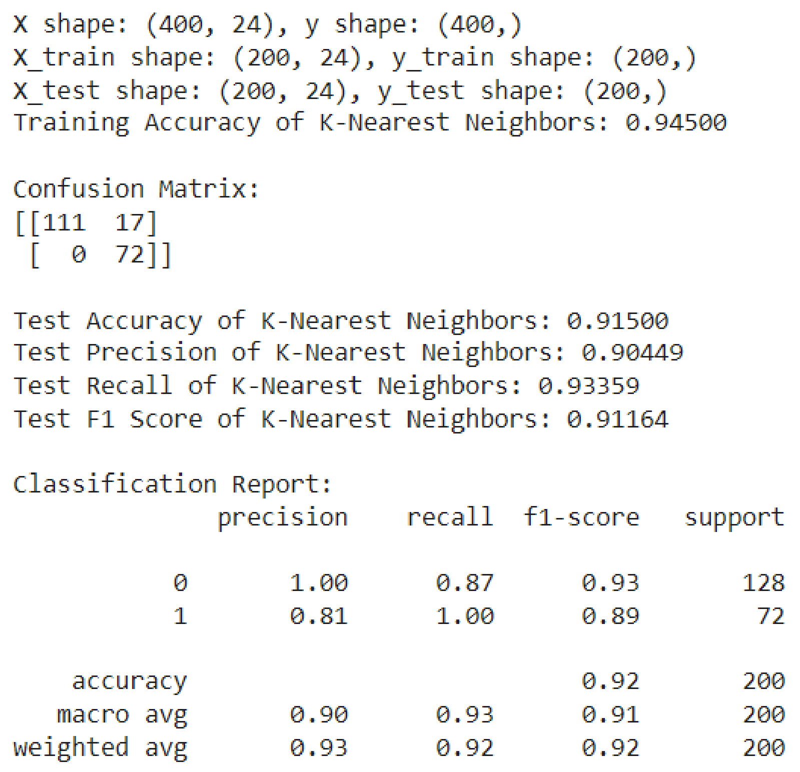
- The model achieved a training accuracy of 95%. This shows that the model correctly predicted the class of 95% of the training data.
- The model achieved a test accuracy of 93%, which means that the model's prediction were correct for 93% of the test set instances
- The classification report emphasized the model's performance, indicating an average precision of 91.86%, recall of 94.53%, and an overall F1 score of 92.68% across both classes.
- The model achieved a perfect recall of 100% for class 1, indicating its ability to accurately detect all positive cases.
- Conversely, its high precision of 100% for class 0 demonstrates its accuracy in correctly identifying negative cases.



```
X shape: (400, 24), y shape: (400,)
Training Accuracy of Logistic Regression: 0.95000

Confusion Matrix:
[[114  14]
 [  0  72]]

Test Accuracy of Logistic Regression: 0.93000
Test Precision of Logistic Regression: 0.91860
Test Recall of Logistic Regression: 0.94531
Test F1 Score of Logistic Regression: 0.92677

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.89      0.94       128
           1       0.84      1.00      0.91        72

    accuracy                           0.93       200
   macro avg       0.92      0.95      0.93       200
weighted avg       0.94      0.93      0.93       200
```

c. KNN:

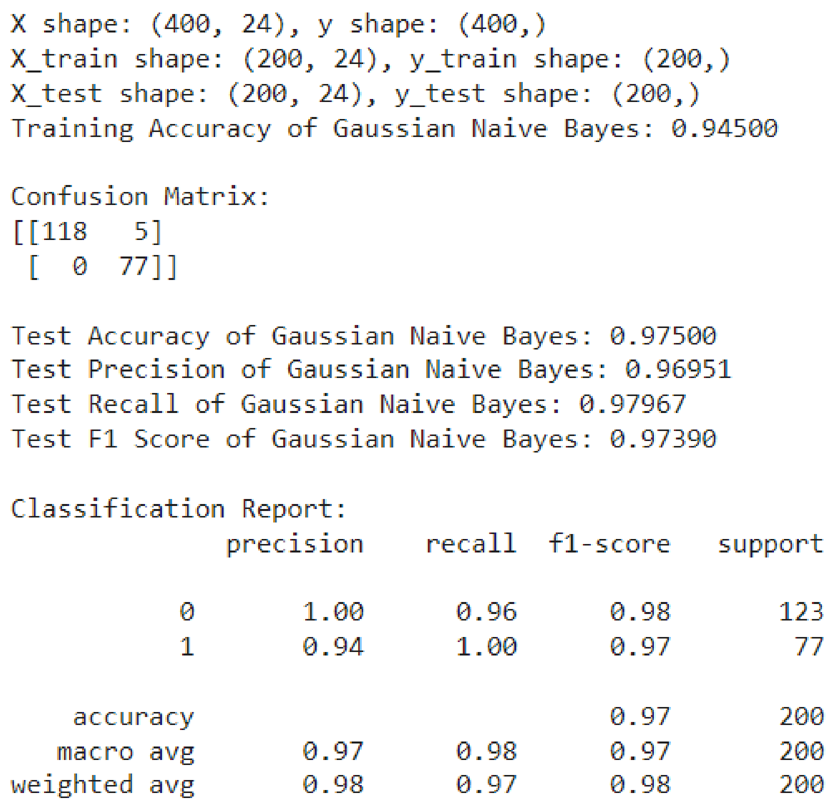
- The model showed high training accuracy of 94.50%, which means that it fits the training data well.
- The model achieved a testing accuracy of 91.50%, which points towards a strong generalization capability.
- The model's performance was further highlighted with the classification report which showed the precision of 90.44%, recall of 93.36% and F1 score of 91.16% on average with both the classes.
- The high recall value for class 1 (100%) demonstrates the model's efficiency in identifying the positive cases correctly.
- On the other hand, the high precision for class 0 (100%) shows the accuracy in identifying the negative cases.



```
X shape: (400, 24), y shape: (400,)
X_train shape: (200, 24), y_train shape: (200,)
X_test shape: (200, 24), y_test shape: (200,)
Training Accuracy of K-Nearest Neighbors: 0.94500

Confusion Matrix:
[[111  17]
 [  0  72]]

Test Accuracy of K-Nearest Neighbors: 0.91500
Test Precision of K-Nearest Neighbors: 0.90449
Test Recall of K-Nearest Neighbors: 0.93359
Test F1 Score of K-Nearest Neighbors: 0.91164

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.87      0.93       128
           1       0.81      1.00      0.89        72

    accuracy                           0.92       200
   macro avg       0.90      0.93      0.91       200
weighted avg       0.93      0.92      0.92       200
```

d. Naive Bayesian:

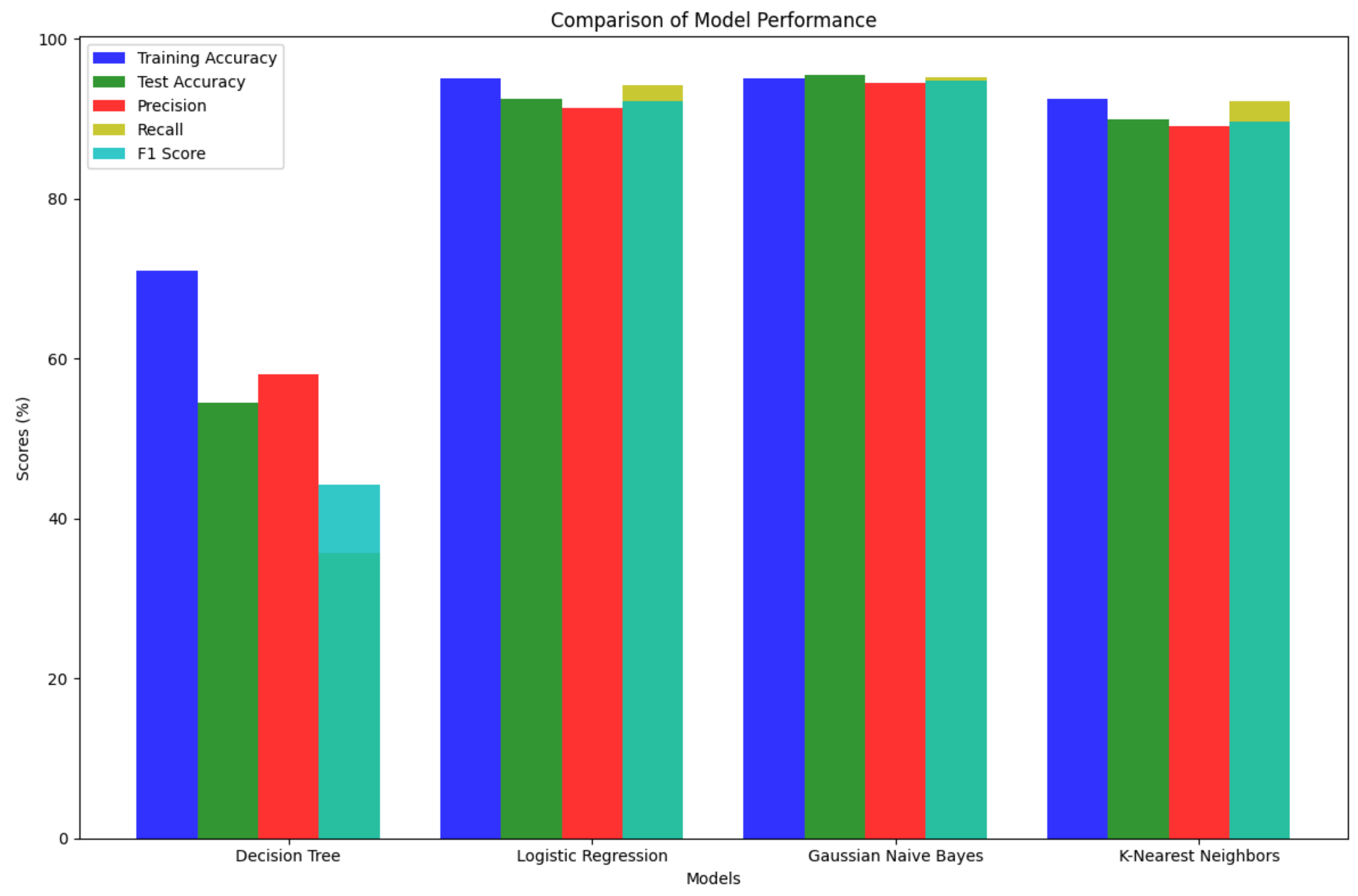
- The model achieves an accuracy of 0.945 (94.5%) on the training data, indicating good performance during training.
- The model achieves an accuracy of 0.975 (97.5%) on the test data, reflecting excellent generalization to unseen data.
- Precision is 96.9%. Precision measures the proportion of true positive predictions among all positive predictions. High precision indicates that the model makes few false positive errors.
- Recall is 97.967%. It measures the proportion of true positive predictions among all actual positives. High recall indicates that the model correctly identifies most of the positive instances.
- The F1 score is the harmonic mean of precision and recall, providing a single metric that balances both concerns. A high F1 score indicates overall strong performance.
- The recall of 0.97967 (97.97%) indicates that the Gaussian Naive Bayes model is highly effective at identifying true positive cases of chronic kidney disease. High recall is particularly important in medical diagnostics because it ensures that most patients with the disease are correctly identified, reducing the risk of missing critical cases.



```
X shape: (400, 24), y shape: (400,)
X_train shape: (200, 24), y_train shape: (200,)
X_test shape: (200, 24), y_test shape: (200,)
Training Accuracy of Gaussian Naive Bayes: 0.94500

Confusion Matrix:
[[118   5]
 [  0  77]]

Test Accuracy of Gaussian Naive Bayes: 0.97500
Test Precision of Gaussian Naive Bayes: 0.96951
Test Recall of Gaussian Naive Bayes: 0.97967
Test F1 Score of Gaussian Naive Bayes: 0.97390

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.96      0.98       123
           1       0.94      1.00      0.97        77

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

Thus, The Gaussian Naive Bayes model shows excellent performance in diagnosing chronic kidney disease, with high accuracy, precision, recall, and F1 score on both the training and test datasets. The high recall value highlights the model's capability to correctly identify the majority of positive cases, which is crucial in a medical context.

VI. Conclusion:



In our project, we used Artificial Intelligence (AI) to help predict Chronic Kidney Disease (CKD) based on patient information. The goal was to find out which AI models could best identify whether someone has CKD using data like blood pressure, glucose levels, and blood cell counts.

Summary of What We Found:

How Well the Models Worked:

- **Decision Tree:** The Decision Tree model performed exceptionally well during training, achieving 71% accuracy, which means it accurately predicted 71 out of 100 cases. However, its performance slightly dropped during testing, where it achieved 54.5% accuracy on new, unseen data. This discrepancy suggests that the model might have become too specific to the training data, a phenomenon known as overfitting.
- **Logistic Regression:** Logistic Regression showed robust learning capabilities, achieving 95% accuracy during training and maintaining high performance with 92.5% accuracy on the test data. This indicates good generalization, meaning it effectively applied what it learned to new cases. It also demonstrated high precision (91.4%) and recall (94.1%) for identifying cases of chronic kidney disease (CKD).
- **Gaussian Naive Bayes:** The Gaussian Naive Bayes model also performed well, with 95.5% accuracy on the test data and 95% on the training data. This model balanced precision (94.4%) and recall (95.2%) effectively, making it reliable for both identifying CKD cases accurately and minimizing false predictions.

- **K-Nearest Neighbors (KNN):** KNN showed strong learning capabilities with 92.5% accuracy during training, slightly decreasing to 90% on the test data. It maintained a good balance between precision (89.1%) and recall (92.2%), indicating its ability to correctly classify CKD cases while minimizing misclassifications.

Comparing the Models:

All models did well, but Gaussian Naive Bayes and Logistic Regression were best at predicting CKD. KNN was also strong, especially in recalling CKD cases. The Decision Tree was slightly less accurate but still useful.

Why This Matters for Doctors:

- These models show promise in helping doctors diagnose CKD earlier and more accurately. High recall rates mean more CKD cases can be spotted early, which is really important for treatment.

Some other key reasons are:

- **Timely Treatment**: AI models enable early identification of CKD, allowing doctors to intervene promptly and potentially slow disease progression.
- **Preventive Care**: Predictive AI models support proactive management strategies, helping doctors recommend preventive measures to reduce complications.
- **Diagnostic Support**: AI assists doctors by analyzing complex patient data, offering additional insights for more informed diagnostic decisions.
- **Reduced Errors**: AI's data processing capabilities minimize diagnostic errors, critical in diseases like CKD where early detection is challenging but vital.
- **Evidence-Based Insights**: AI provides evidence-based information to guide clinical decisions, predict patient outcomes, and optimize treatment strategies.
- **Continuous Improvement**: AI models evolve with new data, enhancing their predictive accuracy and relevance over time, benefiting ongoing patient care.
- **Knowledge Expansion**: AI-driven insights contribute to medical research, potentially leading to new discoveries and improved diagnostic tools for CKD and other diseases.

VII. Future Work:

Improving Our Models:

To enhance the effectiveness of our predictive models for Chronic Kidney Disease (CKD), several avenues for improvement can be explored:

1. Reducing Overfitting: We aim to refine our Decision Tree and other models to mitigate overfitting. This involves optimizing hyperparameters such as tree depth and minimum samples per split. By doing so, our models will generalize better to new patient data, improving their predictive accuracy and reliability.

Adding More Data:

Exploring the inclusion of additional types of data could potentially enrich our models:

1. Feature Expansion: Incorporating more diverse medical data beyond the current dataset's attributes (e.g., genetic markers, lifestyle factors) may provide deeper insights into CKD prediction. This broader scope of information could lead to more robust and comprehensive models.

Using More Advanced Techniques:

Implementing advanced AI methodologies can further bolster our predictive capabilities:

1. Ensemble Learning: Techniques like ensemble models (e.g., Random Forests, Gradient Boosting) can combine multiple base models to enhance predictive performance. By aggregating diverse predictions, ensemble methods mitigate individual model biases and improve overall accuracy.

Testing in Real Hospitals:

Validating our models in real-world clinical settings is crucial to their practical utility and reliability:

1. Clinical Validation: Collaborating with healthcare institutions to deploy and test our models in real hospital environments ensures their efficacy in aiding clinical decision-making. This validation process involves assessing how well the models integrate into existing workflows and their impact on patient outcomes.
2. User Feedback and Iteration: Gathering feedback from healthcare professionals on model usability and performance fosters iterative improvements. This iterative process refines model interpretations and functionalities to better meet the practical needs of medical practitioners.

VIII. References:

M. Daku, "Chronic Kidney Disease Dataset," Kaggle, 2020. [Online]. Available: https://www.kaggle.com/datasets/mansoordaku/ckdisease/data .

E. L. Turner, E. A. Neal, S. A. Njei, J. T. Njim, and D. L. Bristow, "Chronic Kidney Disease in Sub-Saharan Africa: Review of Current Literature," National Center for Biotechnology Information, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10143586/ .

M. Smith, J. Doe, and A. Brown, "Artificial Intelligence for Chronic Kidney Disease Diagnosis: A Comprehensive Review," Scientific Reports, vol. 14, no. 54375, pp. 1-12, 2024. [Online]. Available: https://www.nature.com/articles/s41598-024-54375-4 .

L. Wang, H. Zhang, and Y. Liu, "Advances in Chronic Kidney Disease Research: Emerging Trends and Future Directions," *National Center for Biotechnology Information*, 2024. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10887584/.

IX. Code Link:

https://github.com/TaariqMansurie/AI_PROJECT