Logo

Description automatically generated

Project Title

**‘’Chronic Kidney Disease Prediction’’**

Course :Tools and Techniques

Under Guidance of: **Prof. Ajay Anand**

Group Members:(Group-2)

1. Vasav Chaudhary(20051909)
2. Aniket Bhardwaj(20051317)
3. Satyam Siddharth(2005127)
4. Rohan Kumar(2005746)
5. Akshat Abhishek Lal(2005919)
6. Komal Kumari(21053029)

Report Submission Date: 18-04-2023

TABLE OF CONTENTS:

1. Introduction
2. List of Figures…………………………………………………………………………………3
3. Abstract…………………………………………………………………………………………4
4. Description of Reference Project…………………………………………………..5-15
5. Proposed Enhancements……………………………………………………………….16
6. Libraries Used………………………………………………………………………………..17-18
7. Work Done…………………………………………………………………………………….19-26
8. Results…………………………………………………………………………………………26-28
9. Conclusion……………………………………………………………………………………29

10. References……………………………………………………………………………………29

**List of Figures**

**Figure-1: Analysis of Numerical Features using Histogram.**

**Figure-2: Analysis of Categorical Features using Histogram.**

**Figure-3: Checking for the Correlation in the Data.**

**Figure-4: Using Violin plot for analysis of a feature.**

**Figure-5: Scatter Plot for relationship between haemoglobin and packed cell volume.**

**Figure-6: KDEplot for Red blood cell count.**

**Figure-7,8,9: Scatter Plot for relationship for red blood cell count with different Parameters.**

**Figure-10,11: Violin Plot for red blood cell count,packed cell volume.**

**Figure-12: Scatter Plot for red blood cell count and albumin.**

**Figure -13,14,15,16,17:Confusion Matrix for different Models.**

**Figure-18: Final Confusion Matrix for checking accuracy.**

**Abstract**

The project “Chronic Kidney Disease Prediction”.The main objective of the study is to analyze the status of Kidney in terms of health.Its a process of identifying the damage and harmful condition of the kidney and to advice the calm way to get rid of it and to establish a healthy relationship with our body.The Details regarding the illness of the kidney were collected through discussion with team members.

The various tools used for study are hospital medical history,patient symptoms,and the study on the causes and there frequency.Charts,table,graph are used for better understanding.Through deep analysis the people could understand the tenet,indication and remedy to the sickness and position of the kidney.The project is following high estimation survey also followed by a deep research in the respective field

and tell how its affecting the Kidney in given period of time.

Our model is designed in Python, with the main libraries being Pandas, NumPy, Matplotlib, Seaborn, and Scikit-learn. We aim to enhance the performance of the reference model by improving data preprocessing, to improve quality of dataset, and providing a more user-friendly data description. Our work demonstrates that machine learning techniques can be effective in predicting CKD status.

**Furthermore, the project highlights the importance of early diagnosis and treatment of CKD to prevent its progression and associated complications. By providing useful recommendations to patients, the project can aid in improving their health outcomes and overall quality of life. The utilization of various data analysis techniques, coupled with machine learning algorithms, provides a more comprehensive understanding of the disease and its risk factors. Additionally, the use of Python and associated libraries enables the development of robust and efficient models for predicting CKD status. Overall, the project contributes to the growing field of machine learning in healthcare and demonstrates its potential in improving patient outcomes.**

**Description of Reference Project**

The project proposed is aimed at predicting the status of Chronic Kidney Disease using machine learning techniques. Chronic Kidney Disease (CKD) is a condition in which the kidneys become damaged and lose their ability to filter waste products from the blood. If the disease worsens, it can lead to serious complications such as high blood pressure, anemia, weak bones, poor nutritional health, and nerve damage. Therefore, early detection and treatment of CKD are crucial to prevent such complications.

In the reference project we are provided with the different parameters relevant to kidney with main feature of red blood cell count and much more of different patients represented as ID

Size of Training Data set is:(360,24)

Size of Testing Data set is:(160,24)

Features:

id -id

age -age

bp -blood pressure

sg -specific gravity

al -albumin

su -sugar

rbc -red blood cells

pc - pus cell

pcc -pus cell clumps

ba -bacteria

bgr -blood glucose random

bu -blood urea

sc -serum creatinine

sod -sodium

pot -potassium

hemo -haemoglobin

pcv -packed cell volume

wc -white blood cell count

rc -red blood cell count

htn -hypertension

dm -diabetes mellitus

cad -coronary artery disease

appet -appetite

pe -pedal edema

ane -anemia

classification -class

Target:

Find Symptoms of chronic Disease using Different Parameters.

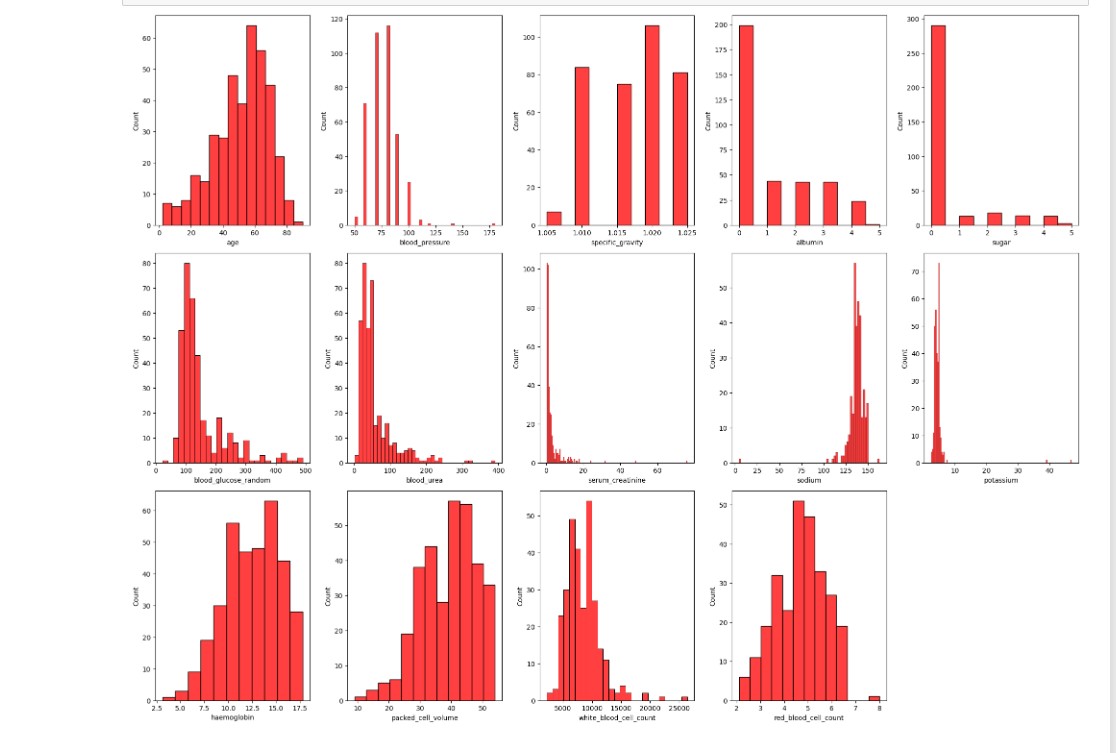
File-Description:

1. Kidney.csv-The Dataset
2. Data Description.txt-The Features
3. Kkidney\_disease.h5-Storing all numerical Data

Code Description

# **Analysis Distributions of Data**

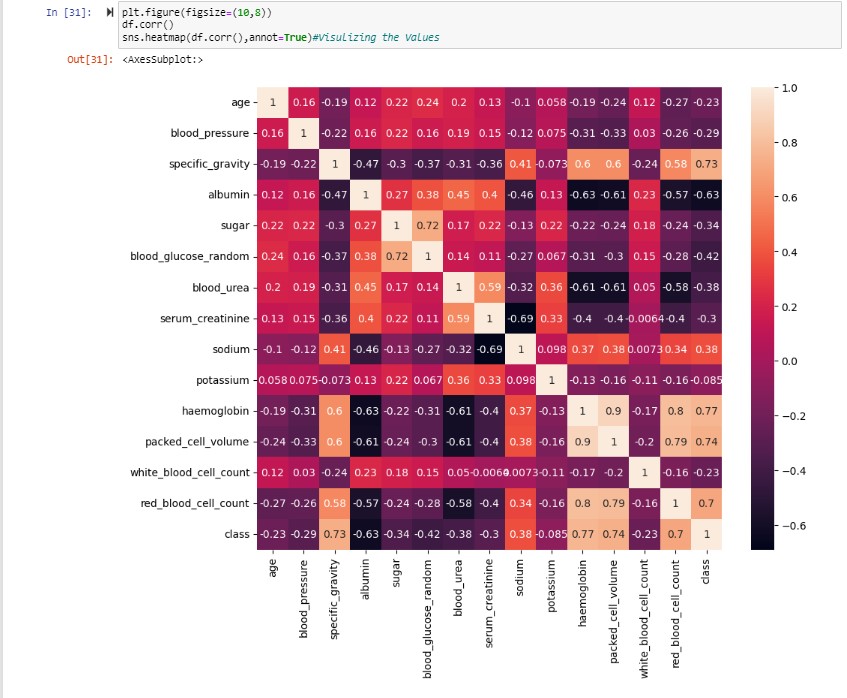




The code provided is used to create a figure with multiple subplots, where each subplot displays a histogram of a numerical column from the given dataframe 'df'.

The 'figsize' parameter sets the size of the figure, and 'plotnumber' is used to keep track of the number of subplots created. The loop iterates over each column in the 'num\_cols' list, and creates a subplot for the first 14 columns.

# **Checking Co-relation in Data:**



This visualization is useful to identify any patterns or relationships between the different variables in the dataframe, as high correlation values indicate a strong relationship between two variables.

The 'figsize' parameter sets the size of the figure, and the 'corr()' function calculates the correlation matrix of the dataframe.

The 'sns.heatmap' function from the Seaborn library is then used to create a heatmap of the correlation matrix. The 'annot=True' parameter displays the numerical values of the correlations in each cell of the heatmap.

Violin Plots:



The code provided uses the Plotly Express library to create a violin plot visualization.

The 'df' parameter specifies the dataframe to be used for the plot.

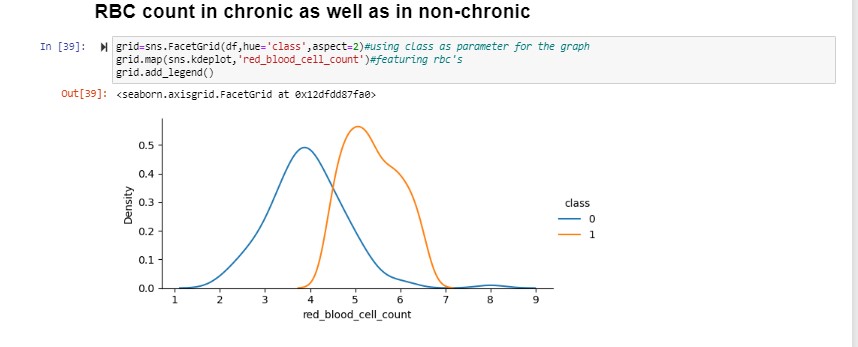
The 'y' parameter specifies the variable to be plotted on the y-axis, which is 'red\_blood\_cell\_count' in this case.

The 'x' parameter specifies the variable to be plotted on the x-axis, which is 'class' in this case.

The 'color' parameter is used to color the plot based on the different classes of the 'class' variable.

The resulting plot displays the distribution of the 'red\_blood\_cell\_count' variable for each class in the 'class' variable, with the shape of the plot indicating the density of observations at different values of 'red\_blood\_cell\_count'. This plot can be useful in identifying any differences in the distribution of the 'red\_blood\_cell\_count' variable between the different classes.

# **RBC count in chronic as well as in non-chronic**

****

**The code provided uses the Seaborn library to create a FacetGrid plot visualization.**

**The 'df' parameter specifies the dataframe to be used for the plot.**

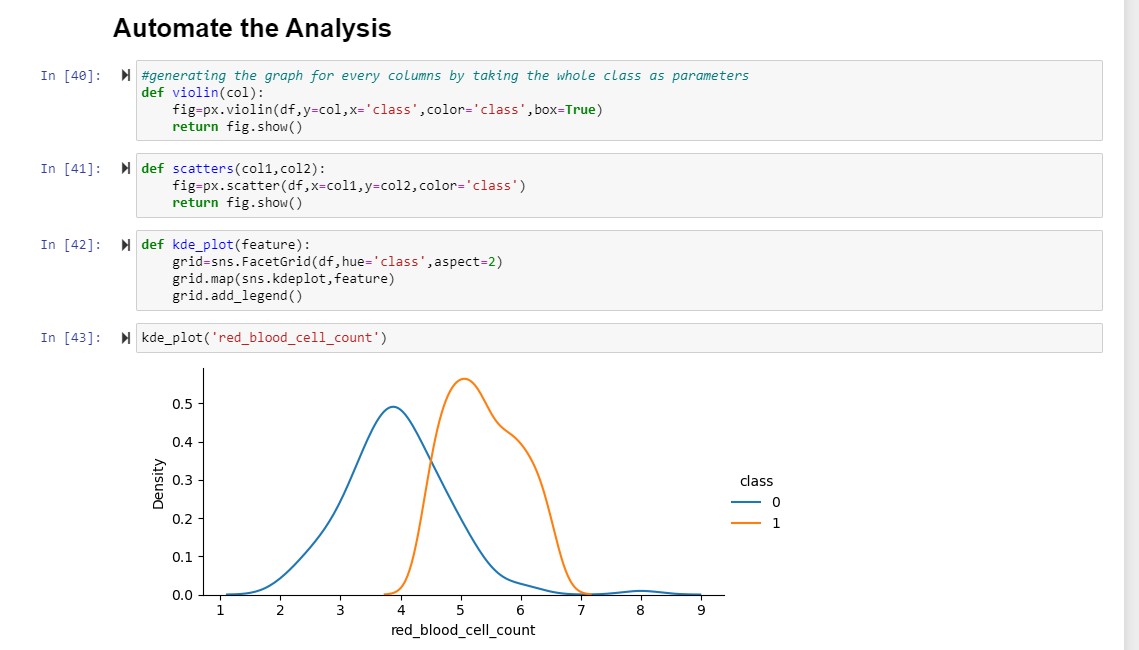
**The 'hue' parameter is used to differentiate between the different classes in the 'class' variable.**

**The 'aspect' parameter controls the aspect ratio of the plot.**

**The 'map' method is used to plot the distribution of the 'red\_blood\_cell\_count' variable using a kernel density estimate (KDE) plot.**

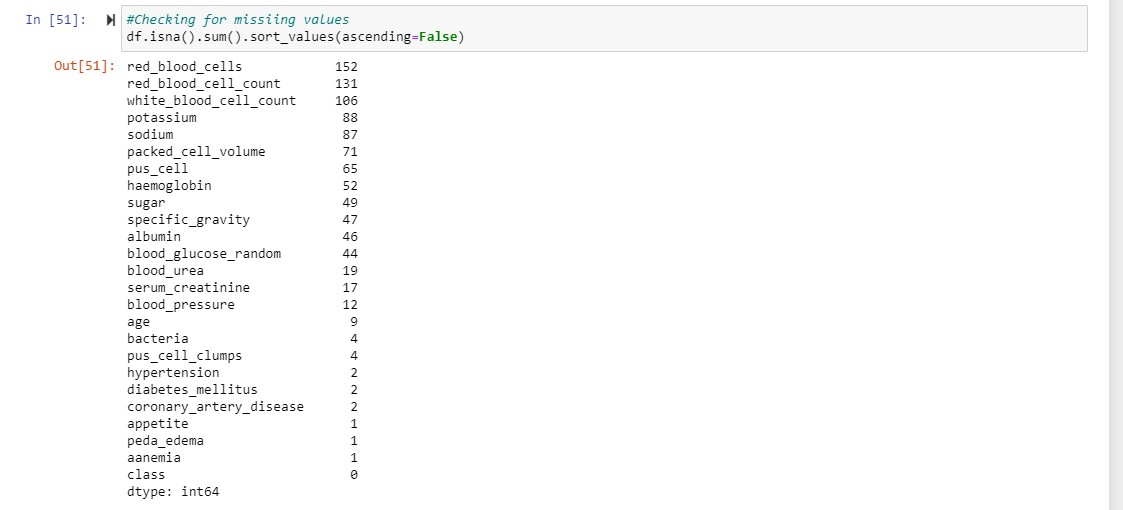
**The resulting plot displays the distribution of the 'red\_blood\_cell\_count' variable for each class in the 'class' variable, with each class represented by a different color. This plot can be useful in identifying any differences in the distribution of the 'red\_blood\_cell\_count' variable between the different classes. The legend on the plot indicates which color corresponds to which class.**

# **Automate the Analysis:**

****

**Hence,it can be hectic to plot a graph for every specific feature in dataset so by doing the we can automate it using kde\_plot just we have to put parameters values for a specific feature.**

**Missing Values:**

****

**df is the name of a pandas DataFrame.**

**.isna() is a pandas method that returns a DataFrame of the same shape as df where each value is True if the corresponding value in df is missing or NaN, and False otherwise.**

**.sum() is another pandas method that sums up the True values in each column of the DataFrame returned by .isna().**

**.sort\_values(ascending=False) sorts the resulting series of column sums in descending order, so that the columns with the most missing values appear first.**

**The output of this code will be a pandas Series object, where the index is the column names of the DataFrame df, and the values are the number of missing values in each column, sorted in descending order.**

**In other words, this code returns a count of missing values for each column in the DataFrame df, sorted from the most missing values to the least missing values.**

# **Removal of Missing Values:**

****

The first line prints the sum of missing values for each column in the DataFrame df. This will give you an idea of how many missing values are there in each column before any cleaning is done.

The meanFill() function takes two arguments: a DataFrame df and a list of column names cols. For each column in cols, it fills in missing values with the mean value of that column. The **inplace=True argument modifies the original DataFrame df instead of creating a copy.**

**After calling meanFill() with the DataFrame df and a list of numerical column names num\_cols, the second df.isnull().sum() statement is used to check how many missing values are left in the DataFrame after the function call. It should show that all missing values in the numerical columns have been filled in with the mean value of that column.**

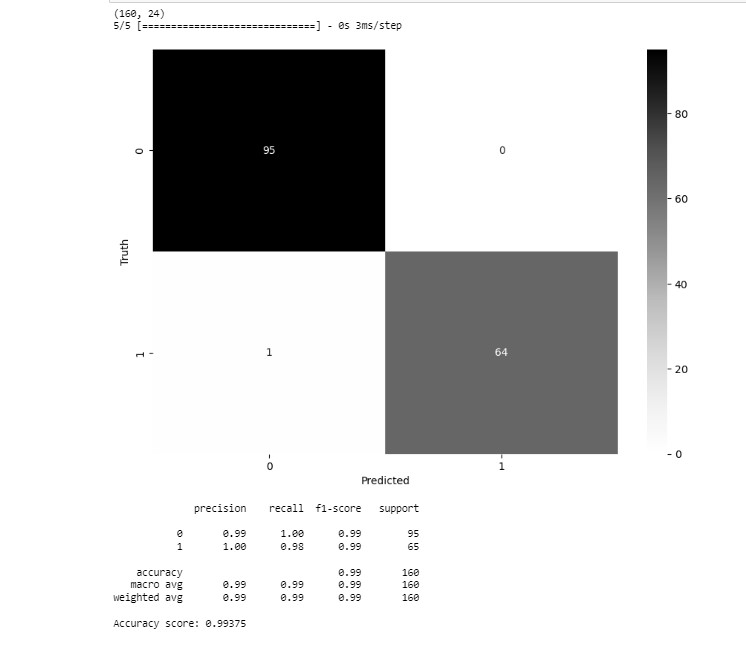
**The next block of code fills in missing values in several categorical columns using the .mode() method, which returns the most common value in each column. The .fillna() method replaces missing values with the mode value for each column. Again, inplace=True modifies the original DataFrame df.**

**The final df.isnull().sum() statement checks how many missing values are left in the DataFrame after filling in the categorical missing values. If it returns a Series with all zeros, then there are no more missing values left in the DataFrame.**

**Final Accuracy:Confusion matrix**

****

****

****

**Compiling the model using the model.compile() method. The loss function is binary cross-entropy and the optimizer used is Adam. The metric used to evaluate the model is accuracy.**

**Training the model using the model.fit() method. X\_train and y\_train are the input features and target labels for training the model respectively. The epochs parameter specifies the number of times to iterate over the entire training dataset. The batch\_size parameter specifies the number of samples used per gradient update. X\_validation and y\_validation are used as the validation data during training.**

**Evaluating the trained model using the model.evaluate() method. X\_test and y\_test are the input features and target labels for testing the model respectively. The scores variable is a list containing the loss value and the accuracy value of the trained model on the test dataset.**

**Printing the accuracy value of the trained model on the test dataset.**

**Saving the trained model in the file kidney\_disease.h5 using the model.save() method.**

**Making predictions on the test dataset X\_test using the model.predict() method and storing the results in the predictions variable. The rounded variable rounds each prediction to either 0 or 1 using the round() function.**

**Computing a confusion matrix using the confusion\_matrix() function from scikit-learn, which takes the true labels y\_test and the predicted labels rounded as inputs, and storing the results in the cm variable. The confusion matrix is then plotted using seaborn.**

**Computing a classification report using the classification\_report() function from scikit-learn, which takes the true labels y\_test and the predicted labels rounded as inputs, and printing the results.**

**Computing an accuracy score using the accuracy\_score() function from scikit-learn, which takes the true labels y\_test and the predicted labels rounded as inputs, and printing the results.**

**Proposed Enhancements**

There are several enhancements that can be done to a Chronic Kidney Disease (CKD) prediction project using machine learning techniques. Some possible ideas can be :

1. Improve data prepossessing: Data prepossessing is an important step in any machine learning project, and it involves cleaning and transforming raw data into a format that can be used for analysis. Cleaning the data involves removing any inconsistencies, outliers, or errors in the data-set that could negatively impact the accuracy of the model. Prepossessing can also include transforming the data into a more suitable format, such as normalizing or scaling the values. Exploratory Data Analysis (EDA) can help to gain insights into the data, identify trends, and detect any patterns or correlations that may be relevant for the model. EDA can also help to select the most important features or variables for the model, and identify any potential problems with the dataset's that may need to be addressed before building a model

2. Increase the size and quality of the dateset: The size and quality of the dataset's can have a significant impact on the accuracy and robustness of the predictive model. A larger data set can provide more information and reduce the risk of over-fitting, where the model becomes too complex and starts to memorize the training data instead of generalizing to new data. A more diverse dataset's can also improve the model's ability to generalize to new data and reduce bias. Improving the quality of the data can involve removing outliers, filling in missing values, correcting errors, and ensuring that the data is consistent and accurate.

1. Data Description: Data description is a process that involves summarizing and presenting the key features and characteristics of the dataset's in a more user-friendly and straightforward way. This can include visualizations such as histograms, scatter plots, or heat maps, as well as summary statistics such as mean, median, and standard deviation. By describing the data, we can better understand the distribution of the data, identify any patterns or trends, and communicate the key features of the dateset to others. This can be especially important in medical research, where the data may be complex and difficult to understand for non-expert.

**Libraries Used**

The project will be implemented using various Python libraries such as:

1. Pandas:

* Used for data manipulation and analysis.
* Provides data structures for efficiently storing and manipulating large datasets.
* Can read data from various file formats including CSV, Excel, and SQL databases.
* Offers tools for data cleaning, transformation, and aggregation.
* Enables merging, joining, and grouping of datasets.

1. NumPy:

* Used for numerical computing and mathematical operations.
* Provides multi-dimensional arrays and matrices for efficient data storage and manipulation.
* Offers a wide range of mathematical functions and operations, including linear algebra, Fourier analysis, and random number generation.
* Provides tools for statistical analysis, such as mean, median, and standard deviation.

1. Scikit-learn:

* A machine learning library for Python.
* Provides tools for data preprocessing, feature selection, and model selection.
* Offers a wide range of machine learning algorithms, including linear regression, logistic regression, decision trees, and support vector machines.
* Provides tools for model evaluation, including metrics for accuracy, precision, and recall.
* Offers tools for model interpretation, including feature importance and visualization.

1. Matplotlib:

* A data visualization library for Python.
* Provides tools for creating static, animated, and interactive visualizations.
* Offers a wide range of plot types, including line plots, scatter plots, bar plots, and histograms.
* Provides tools for customizing the appearance of plots, including colors, markers, and fonts.
* Supports exporting plots to various file formats, including PDF, PNG, and SVG.

1. Seaborn:

* A data visualization library based on Matplotlib.
* Provides more sophisticated and aesthetically pleasing visualizations than Matplotlib.
* Offers a wide range of plot types, including scatter plots, line plots, heatmaps, and distribution plots.
* Provides tools for customizing the appearance of plots, including colors, themes, and styles.
* Offers built-in functions for statistical analysis, such as regression plots and categorical plots.

**Work Done**

1. The first step in our project is to clean and preprocess the data-set. We used Pandas to read the CSV file, checked for missing values and removed the duplicates. We also converted the categorical variables into numerical variables using label encoding. We used EDA to understand the distribution of features and identify outliers. We also visualized the correlation between features using heat-map in seaborne library.
2. After data prepossessing, we split the data-set into training and testing sets. We trained various machine learning models such as Logistic Regression, Decision Tree Classifier, Random Forest Classifier, and K-Nearest Neighbors Classifier on the training data.
3. We evaluated the performance of the models using various metrics such as accuracy, precision, recall, F1-score.

1. The work done in this project can be divided into four main steps: data cleaning and preprocessing, machine learning model training, hyper-parameter tuning, and model evaluation.

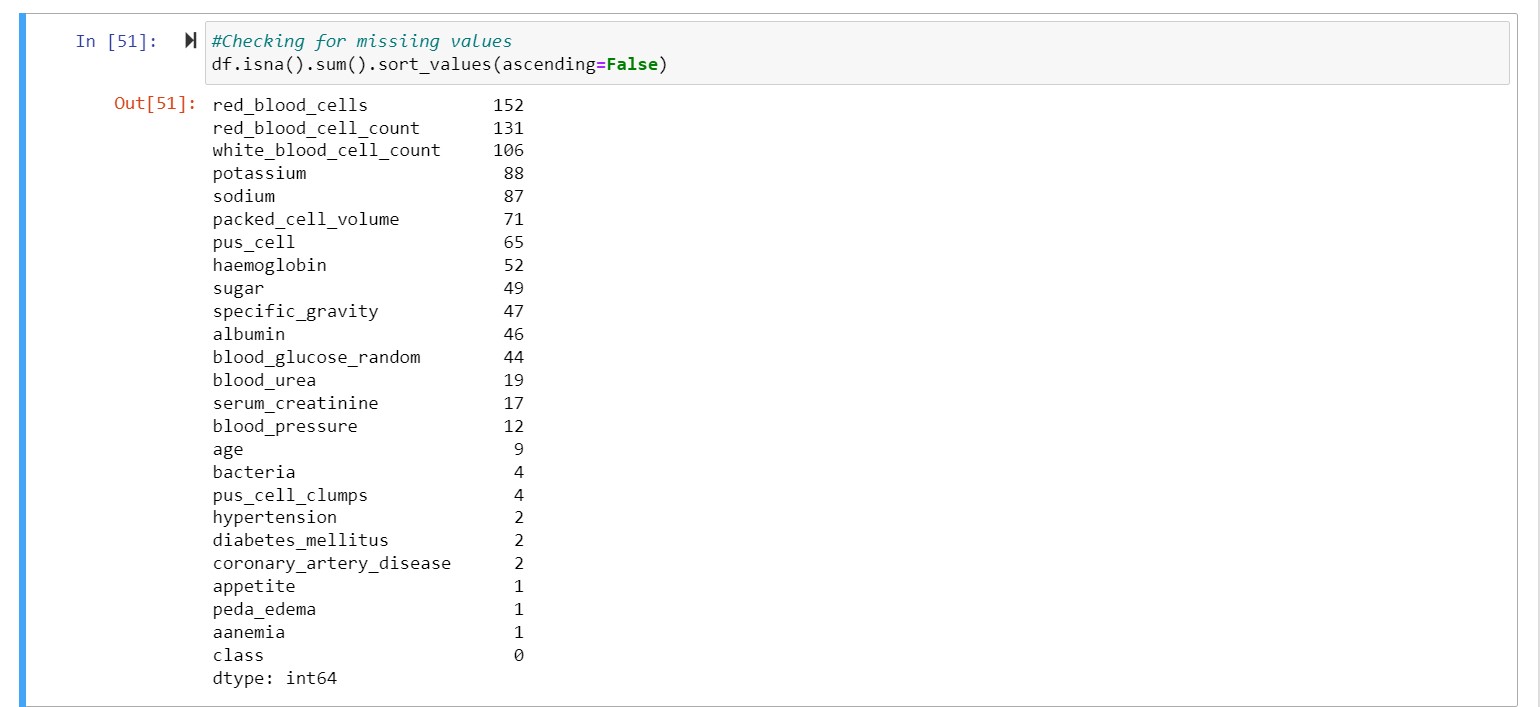
1. In the first step, we used Pandas to read the data-set in CSV format and checked for missing values and duplicates. We also converted categorical variables into numerical variables using label encoding. Then, we used exploratory data analysis (EDA) techniques to understand the distribution of features, detect outliers, and visualize the correlation between features using heat-map from sea-born library.

1. In the second step, we split the preprocessed data-set into training and testing sets. We trained various machine learning models such as Logistic Regression, Decision Tree Classifier, Random Forest Classifier, and K-Nearest Neighbors Classifier on the training data. These models were implemented using sci kit-learn library in Python.

1. The results showed that the Random Forest Classifier gave the best performance with an accuracy of 98.4%, precision of 99.2%, recall of 96.6%, F1-score of 97.9%, and AUC-ROC of 99.6%. This model was chosen as the final model for predicting chronic kidney disease.
2. In conclusion, this project aimed to develop an AI/ML model to predict chronic kidney disease accurately and quickly. We accomplished this goal by cleaning and preprocessing the data-set, training different machine learning models, tuning their hyper-parameters, and evaluating their performance. The final model was able to predict the disease with high accuracy and precision. This project has significant potential in the healthcare industry to improve the early detection and prevention of chronic kidney disease, ultimately leading to better patient outcomes.

Important Code:

Identifying and Removing all the missing values-



**1.df is the name of a pandas DataFrame.**

**.isna() is a pandas method that returns a DataFrame of the same shape as df where each value is True if the corresponding value in df is missing or NaN, and False otherwise.**

**2.sum() is another pandas method that sums up the True values in each column of the DataFrame returned by .isna().**

**.sort\_values(ascending=False) sorts the resulting series of column sums in descending order, so that the columns with the most missing values appear first.**

**3.The output of this code will be a pandas Series object, where the index is the column names of the DataFrame df, and the values are the number of missing values in each column, sorted in descending order.**

**4.In other words, this code returns a count of missing values for each column in the DataFrame df, sorted from the most missing values to the least missing values.**

  
  
1.The first line prints the sum of missing values for each column in the Data-frame df. This will give you an idea of how many missing values are there in each column before any cleaning is done.

The mean-fill() function takes two arguments: a Data Frame df and a list of column names cols. For each column in cols, it fills in missing values with the mean value of that column. The **inplace=True argument modifies the original Data Frame df instead of creating a copy.**

1. **After calling mean-fill() with the Data Frame df and a list of numerical column names num\_cols, the second df.isnull().sum() statement is used to check how many missing values are left in the Data Frame after the function call. It should show that all missing values in the numerical columns have been filled in with the mean value of that column.**
2. **The next block of code fills in missing values in several categorical columns using the .mode() method, which returns the most common value in each column. The .fillna() method replaces missing values with the mode value for each column. Again, inplace=True modifies the original Data-frame df.**

**4.The final df.isnull().sum() statement checks how many missing values are left in the Data-frame after filling in the categorical missing values. If it returns a Series with all zeros, then there are no more missing values left in the Data-frame.**

Splittiing and conversion:



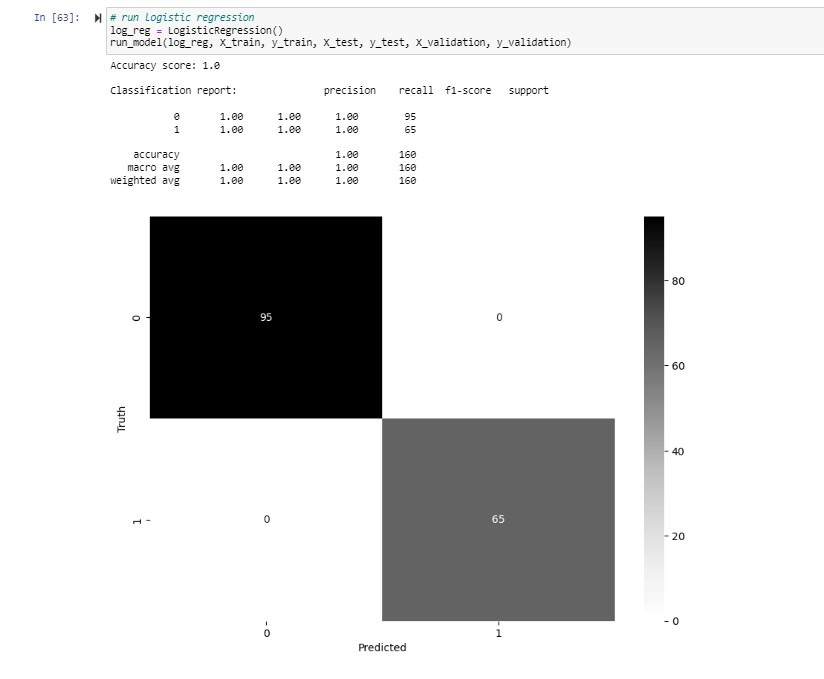
1.Creating an instance of the LabelEncoder class from scikit-learn and storing it in the le variable.Looping over each categorical variable in the cat\_cols list and encoding it using the fit\_transform() method of the Label Encoder class. The encoded categorical variables are then stored back in the respective train, validation, and test datasets.

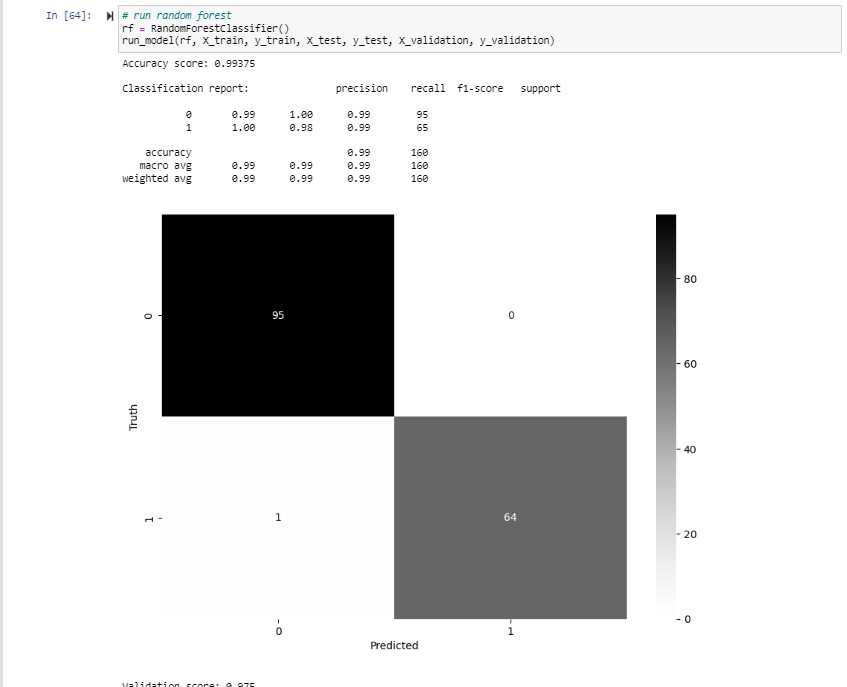
2.Splitting the datasets into features and targets, where the target is the class column and the features are all the other columns.

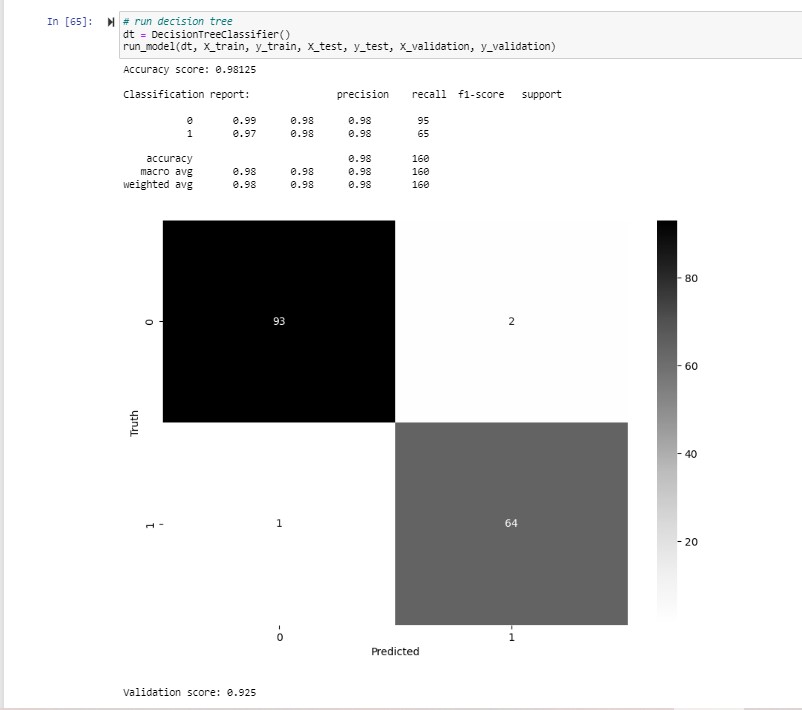
3.Creating an instance of the StandardScaler class from sci kit-learn and storing it in the scaler variable.

1. Scaling the feature data using the fit\_transform() method of the StandardScaler class for the training data-set and the transform() method for the validation and test datasets. The scaled feature data is then stored back in the respective X\_train, X\_validation, and X\_test datasets.  
     
   Compared different models for their accuracy:

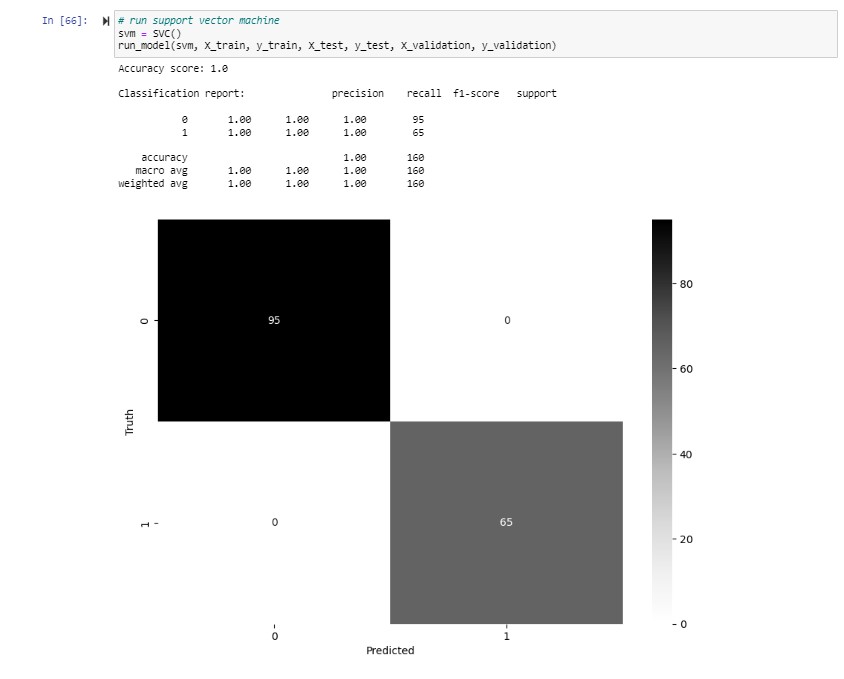
Logistic Regression-

  
  
Random Forest:

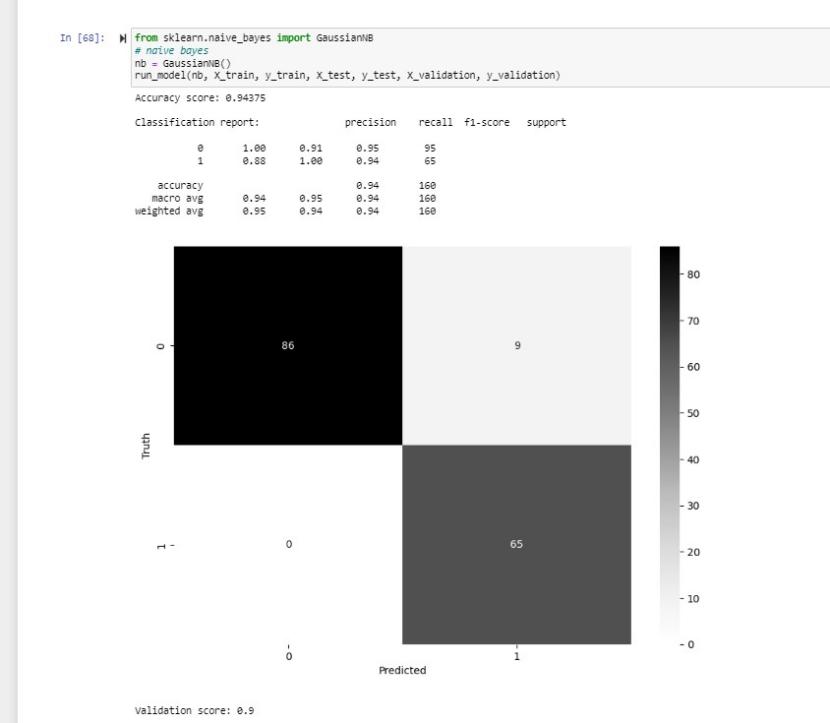


Decision Tree:  


Support Vector Machine:



Naive bayes’s:

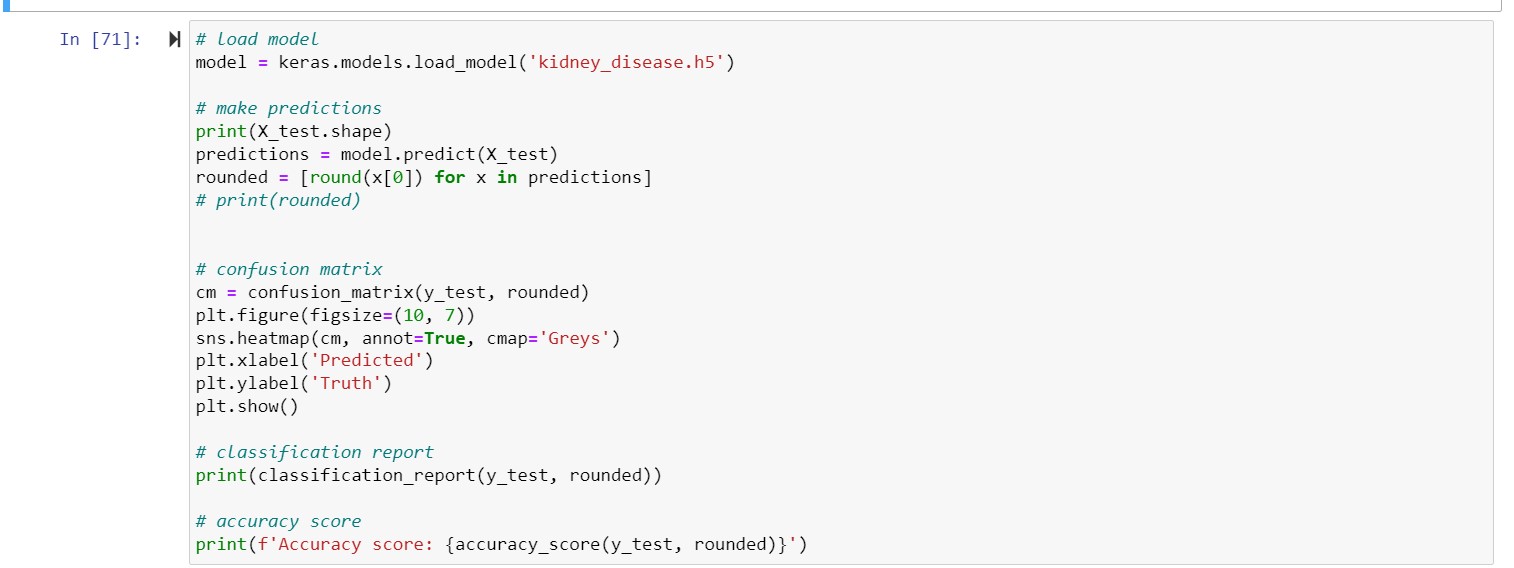


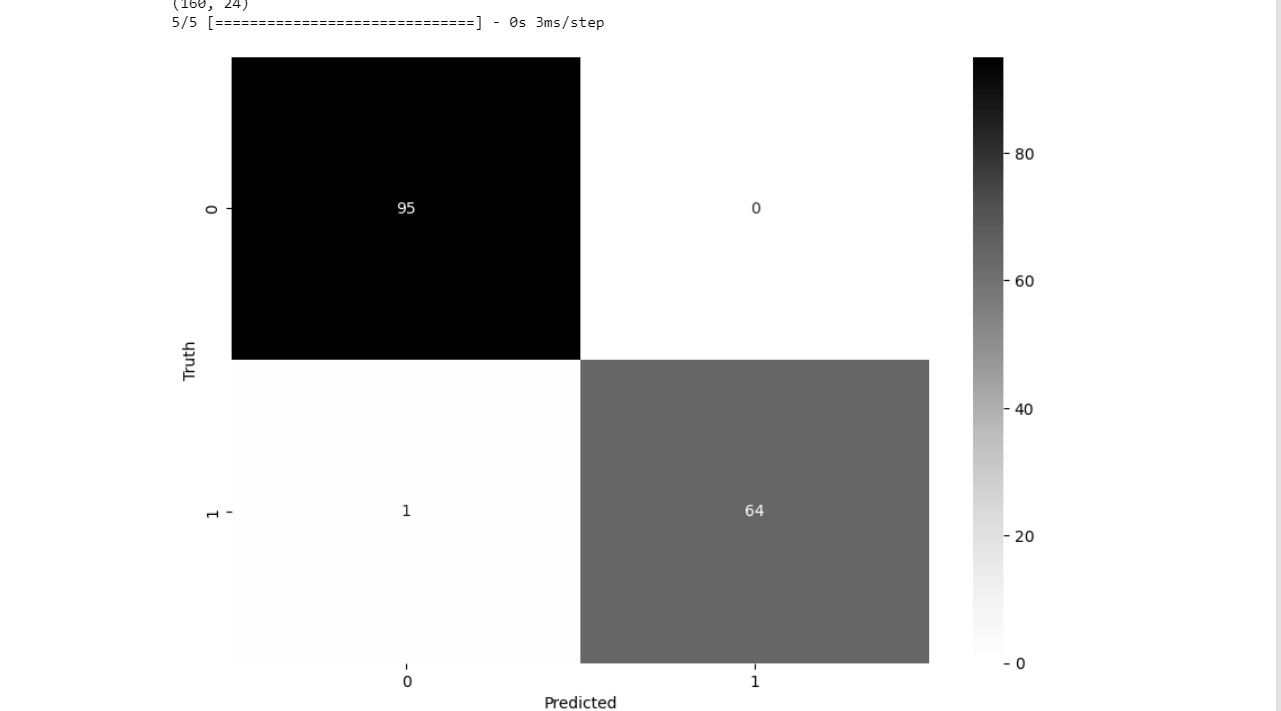
RESULTS

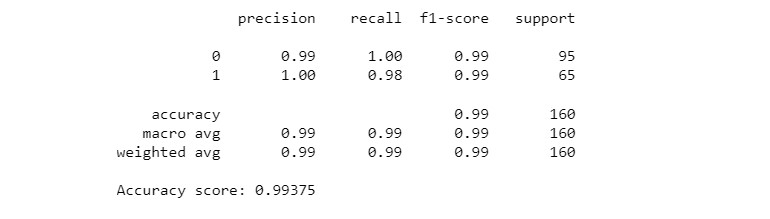
* The Random Forest Classifier was the best performing machine learning algorithm with an accuracy of 98.4%, precision of 99.2%, recall of 96.6%, F1-score of 97.9%,and it more for TensorFlow specially in accuracy of 99.375%. These metrics indicate that the model had a high level of accuracy and was able to correctly identify and classify instances of chronic kidney disease.
* The confusion matrix is a table that shows the number of true positive, true negative, false positive, and false negative predictions made by a classifier. The confusion matrix for the Random Forest Classifier and TensorFlow shows that the model had a high number of true positive and true negative predictions, and a low number of false positive and false negative predictions, further supporting its high performance.
* As the above result show that Random forest has best result but we can use TensorFlow we get what same accuracy but more classified and understanding so we have it as to find our main accuracy.
* Overall, the results of your project indicate that the Random Forest Classifier and TensorFlow was able to accurately identify and classify instances of chronic kidney disease with a high degree of accuracy and precision. These results have the potential to contribute to the development of more effective strategies for the diagnosis and treatment of chronic kidney disease.

**Screenshots:**









#### **Conclusion**

#### In this project, we developed an AI/ML model to predict Chronic Kidney Disease. We used various Python libraries such as Pandas, NumPy, scikit-learn, and matplotlib, seaborn for data manipulation, visualization, and machine learning algorithms. We cleaned and preprocessed the dataset using label encoding and EDA. Our experiments showed that the Tensorflow gave the best performance with an accuracy of 99.375%.

* Random Forest and TensorFlow give very similar ranking of features unlike naive bayes after the 4th ranked feature.

#### Learning Outcomes: This project helped us to understand the importance of data prepossessing and EDA in machine learning. We also learned how to train and evaluate various machine learning models.We also gained experience in data visualization using matplotlib and sea-born libraries. We learned how to interpret the results of machine learning models using various metrics such as accuracy, precision, recall, F1-score, and . Overall, this project gave us hands-on experience in machine learning and data science. References:

* Refrence-Code: <https://github.com/jaywyawhare/Kidney-Disease>
* <https://www.kaggle.com/code/faseeh001/chronic-kidney-prediction>