DISEASE PREDICTION

USING MACHINE LEARNING

Submitted in Partial Fulfilment of the Requirements for the Degree of

Bachelor of Technology in **Information Technology**

Submitted By

Kunal Raj

University Roll no - 10300220051

Gourab Ghosh

University Roll no -10300220041

Aman Kumar

University Roll no -10300220013

Chandra Kant

University Roll no -10300220037

Under the Supervision
Of

Dr. Ramkrishna Ghosh

Associate Professor

Department of Information Technology

Haldia Institute of Technology

ICARE Complex, Hatiberia, Haldia, Dist - Purba Medinipur

West Bengal, India, PIN - 72165

Haldia Institute of Technology

ICARE Complex, Hatiberia, Haldia, Dist - Purba Medinipur

West Bengal, India, PIN – 721657

Certificate of Approval

This is to certify that the report entitled "**Disease Prediction Using Machine Learning"** submitted by **Kunal Raj** (10300220051), **Gourab Ghosh** (10300220041), **Aman Kumar** (10300220013), **Chandra Kant** (10300220037) to the **Haldia Institute of Technology** in partial fulfillment of the **B.Tech. degree in Information Technology** is a bonafide record of the project work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Signature of Head of Department	Signature of Supervisor
Dr. Soumen Paul	**Dr. Ramkrishna Ghosh**
Professor and Head of Department	Associate Professor
Department of IT	**Department of IT**
Haldia Institute of Technology	Haldia Institute of Technology

DECLARATION

We hereby declare that this written submission represents our work in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/ data/ fact/ source in my submission. We understand that any violation of the above will cause disciplinary action by the institute and can also evoke penal action if proper permission has not been taken when needed. This report has not been previously formed the basis for the award of any degree, diploma, or similar title of any other University.

Signatures

Kunal Raj

Gourab Ghosh

Aman Kumar

Chandra Kant

Department of Information Technology

Haldia Institute of Technology

ACKNOWLEDGEMENT

The success and outcome of this project required a lot of guidance and assistance from many people, and we are extremely fortunate to have their support all along the completion of our project work. Whatever we have done is only due to such guidance and assistance, and we would never forget to thank them. We would like to show our greatest appreciation to **Dr. Ramkrishna Ghosh**.

We cannot thank them enough for their tremendous support and help. Without their encouragement and guidance, this project would not have materialized.

Signatures

Kunal Raj
Gourab Ghosh
Aman Kumar
Chandra Kant

Department of Information Technology
Haldia Institute of Technology

ABSTRACT

This project is an attempt to help one to predict the disease he/she is having through the symptoms and the correct readings of the bodily vitals needed. There are times when people keep on ignoring health issues due to high medical fees. This may lead to severe issues later and even death.

If not covered by insurance, medical bills can be a menace. This project is an approach in reducing the effort of a normal person by estimating the kind of disease one has and its severity. We have designed a kidney disease prediction system using multiple machine learning algorithms.

Based on the symptoms, age, and gender of an individual, along with some of the other factors considered, the diagnosis system gives the output giving the information about whether the user is suffering from that disease or not.

According to the severity, some diet plans and some exercises which can minimize the effects of the disease to some extents are also provided. It provides a simple yet effective approach for predicting the disease, if the provided values of vitals are accurate. The user will experience a simple yet effective User Interface and pleasing design.

CONTENTS

INTRODUCTION

Machine Learning model allows us to build models to get quickly cleaned and processed data and deliver results faster. By using this system doctors will make good decisions related to patient diagnoses and according to that, good treatment will be given to the patient, which increases improvement in patient healthcare services. To introduce machine learning in the medical field, healthcare is the prime example. To improve the accuracy of large data, the existing work will be done on unstructured or textual data.

For the prediction of diseases, the existing will be done on linear, KNN, Decision Tree algorithm. Machine Learning is the domain that uses past data for predicting. Machine Learning is the understanding of computer system under which the Machine Learning model learn from data and experience.

The machine learning algorithm has two phases: 1) Training & 2) Testing.

To predict the disease from a patient's symptoms and from the history of the patient, machine learning technology is struggling from past decades.

Healthcare issues can be solved efficiently by using Machine Learning Technology. We are applying complete machine learning concepts to keep the track of patient's health. ML model allows us to build models to get quickly cleaned and processed data and deliver results faster. [6]

By using this system doctors will make good decisions related to patient diagnoses and according to that, good treatment will be given to the patient, which increases improvement in patient healthcare services.

To introduce machine learning in the medical field, healthcare is the prime example. To improve the accuracy of large data, the existing work will be done on unstructured or textual data. For the prediction of diseases, the existing will be done on linear, KNN, Decision Tree algorithm. [5]

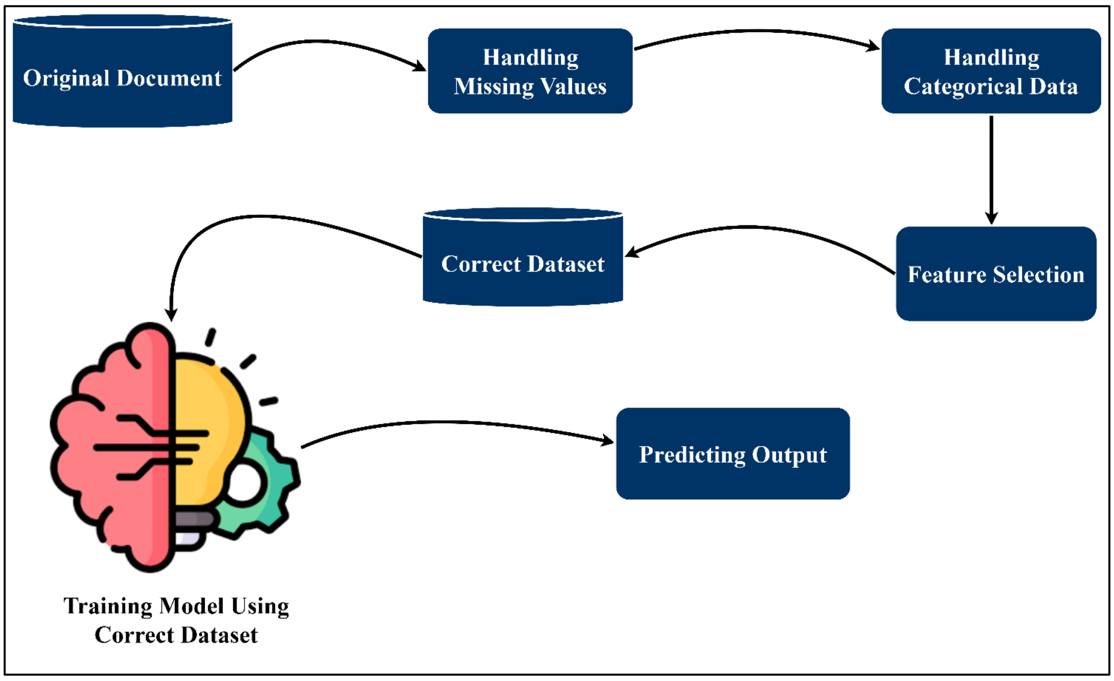


Fig. 1: Block diagram of disease prediction using machine learning (workflow)

Due to an increased amount of data growth in the medical and healthcare field the accurate analysis on medical data which has been benefited from early patient care.

With the help of disease data, data mining finds hidden pattern information in the huge medical data of the data set. [1]

We proposed a disease prediction platform, based on the vitals of the patient. Our disease prediction ML model predicts the occurrence of kidney diseases.

Here are some examples of how useful ML can be in field of medicine -

- **Spotting diseases:** Machine learning can look at X-rays and scans to help find problems early.
- **Medicine magic:** It can help design new drugs and figure out which ones might work best or you.
- **Personalized care:** Like a super-smart assistant, it can suggest the best treatment plan based on your health.

- **Predicting health bumps:** Machine learning can warn doctors if you might get sick, so they can help prevent it.

Machine learning is not meant to replace doctors; rather, it serves as a powerful assistant, augmenting their expertise with data-driven insights and capabilities. [2]

This collaboration between human intelligence and machine learning holds immense promise for the future of healthcare.

By leveraging machine learning's ability to analyze vast amounts of data and identify complex patterns, doctors can gain a deeper understanding of their patients' conditions and make more informed treatment decisions. [2]

Machine learning can also automate routine tasks, freeing up doctors' time to focus on what they do best: building relationships with patients and providing compassionate care. [6]

Ultimately, this synergistic approach between human and machine intelligence has the potential to revolutionize healthcare, making it faster, more precise, and ultimately, more effective. [1]

Along with it, we have also provided an about page which gives information about the symptoms & information about the diseases.

LITERATURE REVIEW

Several studies have been performed on disease prediction using machine learning. The prediction of disease using different ml algorithm help us to find accuracy among ML models.

Bemando et al. investigated the relationship between blood-related diseases and their features utilizing classifier methods such as Gaussian NB, Bernoulli NB, and Random Forest.

These three algorithms anticipate and offer statistical findings in a variety of ways. In this experiment, we discovered that Nave Bayes estimated accuracy was higher than that of other algorithms. [6]

Kumar and Polepaka devised a technique for illness prediction in the medical field. They employed Random Forest and CNN as well as other machine learning methods.

For illness dataset classification, precision, recall, and F1-score, these algorithms deliver better results. In this experiment, Random Forest outperformed other algorithms in terms of accuracy and statistical performance.

On the kidney disease dataset, Nithya et al. developed a method for categorization and cluster-based analysis.

On diverse sets of photos, the authors utilized the K-Means clustering technique to collect the closest familiar images. They calculated 99.61 percent classification accuracy using Artificial Neural Networks for Kidney Disease Image Prediction. [3]

The purpose of the proposed model is to predict whether the patient will suffer or develop chronic kidney disease in the future if he continues their lifestyle.

This information can be used to determine whether the kidney disease is using eGFR (glomerular filtration rate), which helps the doctor plan the appropriate treatment.

Estimated glomerular filtration rate (eGFR) defines the degree of kidney disease and measures kidney function.

A combination of estimated glomerular filtration rate (GFR), age, diet, existing medical conditions, and albuminuria can be used to assess the severity of kidney disease, but requires more accurate information about the risk to the kidney is required to make clinical decisions about diagnosis, treatment, and prognosis. [4]

While the factors mentioned provide a valuable starting point, a more comprehensive evaluation might involve additional tests like kidney ultrasounds, biopsies, or genetic screenings.

These can pinpoint the underlying cause of the kidney damage, assess the extent of scarring, and identify potential complications. [4]

This more detailed picture allows healthcare professionals to tailor treatment plans to address the specific issues at hand, potentially slowing disease progression and improving long-term outcomes for the patient.

PROBLEM DEFINITION

The contemporary need for comprehensive and real-time environmental monitoring has led to the development of an advanced solution aimed at making the environment intelligent and interactive through wireless communication. The proposed model addresses critical challenges in weather monitoring and climate awareness, emphasizing the following key issues:

Early detection & treatment: - Engineers and medical researchers are trying to develop machine learning algorithms and models that can identify chronic kidney disease at an early stage. The problem is that the data generated in the health industry is large and complex, making data analysis difficult.

Reducing the fatality of disease: - Combination of estimated glomerular filtration rate (GFR), age, diet, existing medical conditions, and albuminuria can be used to assess the severity of kidney disease, but requires more accurate information about the risk to the kidney is required to make clinical decisions. [3]

Chronic kidney disease (CKD) is a life-threatening condition that can be difficult to diagnose early because there are no symptoms. The purpose of the proposed study is to develop and validate a predictive model for the prediction of chronic kidney disease.
The main function of the kidney is to filter the blood in the body. Kidney disease is a silent killer because it can cause kidney failure without causing any symptoms or concern. [4]

Chronic kidney disease is defined as a decline in kidney function over a period of months or years. Kidney disease is often caused by diabetes and high blood pressure.

SOFTWARE REQUIREMENTS SPECIFICATION

System configurations:

The software requirement specification can produce at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by established a complete information description, a detailed functional description, a representation of system behavior, and indication of performance and design constrain, appropriate validate criteria, and other information pertinent to requirements.

Software Requirements:

- Google Collab
- Python
- Windows 11

Hardware Requirements:

- RAM: 4GB
- Processor: i3 10th gen or equivalent
- SSD: 256 GB
- Graphics: inbuilt

PROPOSED WORK

Importing Dependencies: -

In machine learning, achieving success hinges on the efficient utilization of various tools and algorithms. These tools, often packaged as libraries or frameworks, serve as the building blocks for data manipulation, model creation, and evaluation. Effectively managing these dependencies is crucial for project reproducibility, maintainability, and collaboration.

Importance of Dependency Management:

1. **Consistency:** Specifying dependencies ensures everyone involved in the project uses the same library versions. This eliminates compatibility issues and unexpected behavior during code execution.
2. **Reproducibility:** A well-defined dependency list allows for replicating the project's results on different machines. This is vital for scientific validation and sharing research findings.
3. **Isolation:** Virtual environments, which isolate project dependencies from the system-wide Python environment, prevent conflicts arising from different project requirements.

Dependency management fosters a streamlined development workflow, facilitates collaboration, and enables the creation of reliable and reproducible machine learning solutions.

Analyzing Features*: -*

Importance of Feature Analysis:

- **Better Predictions:** By selecting features that have a strong correlation with the target variable (what we are trying to predict), the model can learn more accurate patterns. Imagine training a model to predict house prices. Features like square footage and location are likely more important than the colour of the doorknobs.

- **Reduced Complexity:** A high number of features can make a model cumbersome and prone to overfitting, where it memorizes the training data but performs poorly on new data. Feature analysis helps identify and remove irrelevant or redundant features, streamlining the model and improving generalization.
- **Interpretability:** With fewer, well-chosen features, it's easier to understand the inner workings of the model. This is crucial for tasks like fraud detection where you need to know why a certain transaction was flagged as suspicious.

How Feature Analysis is Done:

Several techniques exist for analysing features, broadly categorized into three approaches:

- **Filter Methods:** These rely on statistical analysis to identify features with strong correlations to the target variable. Common techniques include chi-square tests for categorical features and correlation analysis for numerical features.
- **Wrapper Methods:** These involve the machine learning model itself. The algorithm evaluates different combinations of features and selects the set that yields the best performance on a validation dataset. This is a more computationally expensive approach but can be more effective than filter methods.
- **Embedded Methods:** These techniques integrate feature selection into the model training process. A classic example is L1 regularization, which penalizes models with large coefficients, essentially pushing some feature weights towards zero and effectively removing those features from the model.

By employing these techniques, data scientists can transform raw data into a refined set of features that empowers machine learning models to make accurate and interpretable predictions.

Data Cleaning and Preprocessing*:* -

Data preprocessing is the essential first step in machine learning, cleaning raw data for use in models. It improves data quality and model understanding. Common techniques include handling missing values, encoding categorical data, scaling features, and splitting data into training and testing sets. The specific techniques depend on the data and machine learning task.

Steps In Data Preprocessing:

1. Gathering the data*:*

Data is raw information; it is the representation of both human and machine observation of the world. Dataset entirely depends on what type of problem we want to solve. Each problem in machine learning has its own unique approach. Here i am sharing some website with you to get the dataset

Kaggle: Kaggle is one of the best ones to get the dataset. https://www.kaggle.com/datasets

2. Import the dataset & Libraries*:*

First step is usually importing the libraries that will be needed in the program. A library is essentially a collection of modules that can be called and used.

3. Dealing with Missing Values*:*

Sometimes we may find some data are missing in the dataset. if we found then we will remove those rows or we can calculate either **mean, mode or median** of the feature and replace it with missing values. This is an approximation which can add variance to the dataset.

4. Check for null values*:*

We can check the null values in our dataset with pandas library With the help of *info()* we can found total number of entries as well as count of non-null values with datatype of all features.

We work on large dataset so it will be a good thing to get the count of all null values corresponding to each features and it will be done by using *sum()* we work on large dataset so it will be a good thing to get the count of all null values corresponding to each features and it will be done by using *sum().*

5. Replacing Null values with Strategy*:*

For replacing null values, we use the strategy that can be applied on a feature which has numeric data. We can calculate the *Mean, Median or Mode* of the feature and replace it with the missing values

6. Divide the dataset into Dependent & Independent variable*:*

After importing the dataset, the next step would be to identify the independent variable (X) and the dependent variable (Y).

Feature Selection*:* -

Feature selection acts like a discerning chef carefully choosing ingredients. It is the process of selecting the most relevant features from your data to train a model. Just like a well-chosen recipe yields a tastier dish, using the right features can significantly improve our machine learning model. Feature selection offers several key benefits:

- **Enhanced Model Performance:** By eliminating irrelevant or redundant features, you focus the model on the data that truly matters. This can lead to better accuracy, reduced overfitting, and potentially faster training times.
- **Improved Interpretability:** With fewer features to consider, it becomes easier to understand the inner workings of your model. This allows you to

diagnose issues more easily and gain valuable insights from the relationships between the chosen features and the target variable.

- **Reduced Computational Cost:** Training models with fewer features requires less computational power. This is particularly advantageous for datasets with a high number of dimensions or when working with limited resources.

There are three main approaches to feature selection:

- **Filter Methods:** These techniques rely on statistical analysis to evaluate individual features based on their correlation with the target variable. This is a fast and efficient approach, often used as a pre-processing step.
- **Wrapper Methods:** As the name suggests, these methods involve training a machine learning model on different subsets of features and measuring their performance. The feature subset that yields the best performance is then selected. While this approach can be more accurate than filter methods, it can also be computationally expensive.
- **Embedded Methods:** Certain machine learning models have built-in feature selection capabilities. These methods often perform feature selection during the model training process itself. This can be a convenient option, but the selection criteria may be less transparent compared to other methods.

By carefully choosing the right feature selection technique for our specific data and model, we can significantly enhance the effectiveness and efficiency of our machine learning projects.

Fitting into Model: -

Model fitting is the heart of machine learning. It is the process of training a model on data to learn the underlying patterns and relationships. The goal is to create a model that can accurately predict outcomes for new, unseen data.

Importance:

- **Generalizability:** A well-fit model can generalize its learnings to similar data beyond the training set. This allows for reliable predictions on real-world applications.

- **Accuracy:** The fitting process optimizes the model's parameters to minimize errors between predictions and actual values.

How it is done:

1. **Data Preparation:** Clean and pre-process data to ensure its quality and suitability for the chosen machine learning algorithm.
2. **Model Selection:** Choose an appropriate algorithm based on the problem we are trying to solve (classification, regression, etc.).
3. **Training:** The model learns from the training data. The algorithm adjusts internal parameters to minimize the prediction error.
4. **Evaluation:** Use a separate test set to assess the model's performance on unseen data. Identify issues like overfitting (memorizing the training data) or underfitting (failing to capture the patterns).
5. **Refinement:** Based on the evaluation, you may need to adjust the model by tuning hyperparameters (settings that control the model's behavior) or even go back to step 2 and try a different model.

Fitting is an iterative process. By refining the model, we can achieve a good balance between accuracy and generalizability.

ALGORITHMS USED

KN*N:*

K-Nearest Neighbours (KNN) is a simple and intuitive machine learning algorithm used for both classification and regression tasks. KNN operates based on the principle that objects (data points) with similar features tend to belong to the same class or have similar output values.

1. **KNN Principle**: The basic idea behind KNN is to classify a new data point by looking at the K nearest neighbours in the training dataset. The class label or output value of the new data point is determined by the majority vote (for classification) or the average (for regression) of the K nearest neighbours.

2. **Distance Metric**: KNN calculates the distance between data points using a distance metric such as Euclidean distance or Manhattan distance. The distance metric determines the similarity or dissimilarity between feature vectors.

3. **Parameter K**: The value of K in KNN represents the number of nearest neighbours to consider for making predictions. It is a hyperparameter that needs to be predefined before training the model. A larger K value smooths out the decision boundaries but may lead to a loss of local patterns, while a smaller K value may be sensitive to noise in the data.

4. **Training**: KNN does not involve explicit training or model fitting. Instead, it stores the training dataset and uses it directly during the prediction phase.

5. **Classification**: In the case of classification, KNN assigns the class label of the new data point based on the majority class among its K nearest neighbours. Each neighbour’s vote is weighted equally. For example, if K = 5 and three neighbours belong to Class A while two belong to Class B, the new data point will be classified as Class A.

6. **Regression**: For regression tasks, KNN calculates the average output value of the K nearest neighbours and assigns it as the predicted value for the new data point. This average value represents a continuous estimate based on the values of the neighbouring data points.

7. **Scaling**: It is often necessary to scale the input features before applying KNN to avoid bias towards features with larger numerical ranges. Feature scaling ensures that all features contribute equally to the distance calculation.

8. **Choice of K**: The choice of the optimal K value depends on the dataset and the problem at hand. It can be determined through techniques such as cross validation or grid search, where different K values are tested to find the one that yields the best performance.

KNN is a straightforward algorithm to implement and interpret. However, it can be computationally expensive, especially with large datasets, as it requires calculating distances for all data points. Additionally, KNN is sensitive to the choice of the distance metric and the value of K. Despite these considerations, KNN is widely used, especially in cases where the decision boundaries are not well-defined or when there is no explicit training phase required.

Logistic Regression*:*

Logistic regression is a popular and widely used machine learning algorithm for binary classification tasks. It is used to predict a categorical outcome or class label based on a set of input features. The algorithm gets its name from the logistic function, also known as the sigmoid function, which is used to model the relationship between the input variables and the probability of the outcome.

1. **Problem Statement**: Logistic regression is primarily used when the outcome variable is binary or categorical, such as yes/no, true/false, or 0/1. The goal is to estimate the probability of a particular class label based on input features.

2. **Hypothesis Representation**: Logistic regression uses a logistic or sigmoid function to map the input features to a probability value between 0 and 1. The logistic function transforms a linear combination of the input features using the equation:

$$p = 1 / (1 + e^{(-z)}) \text{ ------------ (I)}$$

where p is the predicted probability, and z is the linear combination of the input features and their respective weights.

3. **Training**: The logistic regression model is trained using a labelled dataset, where the input features and corresponding class labels are known. The algorithm estimates the optimal values for the weights by minimizing a cost function, such as the maximum likelihood or cross-entropy loss function.

4. **Decision Boundary**: Once the model is trained, it can make predictions by applying the learned weights to new input data. The decision boundary is the threshold at which the predicted probability is used to classify the data into one of the two classes. By default, the decision boundary is set at 0.5, but it can be adjusted based on the problem's requirements.

5. **Evaluation**: The performance of the logistic regression model is assessed using various evaluation metrics, such as accuracy, precision, recall, and F1 score. These metrics help measure the model's ability to correctly classify instances and handle class imbalance.

Logistic regression has several advantages, including simplicity, interpretability, and the ability to handle both numerical and categorical input features.

However, it assumes a linear relationship between the input features and the log-odds of the outcome, and it may struggle with complex nonlinear patterns. In such cases, more advanced algorithms like decision trees or neural networks might be more appropriate.

Decision Tree*:*

Decision trees are a fundamental and versatile machine learning algorithm used for both classification (predicting categories) and regression (predicting continuous values). They are popular for their simplicity, interpretability, and effectiveness.

How it Works:

Imagine a flowchart where each node represents a question about the data features, and each branch represents the answer. The tree recursively splits the data based on these questions, aiming to arrive at a final prediction (classification) or a predicted value (regression) at the leaf nodes.

Building a Decision Tree:

1. **Start with the Root Node:** This node represents the entire dataset.
2. **Choose the Best Splitting Feature:** At each internal node, the algorithm finds the feature that best separates the data into subsets that are more homogeneous regarding the target variable.
3. **Split the Data:** The data is split based on the chosen feature's values, creating branches.
4. **Repeat:** Steps 2 and 3 are repeated for each branch until a stopping criterion is met, such as reaching a certain depth or data purity.

Importance:

- **Interpretability:** Decision trees are easy to understand and visualize. You can follow the branches to see the logic behind the model's predictions.
- **No Feature Scaling:** Unlike some algorithms, decision trees do not require data scaling, making them simpler to implement.
- **Effective for Various Problems:** They can handle both categorical and numerical data, making them suitable for a wide range of tasks.

Use Cases:

- **Customer churn prediction:** Identify customers at risk of leaving.
- **Loan approval prediction:** Assess creditworthiness of loan applicants.
- **Spam email classification:** Filter out unwanted emails.
- **Medical diagnosis support:** Assist doctors in preliminary diagnosis based on symptoms.

Evaluation:

- **Accuracy Metrics:** Metrics like classification accuracy or mean squared error (regression) are used to assess the model's performance on a separate test set.
- **Overfitting:** Decision trees can overfit the training data, leading to poor performance on unseen data.
  Techniques like pruning or using ensemble methods (combining multiple trees) can help mitigate this.

CODING

Importing dependencies:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns #for analysis from sklearn.preprocessing
import LabelEncoder #to encode a datatype to another one

data = pd.read_csv('/content/kidney_disease.csv')

# make copy of data
df = data.copy ()
df.head (10)
```

Analyzing features:

```
figure, axe = plt.subplots(5,3,figsize=(15, 20))
axe[0,0].hist(df['bp'])
axe[0,0].set_xlabel("histogram bp")
axe[0,1].hist(df['age'])
axe[0,1].set_xlabel("histogram age")
axe[0,2].hist(df['bu'])
axe[0,2].set_xlabel("histogram bu")

axe[1,0].hist(df["sg"])
axe[1,0].set_xlabel("histogram sg")
axe[1,1].hist(df["al"])
axe[1,1].set_xlabel("histogram al")
```

```
axe[1,2].hist(df["sc"])
axe[1,2].set_xlabel("histogram sc")

axe[2,0].hist(df["su"])
axe[2,0].set_xlabel("histogram su")
axe[2,1].hist(df["bgr"])
axe[2,1].set_xlabel("histogram bgr")
axe[2,2].hist(df["sod"])
axe[2,2].set_xlabel("histogram sod")

axe[3,0].hist(df["pot"],bins=30)
axe[3,0].set_xlabel("histogram pot")
axe[3,1].hist(df["hemo"])
axe[3,1].set_xlabel("histogram hemo")
axe[3,2].hist(df["classification"])
axe[3,2].set_xlabel("histogram classification")

axe[4,0].hist(df["pcv"])
axe[4,0].set_xlabel("histogram pcvt")
axe[4,1].hist(df["wc"])
axe[4,1].set_xlabel("histogram wc")
axe[4,2].hist(df["rc"])
axe[4,2].set_xlabel("histogram rc")

plt.show()
```

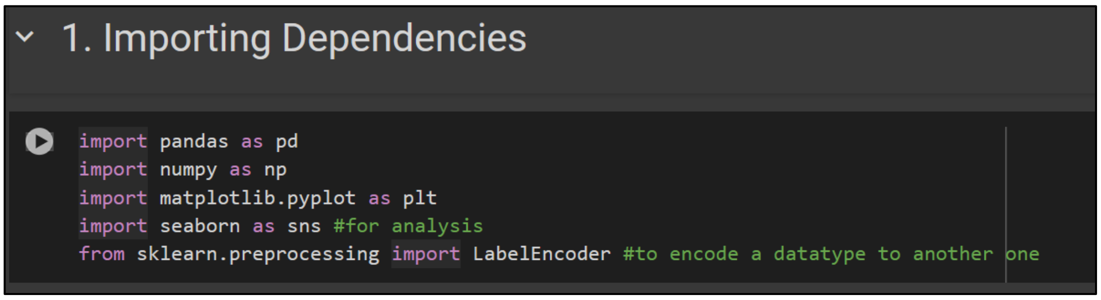
Data cleaning and Processing:

```
#fill null values by mean
df['age'].fillna(df['age'].mean(), inplace-True)
df['bp'].filina(df('bp'].median(), inplace-True)
df[sg"].fillna(df['sg'].mean(), inplace-True)
```

```
df['al].filina(df['al').median(), inplace-True)
df['su'].filina(df['su'].median(), inplace-True) df['rbc'].Filina(df['rbc'].mode()[0],
inplace-True)
df['pc'].fillna(df['pc'].mode()[0], inplace-True)

#filling missing values in column 'pcc' by 'notpresent' which is mode
df['pcc'].fillna(df['pcc'].mode()[0], inplace True)

#filling missing values in column 'ba'' by 'notpresent' which made of ba
df['ba'].fillna(df['ba'].mode()[0], inplace True) df]'bgr'].fillna(df['bgr'].median(),
inplace=True)
df['bu'].fillna(df['bu'].mean(), inplace=True) df['sc'].fillna(df['sc'].mean(),
inplace=True)
df['sod'].fillna(df['sod'].mean(), inplace True)
df['pot'].fillna(df['pot'].mean(), inplace=True), df['hemo'].fillna(df['hemo'].mean(),
inplace-Trud

#Convert 'pcv' column to numeric data type
df['pcv'] = pd.to_numeric(df['pcv'], errors coerce')
df['pcv'].fillna(df['pcv'].mean(), inplace=True)

#convert 'wc' column datatype to numeric data type
df['wc'] pd.to_numeric(df('wc'), errors='coerce')
df['wc'].fillna(df['wc'].mean(), inplace=True) #change 'rc' to numeric data type
df['rc'] = pd.to_numeric(df['rc'], errors= 'coerce')
df['rc'].fillna(df['rc'].mean(), inplace=True)
df['htn'].filina(df['htn'].mode()[0], inplace=True) df['de'].fillna(df['dm'].mode()[0],
inplace=True)
df['cad'].fillna(df['cad'].mode()[0], inplace True)
df['appet'].filina (df['appet'].mode()[e], inplace=True)
df['pe'].fillna(df['pe'].mode()[0], inplace True)
df['ane'].fillna(df('ane'].mode()[0], inplace-True)
```

IMPLEMENTATION

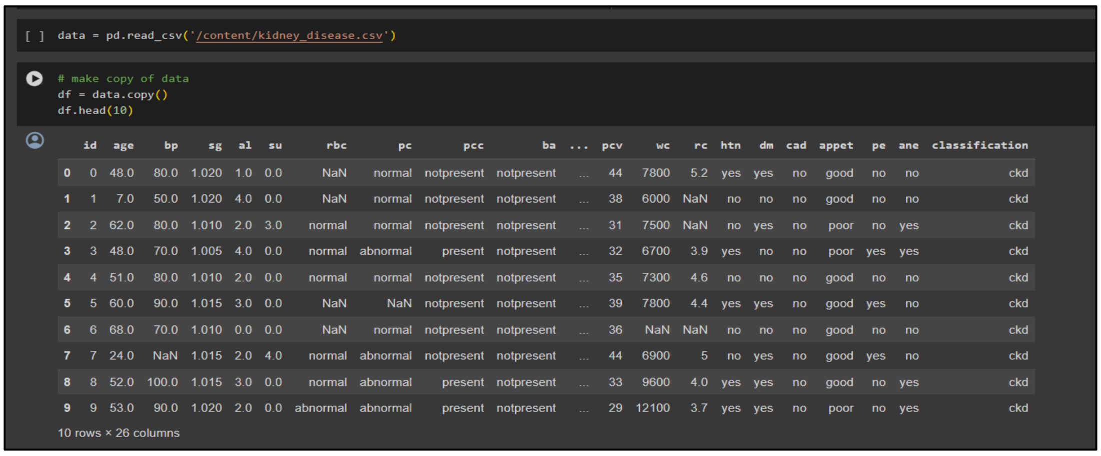
Following are the screenshots for the implementation of the code using Google Collab. The code written along with their respective outputs are shown below:



1. Importing Dependencies

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns #for analysis
from sklearn.preprocessing import LabelEncoder #to encode a datatype to another one
```

Fig. 2.1: Importing dependencies

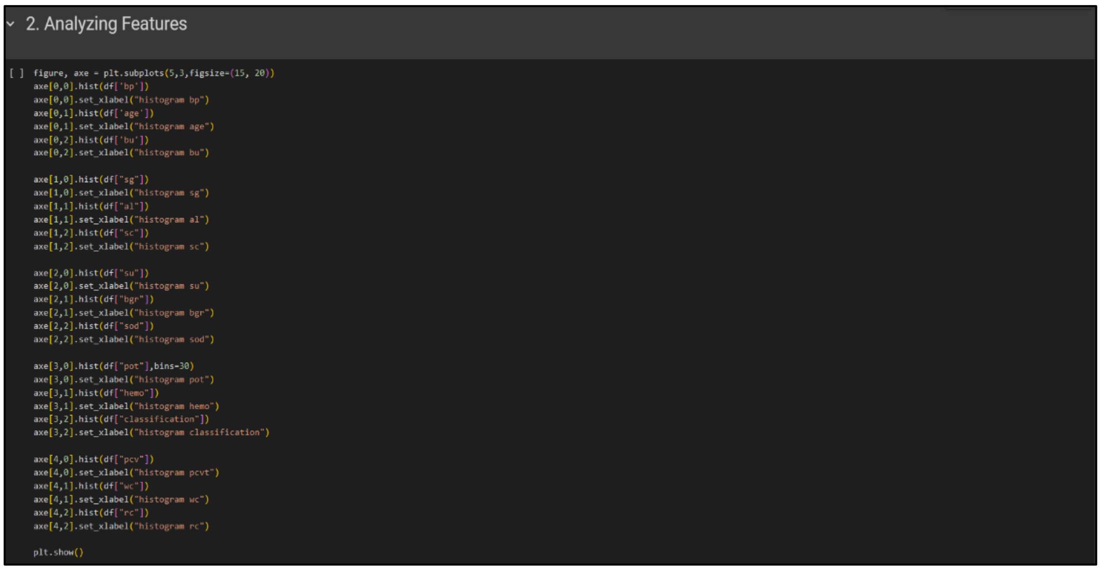


```
[ ] data = pd.read_csv('/content/kidney_disease.csv')
```

```
# make copy of data
df = data.copy()
df.head(10)
```

```
   id   age     bp     sg   al   su       rbc        pc         pcc          ba  ...  pcv     wc   rc  htn   dm  cad  appet   pe  ane  classification
0   0  48.0   80.0  1.020  1.0  0.0       NaN    normal  notpresent  notpresent  ...   44   7800  5.2  yes  yes   no   good   no   no             ckd
1   1   7.0   50.0  1.020  4.0  0.0       NaN    normal  notpresent  notpresent  ...   38   6000  NaN   no   no   no   good   no   no             ckd
2   2  62.0   80.0  1.010  2.0  3.0    normal    normal  notpresent  notpresent  ...   31   7500  NaN   no  yes   no   poor   no  yes             ckd
3   3  48.0   70.0  1.005  4.0  0.0    normal  abnormal     present  notpresent  ...   32   6700  3.9  yes   no   no   poor  yes  yes             ckd
4   4  51.0   80.0  1.010  2.0  0.0    normal    normal  notpresent  notpresent  ...   35   7300  4.6   no   no   no   good   no   no             ckd
5   5  60.0   90.0  1.015  3.0  0.0       NaN       NaN  notpresent  notpresent  ...   39   7800  4.4  yes  yes   no   good  yes   no             ckd
6   6  68.0   70.0  1.010  0.0  0.0       NaN    normal  notpresent  notpresent  ...   36    NaN  NaN   no   no   no   good   no   no             ckd
7   7  24.0    NaN  1.015  2.0  4.0    normal  abnormal  notpresent  notpresent  ...   44   6900    5   no  yes   no   good  yes   no             ckd
8   8  52.0  100.0  1.015  3.0  0.0    normal  abnormal     present  notpresent  ...   33   9600  4.0  yes  yes   no   good   no  yes             ckd
9   9  53.0   90.0  1.020  2.0  0.0  abnormal  abnormal     present  notpresent  ...   29  12100  3.7  yes  yes   no   poor   no  yes             ckd
10 rows × 26 columns
```

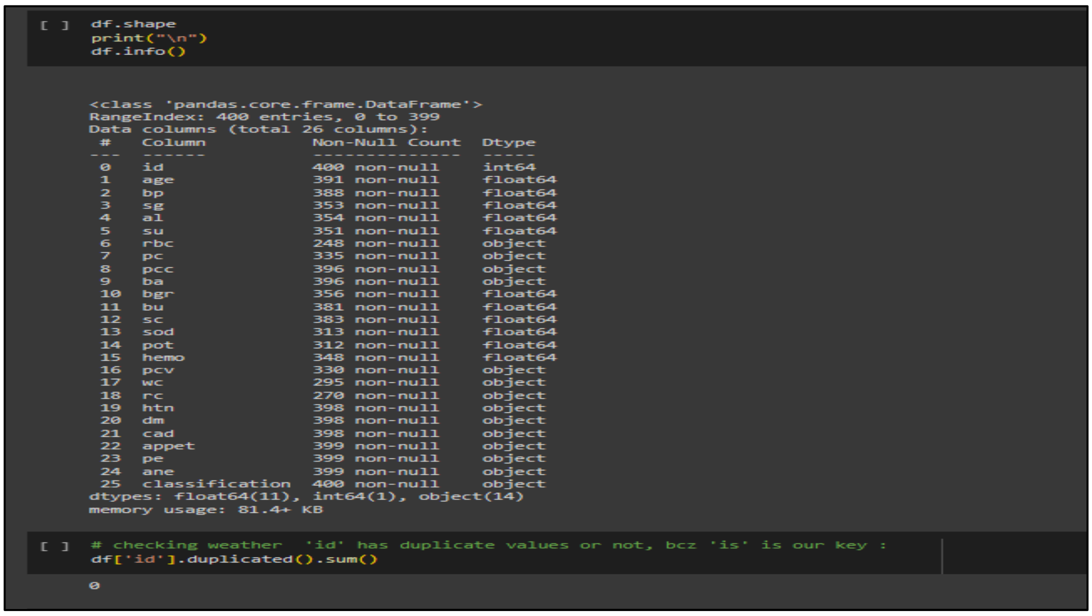
Fig. 2.2: Making copy of the data



2. Analyzing Features

```
[ ] figure, axe = plt.subplots(5,3,figsize=(15, 20))
    axe[0,0].hist(df['bp'])
    axe[0,0].set_xlabel("histogram bp")
    axe[0,1].hist(df['age'])
    axe[0,1].set_xlabel("histogram age")
    axe[0,2].hist(df['bu'])
    axe[0,2].set_xlabel("histogram bu")

    axe[1,0].hist(df["sg"])
    axe[1,0].set_xlabel("histogram sg")
    axe[1,1].hist(df["al"])
    axe[1,1].set_xlabel("histogram al")
    axe[1,2].hist(df["sc"])
    axe[1,2].set_xlabel("histogram sc")

    axe[2,0].hist(df["su"])
    axe[2,0].set_xlabel("histogram su")
    axe[2,1].hist(df["bgr"])
    axe[2,1].set_xlabel("histogram bgr")
    axe[2,2].hist(df["sod"])
    axe[2,2].set_xlabel("histogram sod")

    axe[3,0].hist(df["pot"],bins=30)
    axe[3,0].set_xlabel("histogram pot")
    axe[3,1].hist(df["hemo"])
    axe[3,1].set_xlabel("histogram hemo")
    axe[3,2].hist(df["classification"])
    axe[3,2].set_xlabel("histogram classification")

    axe[4,0].hist(df["pcv"])
    axe[4,0].set_xlabel("histogram pcvt")
    axe[4,1].hist(df["wc"])
    axe[4,1].set_xlabel("histogram wc")
    axe[4,2].hist(df["rc"])
    axe[4,2].set_xlabel("histogram rc")

    plt.show()
```

Fig. 3.1: Analyzing features



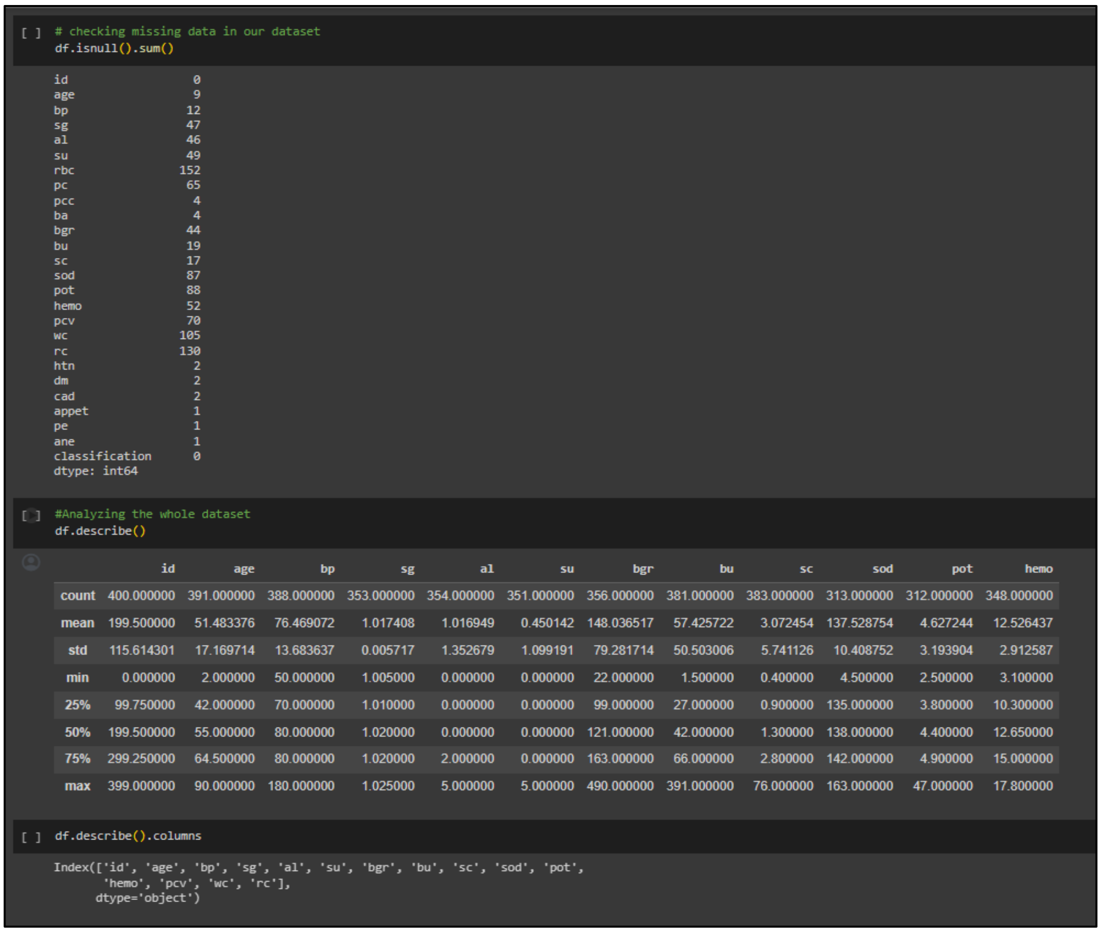
```
[ ]  df.shape
     print("\n")
     df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              400 non-null    int64
 1   age             391 non-null    float64
 2   bp              388 non-null    float64
 3   sg              353 non-null    float64
 4   al              354 non-null    float64
 5   su              351 non-null    float64
 6   rbc             248 non-null    object
 7   pc              335 non-null    object
 8   pcc             396 non-null    object
 9   ba              396 non-null    object
 10  bgr             356 non-null    float64
 11  bu              381 non-null    float64
 12  sc              383 non-null    float64
 13  sod             313 non-null    float64
 14  pot             312 non-null    float64
 15  hemo            348 non-null    float64
 16  pcv             330 non-null    object
 17  wc              295 non-null    object
 18  rc              270 non-null    object
 19  htn             398 non-null    object
 20  dm              398 non-null    object
 21  cad             398 non-null    object
 22  appet           399 non-null    object
 23  pe              399 non-null    object
 24  ane             399 non-null    object
 25  classification  400 non-null    object
dtypes: float64(11), int64(1), object(14)
memory usage: 81.4+ KB
```

```
[ ]  # checking weather  'id' has duplicate values or not, bcz 'is' is our key :
     df['id'].duplicated().sum()
```

```
0
```

Fig. 3.2. Finding types of data and their duplicates



```
[ ]  # checking missing data in our dataset
     df.isnull().sum()
```

```
id                  0
age                 9
bp                 12
sg                 47
al                 46
su                 49
rbc               152
pc                 65
pcc                 4
ba                  4
bgr                44
bu                 19
sc                 17
sod                87
pot                88
hemo               52
pcv                70
wc                105
rc                130
htn                 2
dm                  2
cad                 2
appet               1
pe                  1
ane                 1
classification      0
dtype: int64
```

```
[ ]  #Analyzing the whole dataset
     df.describe()
```

	id	age	bp	sg	al	su	bgr	bu	sc	sod	pot	hemo
count	400.000000	391.000000	388.000000	353.000000	354.000000	351.000000	356.000000	381.000000	383.000000	313.000000	312.000000	348.000000
mean	199.500000	51.483376	76.469072	1.017408	1.016949	0.450142	148.036517	57.425722	3.072454	137.528754	4.627244	12.526437
std	115.614301	17.169714	13.683637	0.005717	1.352679	1.099191	79.281714	50.503006	5.741126	10.408752	3.193904	2.912587
min	0.000000	2.000000	50.000000	1.005000	0.000000	0.000000	22.000000	1.500000	0.400000	4.500000	2.500000	3.100000
25%	99.750000	42.000000	70.000000	1.010000	0.000000	0.000000	99.000000	27.000000	0.900000	135.000000	3.800000	10.300000
50%	199.500000	55.000000	80.000000	1.020000	0.000000	0.000000	121.000000	42.000000	1.300000	138.000000	4.400000	12.650000
75%	299.250000	64.500000	80.000000	1.020000	2.000000	0.000000	163.000000	66.000000	2.800000	142.000000	4.900000	15.000000
max	399.000000	90.000000	180.000000	1.025000	5.000000	5.000000	490.000000	391.000000	76.000000	163.000000	47.000000	17.800000

```
[ ]  df.describe().columns
```

```
Index(['id', 'age', 'bp', 'sg', 'al', 'su', 'bgr', 'bu', 'sc', 'sod', 'pot',
       'hemo', 'pcv', 'wc', 'rc'],
      dtype='object')
```

Fig. 3.3. Checking missing values and analyzing whole dataset

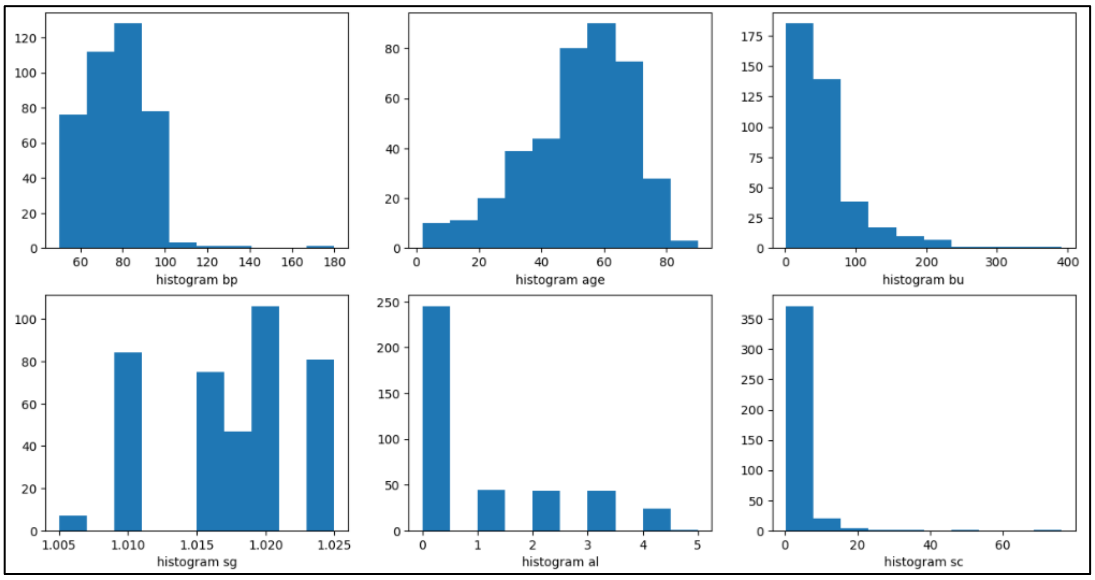


Fig. 4.1(i). Data plotting and for each feature

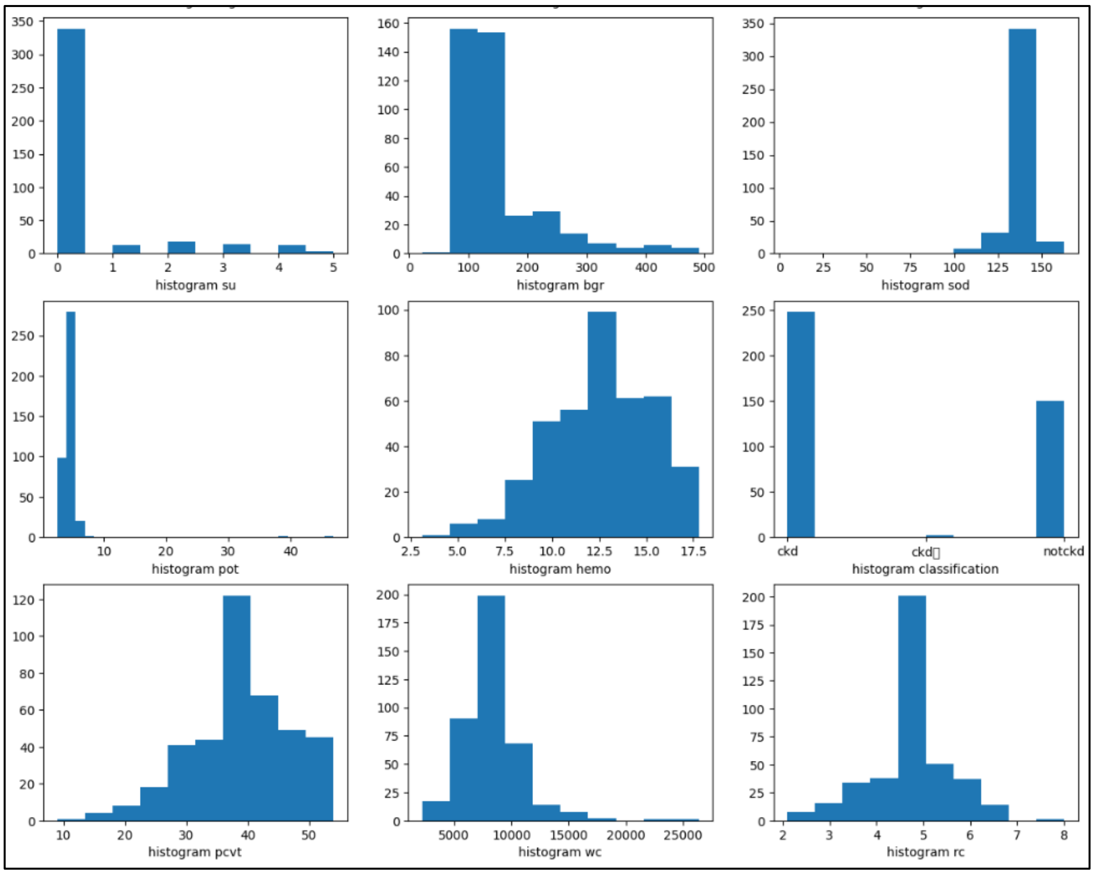
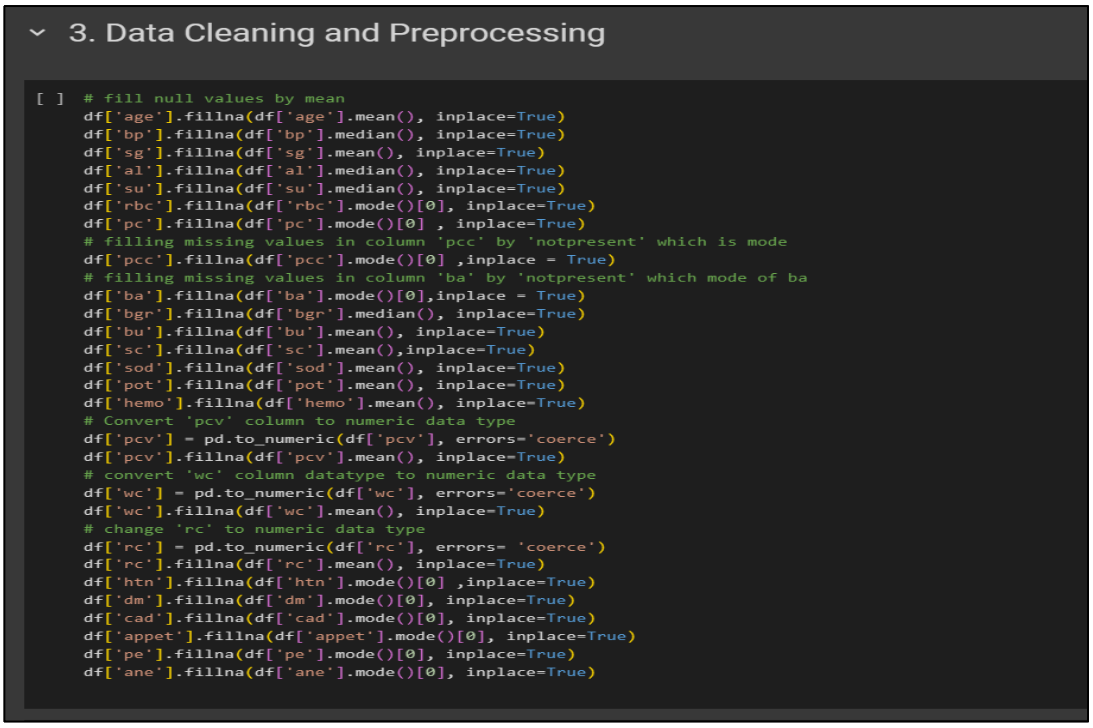


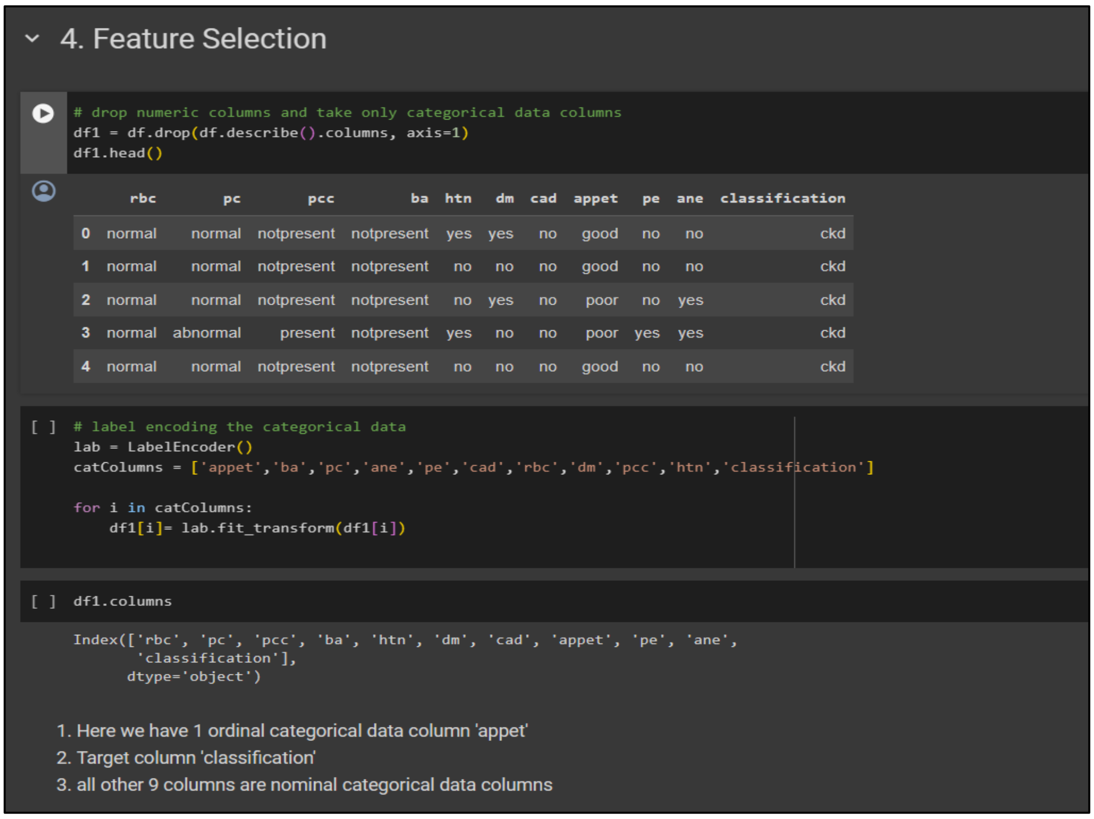
Fig. 4.1(ii). Data plotting and for each feature



3. Data Cleaning and Preprocessing

```
[ ]  # fill null values by mean
     df['age'].fillna(df['age'].mean(), inplace=True)
     df['bp'].fillna(df['bp'].median(), inplace=True)
     df['sg'].fillna(df['sg'].mean(), inplace=True)
     df['al'].fillna(df['al'].median(), inplace=True)
     df['su'].fillna(df['su'].median(), inplace=True)
     df['rbc'].fillna(df['rbc'].mode()[0], inplace=True)
     df['pc'].fillna(df['pc'].mode()[0] , inplace=True)
     # filling missing values in column 'pcc' by 'notpresent' which is mode
     df['pcc'].fillna(df['pcc'].mode()[0] ,inplace = True)
     # filling missing values in column 'ba' by 'notpresent' which mode of ba
     df['ba'].fillna(df['ba'].mode()[0],inplace = True)
     df['bgr'].fillna(df['bgr'].median(), inplace=True)
     df['bu'].fillna(df['bu'].mean(), inplace=True)
     df['sc'].fillna(df['sc'].mean(),inplace=True)
     df['sod'].fillna(df['sod'].mean(), inplace=True)
     df['pot'].fillna(df['pot'].mean(), inplace=True)
     df['hemo'].fillna(df['hemo'].mean(), inplace=True)
     # Convert 'pcv' column to numeric data type
     df['pcv'] = pd.to_numeric(df['pcv'], errors='coerce')
     df['pcv'].fillna(df['pcv'].mean(), inplace=True)
     # convert 'wc' column datatype to numeric data type
     df['wc'] = pd.to_numeric(df['wc'], errors='coerce')
     df['wc'].fillna(df['wc'].mean(), inplace=True)
     # change 'rc' to numeric data type
     df['rc'] = pd.to_numeric(df['rc'], errors= 'coerce')
     df['rc'].fillna(df['rc'].mean(), inplace=True)
     df['htn'].fillna(df['htn'].mode()[0] ,inplace=True)
     df['dm'].fillna(df['dm'].mode()[0], inplace=True)
     df['cad'].fillna(df['cad'].mode()[0], inplace=True)
     df['appet'].fillna(df['appet'].mode()[0], inplace=True)
     df['pe'].fillna(df['pe'].mode()[0], inplace=True)
     df['ane'].fillna(df['ane'].mode()[0], inplace=True)
```

Fig. 5.1. Data Cleaning and Preprocessing



4. Feature Selection

```
# drop numeric columns and take only categorical data columns
df1 = df.drop(df.describe().columns, axis=1)
df1.head()
```

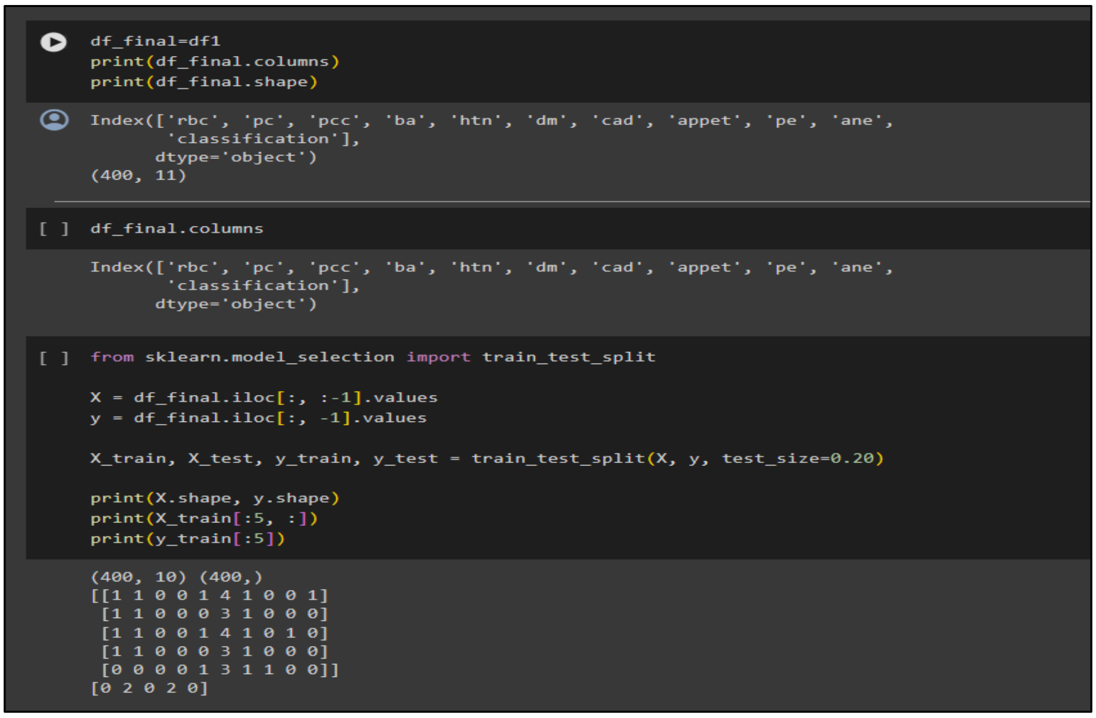
	rbc	pc	pcc	ba	htn	dm	cad	appet	pe	ane	classification
0	normal	normal	notpresent	notpresent	yes	yes	no	good	no	no	ckd
1	normal	normal	notpresent	notpresent	no	no	no	good	no	no	ckd
2	normal	normal	notpresent	notpresent	no	yes	no	poor	no	yes	ckd
3	normal	abnormal	present	notpresent	yes	no	no	poor	yes	yes	ckd
4	normal	normal	notpresent	notpresent	no	no	no	good	no	no	ckd

```
[ ]  # label encoding the categorical data
     lab = LabelEncoder()
     catColumns = ['appet','ba','pc','ane','pe','cad','rbc','dm','pcc','htn','classification']

     for i in catColumns:
         df1[i]= lab.fit_transform(df1[i])
```

```
[ ]  df1.columns

     Index(['rbc', 'pc', 'pcc', 'ba', 'htn', 'dm', 'cad', 'appet', 'pe', 'ane',
            'classification'],
           dtype='object')
```

1. Here we have 1 ordinal categorical data column 'appet'
2. Target column 'classification'
3. all other 9 columns are nominal categorical data columns

Fig 6.1. Feature Selection (I)



```
df_final=df1
print(df_final.columns)
print(df_final.shape)
```

```
Index(['rbc', 'pc', 'pcc', 'ba', 'htn', 'dm', 'cad', 'appet', 'pe', 'ane',
       'classification'],
      dtype='object')
(400, 11)
```

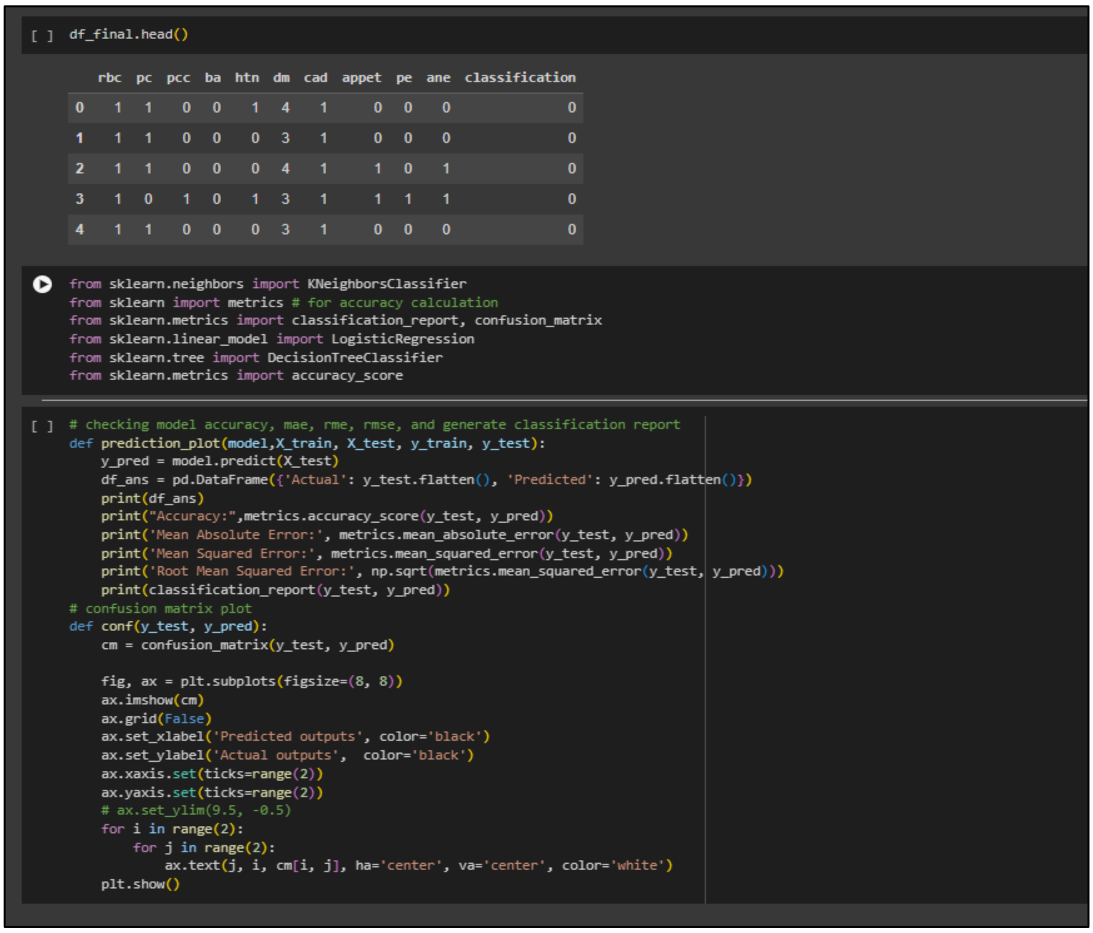
```
[ ] df_final.columns
```

```
Index(['rbc', 'pc', 'pcc', 'ba', 'htn', 'dm', 'cad', 'appet', 'pe', 'ane',
       'classification'],
      dtype='object')
```

```
[ ] from sklearn.model_selection import train_test_split

    X = df_final.iloc[:, :-1].values
    y = df_final.iloc[:, -1].values

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

    print(X.shape, y.shape)
    print(X_train[:5, :])
    print(y_train[:5])
```

```
(400, 10) (400,)
[[1 1 0 0 1 4 1 0 0 1]
 [1 1 0 0 0 3 1 0 0 0]
 [1 1 0 0 1 4 1 0 1 0]
 [1 1 0 0 0 3 1 0 0 0]
 [0 0 0 0 1 3 1 1 0 0]]
[0 2 0 2 0]
```

Fig 6.1. Feature Selection (I)



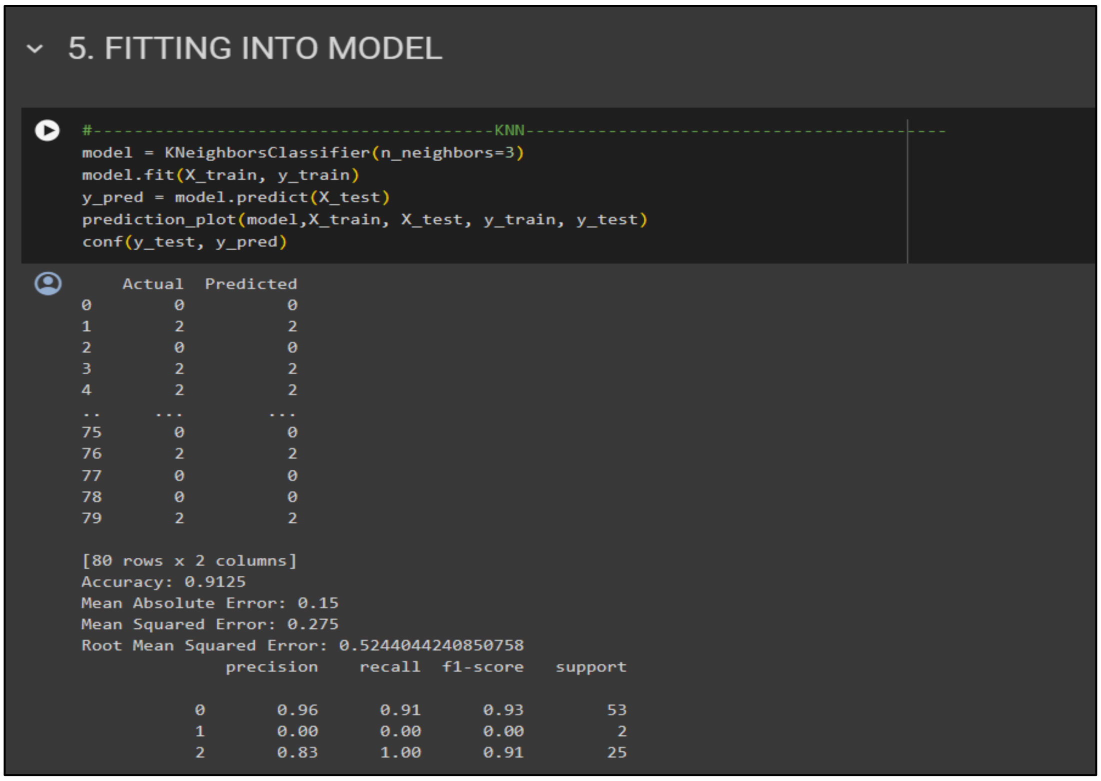
```
[ ] df_final.head()
```

	rbc	pc	pcc	ba	htn	dm	cad	appet	pe	ane	classification
0	1	1	0	0	1	4	1	0	0	0	0
1	1	1	0	0	0	3	1	0	0	0	0
2	1	1	0	0	0	4	1	1	0	1	0
3	1	0	1	0	1	3	1	1	1	1	0
4	1	1	0	0	0	3	1	0	0	0	0

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics # for accuracy calculation
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

```
[ ] # checking model accuracy, mae, rme, rmse, and generate classification report
    def prediction_plot(model,X_train, X_test, y_train, y_test):
        y_pred = model.predict(X_test)
        df_ans = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()})
        print(df_ans)
        print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
        print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
        print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
        print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
        print(classification_report(y_test, y_pred))
    # confusion matrix plot
    def conf(y_test, y_pred):
        cm = confusion_matrix(y_test, y_pred)

        fig, ax = plt.subplots(figsize=(8, 8))
        ax.imshow(cm)
        ax.grid(False)
        ax.set_xlabel('Predicted outputs', color='black')
        ax.set_ylabel('Actual outputs',  color='black')
        ax.xaxis.set(ticks=range(2))
        ax.yaxis.set(ticks=range(2))
        # ax.set_ylim(9.5, -0.5)
        for i in range(2):
            for j in range(2):
                ax.text(j, i, cm[i, j], ha='center', va='center', color='white')
        plt.show()
```

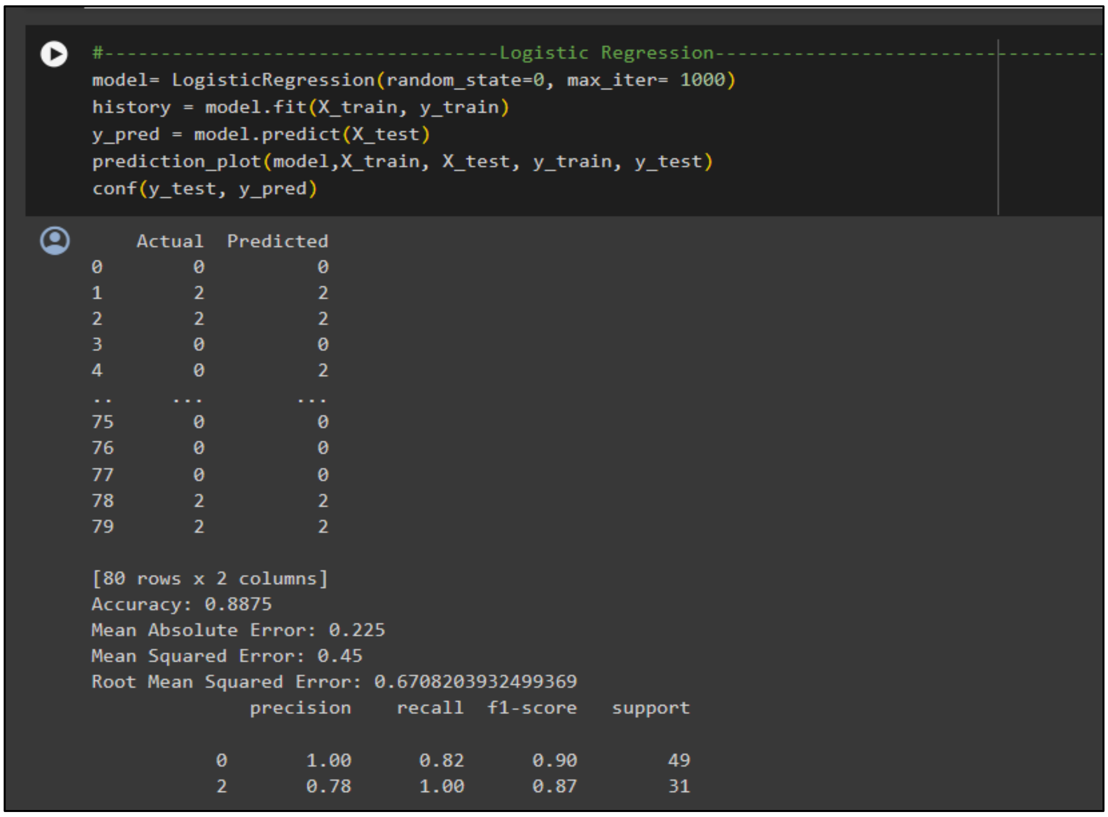
Fig 6.1. Feature Selection (II)



```
5. FITTING INTO MODEL

#--------------------------------------KNN--------------------------------------
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
prediction_plot(model,X_train, X_test, y_train, y_test)
conf(y_test, y_pred)
```

```
    Actual  Predicted
0        0          0
1        2          2
2        0          0
3        2          2
4        2          2
..     ...        ...
75       0          0
76       2          2
77       0          0
78       0          0
79       2          2

[80 rows x 2 columns]
Accuracy: 0.9125
Mean Absolute Error: 0.15
Mean Squared Error: 0.275
Root Mean Squared Error: 0.5244044240850758
              precision    recall  f1-score   support

           0       0.96      0.91      0.93        53
           1       0.00      0.00      0.00         2
           2       0.83      1.00      0.91        25
```

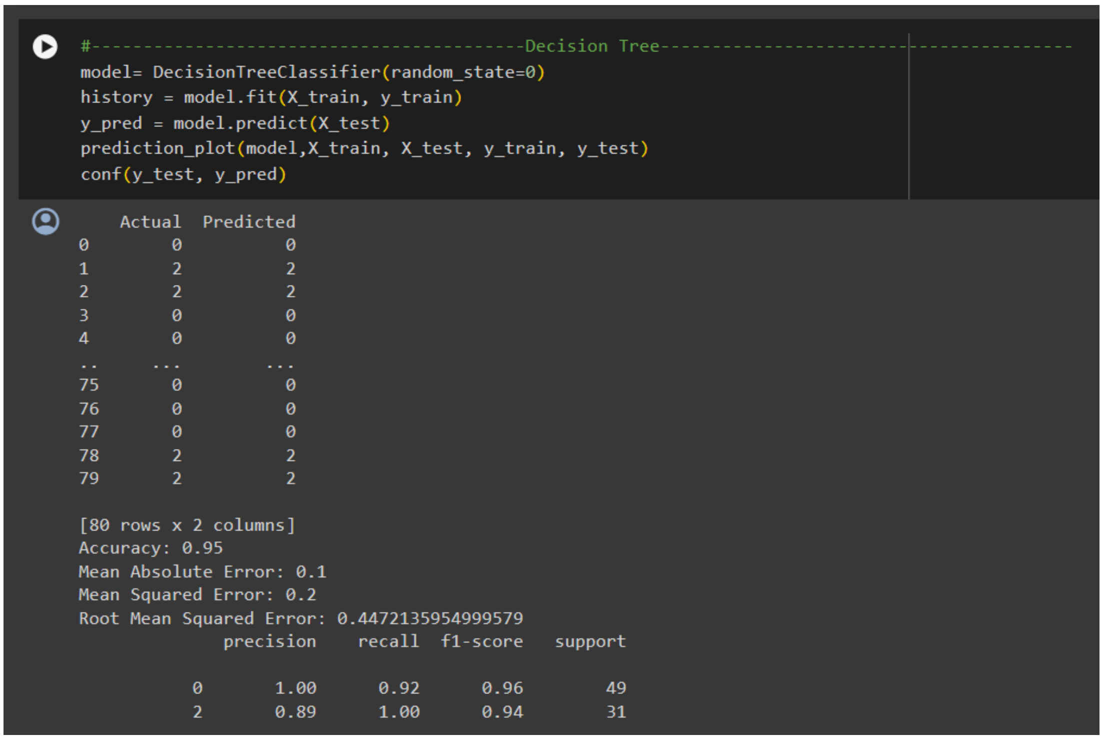
Fig 7.1. Fitting into the model for KNN



```
#---------------------------------Logistic Regression-------------------------------------
model= LogisticRegression(random_state=0, max_iter= 1000)
history = model.fit(X_train, y_train)
y_pred = model.predict(X_test)
prediction_plot(model,X_train, X_test, y_train, y_test)
conf(y_test, y_pred)
```

```
    Actual  Predicted
0        0          0
1        2          2
2        2          2
3        0          0
4        0          2
..     ...        ...
75       0          0
76       0          0
77       0          0
78       2          2
79       2          2

[80 rows x 2 columns]
Accuracy: 0.8875
Mean Absolute Error: 0.225
Mean Squared Error: 0.45
Root Mean Squared Error: 0.6708203932499369
              precision    recall  f1-score   support

           0       1.00      0.82      0.90        49
           2       0.78      1.00      0.87        31
```

Fig 7.2. Fitting into the model for Logistic Regression



```
#---------------------------------------Decision Tree-------------------------------------
model= DecisionTreeClassifier(random_state=0)
history = model.fit(X_train, y_train)
y_pred = model.predict(X_test)
prediction_plot(model,X_train, X_test, y_train, y_test)
conf(y_test, y_pred)
```

```
    Actual  Predicted
0        0          0
1        2          2
2        2          2
3        0          0
4        0          0
..     ...        ...
75       0          0
76       0          0
77       0          0
78       2          2
79       2          2

[80 rows x 2 columns]
Accuracy: 0.95
Mean Absolute Error: 0.1
Mean Squared Error: 0.2
Root Mean Squared Error: 0.4472135954999579
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        49
           2       0.89      1.00      0.94        31
```

Fig 7.3. Fitting into the model for Decision Tree

MACHINE LEARNING
ACCURACY EVALUATION METRICS

Choosing the right evaluation metric is crucial in machine learning. It helps us understand how well a model performs and identify areas for improvement.

Classification Metrics:

1. **Accuracy:**
- **Importance:** Overall effectiveness, but can be misleading for imbalanced datasets.
- **Use:** Provides a baseline understanding of correct predictions.
- **Formula:** Accuracy = (True Positives + True Negatives) / Total Samples

2. **Precision:**
- **Importance:** Measures the proportion of positive predictions that are actually correct (avoiding false positives).
- **Use:** Ideal for imbalanced datasets where false positives are costly.
- **Formula:** Precision = True Positives / (True Positives + False Positives)

3. **Recall:**
- **Importance:** Measures the proportion of actual positive cases that are correctly identified (avoiding false negatives).
- **Use:** Important when missing true positives is critical (e.g., medical diagnosis).
- **Formula:** Recall = True Positives / (True Positives + False Negatives)

4. **F1-Score:**
- **Importance:** Combines precision and recall into a single metric, addressing their limitations.
- **Use:** Provides a balanced view of both precision and recall.
- **Formula:** F1-Score = 2 * (Precision * Recall) / (Precision + Recall)

5. **Confusion Matrix:**

- **Importance:** Visualization tool that helps understand the model's performance across different classes.
- **Use:** Provides insights into classification errors (false positives, false negatives).
- **How it is done:** A table with rows representing actual classes and columns representing predicted classes. Each cell shows the number of cases assigned to that combination.

Regression Metrics:

1. **Mean Squared Error (MSE):**

- **Importance:** Measures the average squared difference between predicted and actual values.
- **Use:** Commonly used for regression tasks. Sensitive to outliers.
- **Formula:** MSE = (1/n) * Σ (yi - ŷi) ^2

  where yi is the actual value, ŷi is the predicted value, and n is the number of samples.

2. **Root Mean Squared Error (RMSE):**

- **Importance:** Like MSE but in the units of the target variable, making interpretation easier.
- **Use:** Preferred over MSE for easier interpretation of the error magnitude.
- **Formula:** RMSE = √(MSE)

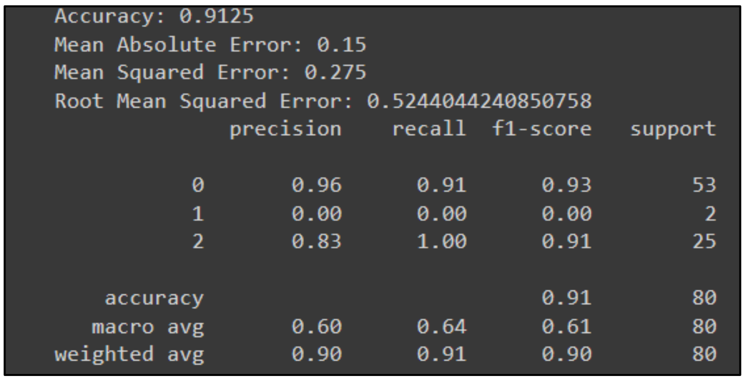
3. **Mean Absolute Error (MAE):**

- **Importance:** Measures the average absolute difference between predicted and actual values, less sensitive to outliers than MSE.
- **Use:** Useful for regression tasks when outliers are a concern.
- **Formula:** MAE = (1/n) * Σ|yi - ŷi|

  where yi is the actual value, ŷi is the predicted value, and n is the number of samples.

The final evaluation metrics obtained using the algorithms are as follows:

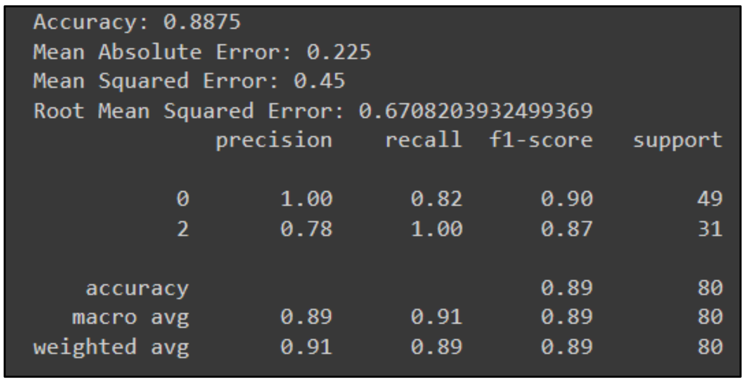
KNN Evaluation:



```
Accuracy: 0.9125
Mean Absolute Error: 0.15
Mean Squared Error: 0.275
Root Mean Squared Error: 0.5244044240850758
              precision    recall  f1-score   support

           0       0.96      0.91      0.93        53
           1       0.00      0.00      0.00         2
           2       0.83      1.00      0.91        25

    accuracy                           0.91        80
   macro avg       0.60      0.64      0.61        80
weighted avg       0.90      0.91      0.90        80
```

Fig. 8.1. Evaluation Metrics for KNN

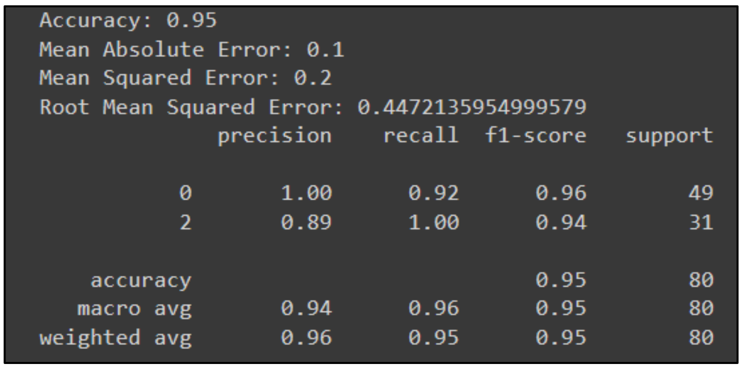
Logistic Regression Evaluation:



```
Accuracy: 0.8875
Mean Absolute Error: 0.225
Mean Squared Error: 0.45
Root Mean Squared Error: 0.6708203932499369
              precision    recall  f1-score   support

           0       1.00      0.82      0.90        49
           2       0.78      1.00      0.87        31

    accuracy                           0.89        80
   macro avg       0.89      0.91      0.89        80
weighted avg       0.91      0.89      0.89        80
```

Fig. 8.2. Evaluation Metrics for Logistic Regression

Decision Tree Evaluation:



```
Accuracy: 0.95
Mean Absolute Error: 0.1
Mean Squared Error: 0.2
Root Mean Squared Error: 0.4472135954999579
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        49
           2       0.89      1.00      0.94        31

    accuracy                           0.95        80
   macro avg       0.94      0.96      0.95        80
weighted avg       0.96      0.95      0.95        80
```

Fig. 8.3. Evaluation Metrics for Decision Tree

RESULTS

(i) Following is the table to show performance of the project and compare them among the different algorithms used with their accuracies:-

The values for the table have been calculated as follows:-

I. **Accuracy (in %)** = Accuracy * 100%
II. **Precision** = (Macro Average + Weighted Average) / 2
III. **Recall** = (Macro Average + Weighted Average) / 2
IV. **F1-Score** = (Macro Average + Weighted Average) / 2

These values have been calculated for all the evaluation metrics of all the algorithms used for training the model.

Metrics → Algos used ↓	Accuracy (in %)	Precision	Recall	F1-Score
KNN	**91.25**	**0.750**	**0.775**	**0.755**
Logistic Regression	**88.75**	**0.900**	**0.900**	**0.890**
Decision Tree	**95.00**	**0.950**	**0.955**	**0.950**

Fig. 9. Table of results obtained by using the respective algorithms

(ii) Following are the final results, i.e., the confusion matrices of the outputs formed by applying the above algorithms on the machine learning model: -

KNN:

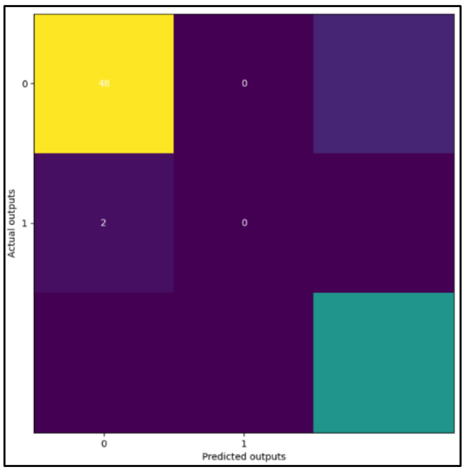


Fig. 10.1. Confusion Matrix for KNN

Logistic Regression:

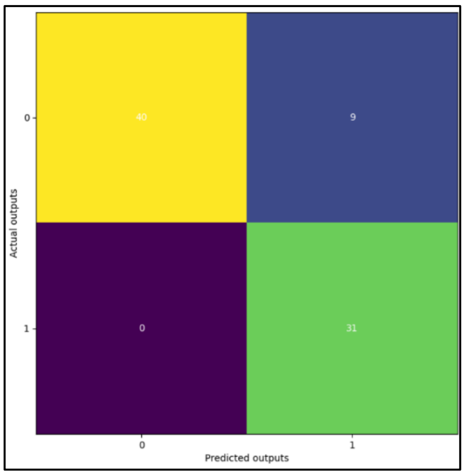


Fig. 10.2. Confusion Matrix for Logistic Regression

Decision Tree:

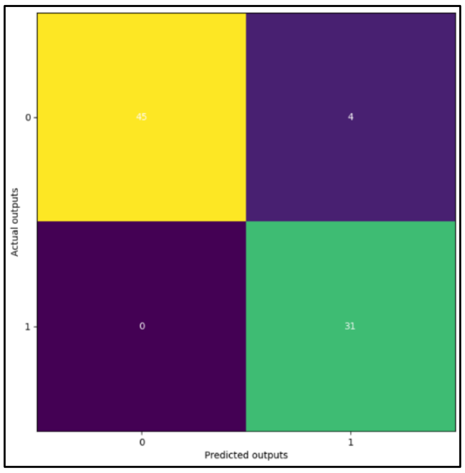


Fig. 10.3. Confusion Matrix for Decision Tree

CONCLUSION AND FUTURE SCOPE

This training has introduced us to Machine Learning. Now, we know that **Machine Learning is a technique of training machines to perform the activities a human brain can do, a bit faster and better than an average human-being**. Today we have seen that the machines can beat human champions in games such as Chess, Mahjong, which are considered very complex.

We have seen that machines can be trained to perform human activities in several areas and can aid humans in living better lives. Machine learning is quickly growing field in computer science. It has applications in nearly every other field of study and is already being implemented commercially because **machine learning can solve problems too difficult or time consuming for humans to solve.**

To describe machine learning in general terms, a variety models are used to learn patterns in data and make accurate predictions based on the patterns it observes. Machine Learning can be a Supervised or Unsupervised. If we have a lesser amount of data and clearly labelled data for training, we opt for Supervised Learning.

Unsupervised Learning would generally give better performance and results for large data sets. If we have a huge data set easily available, we go for deep learning techniques. We also have learned Reinforcement Learning and Deep Reinforcement Learning. We now know what Neural Networks are, their applications and limitations.

Specifically, we have developed a thought process for approaching problems that machine learning works so well at solving. We have learnt how machine learning is different than descriptive statistics.

REFERENCES

1. C. Bemando, E. Miranda, M. Aryuni, "Machine-Learning-Based Prediction Models of Coronary Heart Disease Using Naïve Bayes and Random Forest Algorithms," in 2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM), (IEEE, 2021), pp. 232–237

2. . R.P. Ram Kumar, Sanjeeva Polepaka, Performance comparison of random forest classifier and convolution neural network in predicting heart diseases, in Proceedings of the Third International Conference on Computational Intelligence and Informatics. ed. by K. Srujan Raju, A. Govardhan, B. Padmaja Rani, R. Sridevi, M. Ramakrishna Murty (Springer, Singapore, 2020)

3. A. Nithya, A. Appathurai, N. Venkatadri, D.R. Ramji, C.A. Palagan, Kidney disease detection and segmentation using artificial neural network and multi-kernel k-means clustering for ultrasound images. Measurement (2020). https://doi.org/10.1016/j.measu rement.2019.106952

4. F. Aqlan, R. Markle, A. Shamsan, "Data mining for chronic kidney disease prediction." in IIE Annual Conference. Proceedings, Institute of Industrial and Systems Engineers, (IISE 2017), pp. 1789–1794

5. A. Nishanth, T. Thiruvaran, Identifying important attributes for early detection of chronic kidney disease. IEEE Rev. Biomed. Eng. 11, 208–216 (2018)

6. R.S. Walse, G.D. Kurundkar, S.D. Khamitkar, A.A. Muley, P.U. Bhalchandra, S.N. Lokhande, Effective use of naïve bayes, decision tree, and random forest techniques for analysis of chronic kidney disease, in International Conference on Information and Communication Technology for Intelligent Systems. ed. by T. Senjyu, P.N. Mahalle, T. Perumal, A. Joshi (Springer, Singapore, 20).