INTRODUCTION

Overview

Chronic Kidney Disease (CKD) is a major medical problem and can be cured if treated in the early stages. Usually, people are not aware that medical tests, we take for different purposes could contain valuable information concerning kidney diseases. Consequently, attributes of various medical tests are investigated to distinguish which attributes may contain helpful information about the disease. The information says that it helps us to measure the severity of the problem and we make use of such information to build a machine learning model that predicts Chronic Kidney Disease

**Proposed System**

In this section, a detailed description of the data set creation, model preparation, and disease prediction has been given. The first action is data collection. Our proposed system collects structured and unstructured data obtained from various sources. After data collection, they are subjected to preprocessing and are split into cleaning and test data sets. Then the training data set is trained with the machine learning algorithms such as Random forest,Gaussian Naïve Bayes and Gradient Boosting classifier to a number of epochs for improving the accuracy of the prediction results. After multiple epochs, once the desired target is achieved, the developed model is ready for testing.

**THEORETICAL ANALYSIS**

**Block Diagram**

Diagram

Description automatically generated

**Hardware**/**Software** **Designing**

• 2 GB ram or above

• Dual core processor or above

• Internet connection

Software requirements

• Anaconda Navigator

• Python packages

• VS Studio

**FLOWCHART**

• User interacts with the UI (User Interface) to fill the information asked.

• Given input is analyses by the model which is integrated.

• Once model analyses the given information, the prediction is showcased on the UI

1. Data Collection

a. Collect the dataset or create the dataset

2. Understanding the data

a. Importing the required libraries

b. Reading the Dataset

c. EDA on Dataset

d. Take care of missing data

e. Data Visualization

f. Cleaning the Text

g. Building count vectors with scikit-learn Count-Vectorizer for text classification

h. Splitting Data into Train and Test

3. Model Building

a. Training and testing the model

b. Evaluation of Model

c. Saving the model

4. Application Building

a. Create an HTML file

b. Build Python Code

5. Final UI

a. Dashboard Of the flask app

Importing The Libraries

Import the necessary libraries as shown in the image

Import the required libraries for the model to run. The first step is usually importing the libraries that will be needed in the program.

●**Numpy**- It is an open-source numerical Python library. It contains a multi-dimensional array and matrix data structures. It can be used to perform mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.

**●Pandas** - It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

**●Seaborn**- Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

**●Matplotlib**- Visualisation with python. It is a comprehensive library for creating static,animated, and interactive visualizations in Python

 Reading The Dataset

●Import the dataset

●Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

●In pandas, we have a function called read\_csv() to read the dataset. As a parameter, we have to give the directory of the CSV file.

Table

Description automatically generated

Data Pre-Processing

In simple words, analysis is understanding the data with some features. Here we have displayed some graphs to analyze the data.

As we have understood how the data is let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

●Handling missing values

●Handling categorical data

●Handling outliers

●Scaling Techniques

●Splitting dataset into training and test set

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

A picture containing text

Description automatically generated

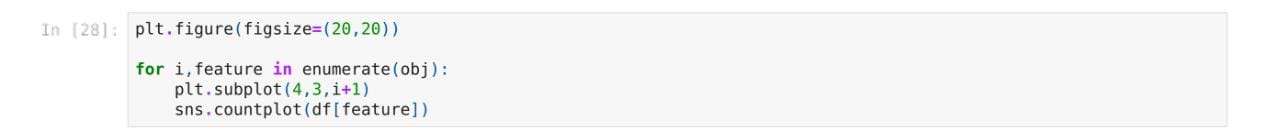
Graphical user interface, text, application, website

Description automatically generated

* In this dataset the missing values can be replaced by the mean for numeric columns.
* In Categorical columns missing values can be replaced by the mode.

Analysis

Here we perform some analysis techniques on the dataset features using the seaborn library.

Treemap chart

Description automatically generated with low confidence

Chart, bar chart, waterfall chart

Description automatically generated

Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project, we are using manual encoding with the help of list comprehension.

Graphical user interface, text, application, email

Description automatically generated

Splitting Data Into Train And Test

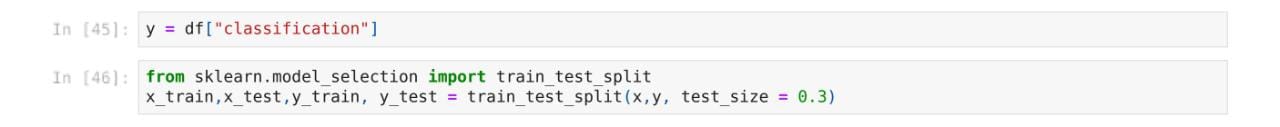
Now let’s split the Dataset into train and test sets

Changes: first split the dataset into x and y and then split the data set

Here x and y variables are created. On the x variable, df is passed by dropping the target variable. And on y target variable is passed. For splitting training and testing data, we are using the train\_test\_split() function from sklearn. As parameters, we are passing x, y, train\_size,test\_size.

Graphical user interface, application

Description automatically generated



**Model Building**

Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

Random Forest Model

A function named randomForest is created and train and test data are passed as the parameters. In RandomForestClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

Table

Description automatically generated with medium confidence

Gaussian Naïve Bayes Model

A function named GaussianNB is created and train and test data are passed as the parameters. Inside the function, the Gaussian Naive Bayes algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

Text

Description automatically generated

Gradient Boosting Classifier Model

A function named GradientBoostingClassifier is created and train and test data are passed as the parameters. Inside the function, the GradientBoostingClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

Text

Description automatically generatedSaving the Model

Now we choosing the model Random forest classifier.We save the model by using the pickel or joblib library.

Save the model using joblib.dump() function.

Graphical user interface, application, email, Teams

Description automatically generated

Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the user where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

Building HTML Pages

Building server side script

Building Html Pages

For this project create three HTML files namely

* index.html
* result.html

and save them in the templates folder.

Let’s see how our home.html page looks like:

A picture containing text, indoor

Description automatically generated

A picture containing text, indoor

Description automatically generated

Build Python Code

Import the libraries

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (\_name\_) as argument.

Render HTML page:

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, ‘/’ URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the result.html page earlier.

Text

Description automatically generated

Run The Application

•Open anaconda prompt from the start menu

•Navigate to the folder where your python script is.

•Now type “python app.py” command

•Navigate to the localhost where you can view your web page.

•Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

Text

Description automatically generated

Result

A picture containing diagram

Description automatically generated

Website

Description automatically generated

**ADVANTAGES AND DISADVANTAGES**

**Advantages**

* Help physicians to identify effective treatments and best practices.
* Patients exploit better and greater affordable healthcare services.
* Increases in the speed of working with large datasets and rapid report
* generation, faster analysis, improved operational efficiency and
* reduced operating cost.
* Data Mining can extract predictive knowledge from large databases.

**Disadvantages**

* Data Ownership issues.
* Privacy and Security related to Human Data Administration.
* It Involves privacy issues and security issues and
* Misuse or incorrect information.

**Applications**

* App
* Websites

CONCLUSION

Using this project, we can predict does we have chronic kidney disease or not. This program will check the information of our health report and predicts whether we have chronic kidney disease or not. When our health report is bad then it says that we have chronic kidney disease.

FUTURE SCOPE

This program allows users to predict if that whether we have chronic kidney disease or not. By the help of this prediction, we can take following measures to cure CKD. It helps to take measure for CKD before our kidney completely fails.