INTRODUCTION

Overview

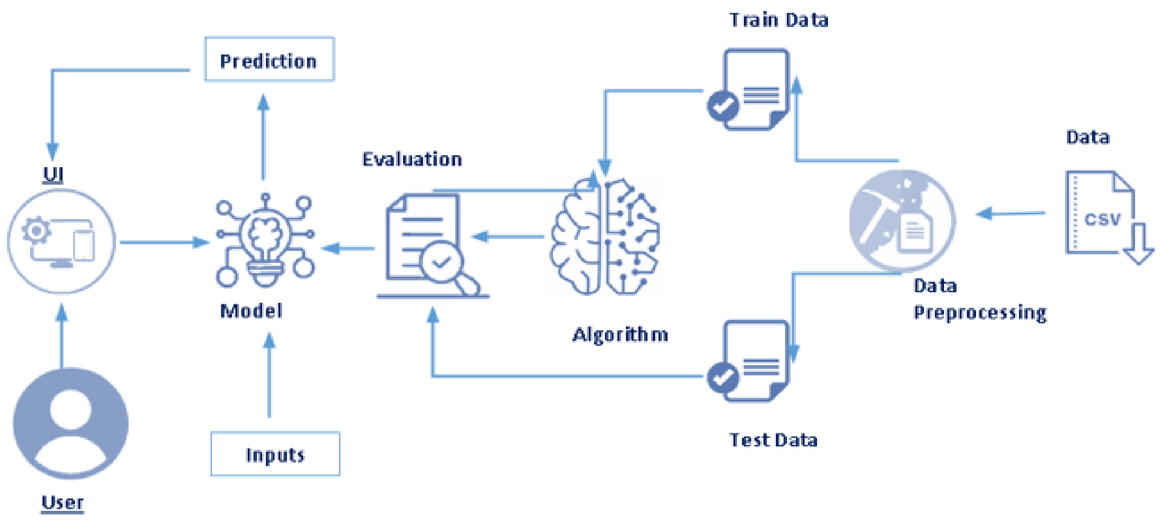
Chronic Kidney Disease (CKD) is a major medical problem and can be cured if treated in the early stages. Usually, people are not aware that medical tests, we take for different purposes could contain valuable information concerning kidney diseases. Consequently, attributes of various medical tests are 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 CNN and KNN 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.

3.THEORETICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware/Software Designing

- 2 GB ram or above
- Dual core processor or above
- Internet connection

Software requirements

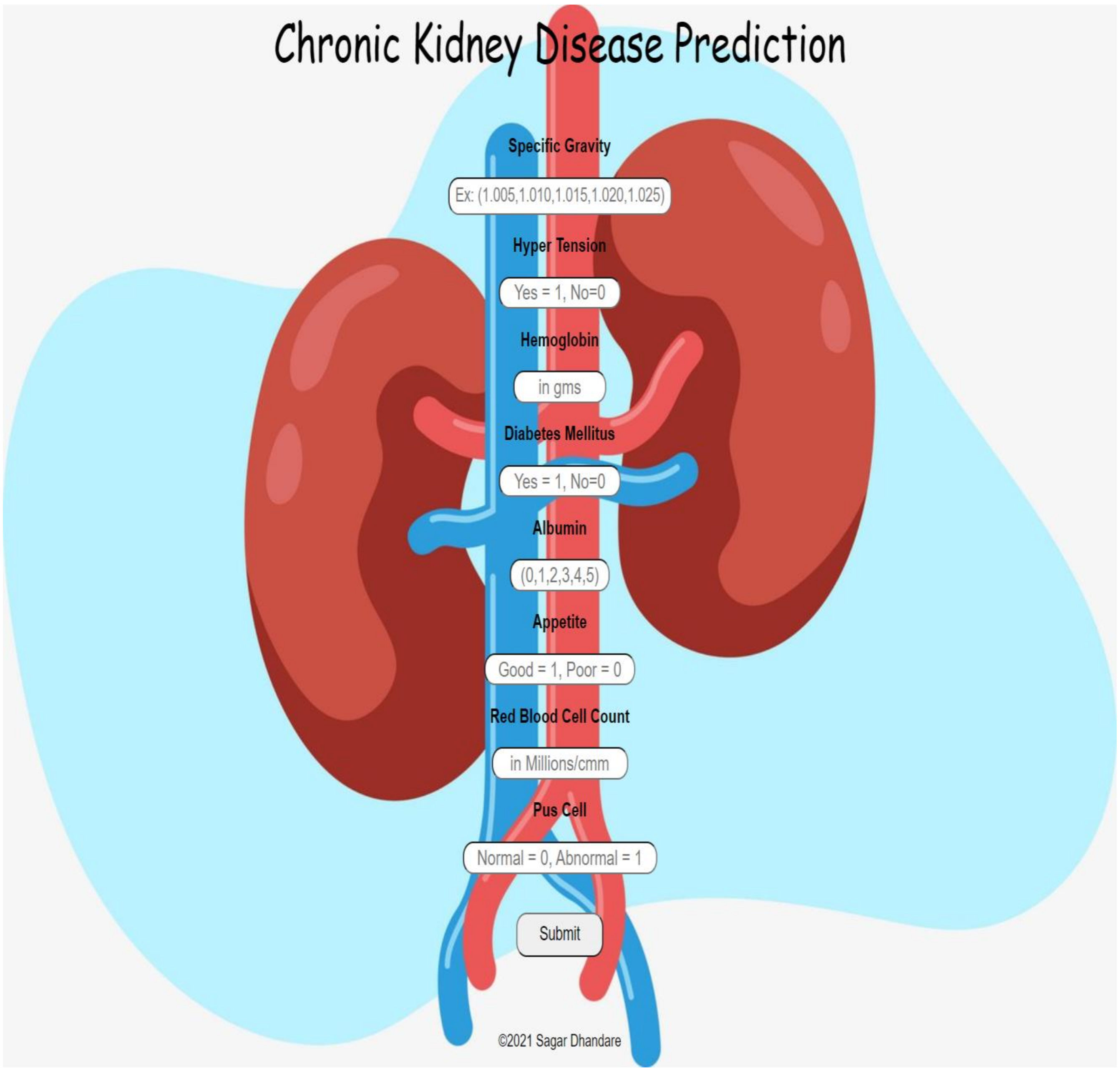
- Anaconda Navigator
- Python packages
- VS Studio

4. FLOWCHART

- User interacts with the UI (User Interface) to fill the information asked.
- Given input is analysed 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

6. RESULT



Chronic Kidney Disease Prediction

Specific Gravity

Ex: (1.005,1.010,1.015,1.020,1.025)

Hyper Tension

Yes = 1, No=0

Hemoglobin

in gms

Diabetes Mellitus

Yes = 1, No=0

Albumin

(0,1,2,3,4,5)

Appetite

Good = 1, Poor = 0

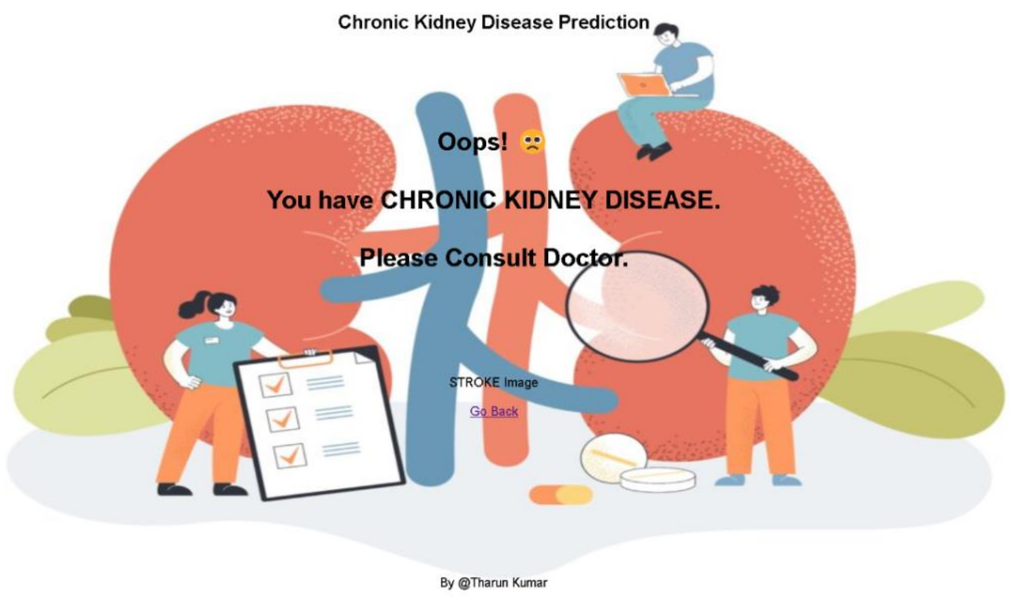
Red Blood Cell Count

in Millions/cmm

Pus Cell

Normal = 0, Abnormal = 1

©2021 Sagar Dhandare



Chronic Kidney Disease Prediction

Oops! 🙁

You have CHRONIC KIDNEY DISEASE.

Please Consult Doctor.

STROKE Image

Go Back

By @Tharun Kumar

6. 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.

8.Applications

- App
- Websites

9. CONCLUSION

Using this project, we can predict if 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 say that we have chronic kidney disease.

10. 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.

Appendix

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read_csv("kidney_disease.csv")

df.info()

df.head(10)
df.describe()
df["rc"].unique()
df["rc"] = df["rc"].replace("\t?", '0')
df["rc"].unique()
df["rc"] = df["rc"].astype(float)
df["wc"].unique()
df["pcv"] = df["pcv"].replace(("\t?", "\t43"), (0, 43))
df["pcv"] = df["pcv"].astype(float)
df.info()
df.drop(["id"], axis = 1, inplace = True)
df.describe()
obj = (df.select_dtypes(include = object)).columns
numeric = (df.select_dtypes(include = np.number)).columns
print(numeric, obj)
for i in obj:
   print(i)
   print(df[i].unique())
df["dm"] = df["dm"].replace({"\tno":"no", "\tyes":"yes", " yes":"yes"})
df["cad"] = df["cad"].replace("\tno", "no")
df["classification"] = df["classification"].replace("ckd\t", "ckd")
for i in obj:
   print(i)
   print(df[i].unique())
df.isnull().sum()
```

```
df["rc"].fillna(df["rc"].mean(), inplace=True)

for i in numeric:

    df[i].fillna(df[i].mean(),inplace=True)

for i in obj:

    df[i].fillna(df[i].mode()[0], inplace=True)

df.head(10)

from sklearn.preprocessing import LabelEncoder

for i in obj:

    le = LabelEncoder()

    df[i] = le.fit_transform(df[i])

df.info()

df.shape

df.corr()

import seaborn as sns

plt.figure(figsize = (15,15))

sns.heatmap(df.corr(), annot = True, fmt=".2f",linewidth=0.5)

df.corrwith(df.classification).plot(kind="bar", grid=True,figsize=(12,8),
title = "corr with target")

x = df.drop(["classification", "pot", "ba", "wc", "age", 'su', "ane"], axis =
1)

x.columns

x.columns=['Blood_pressure', "Specafic_gravity", "albumin",
"red_blood_cells", 'pus_cell', 'pus_cell_clumps',
"blood_glucose_random", 'blood_urea', 'serum_creatinine', 'sodium',
"hemoglobin", 'packed_cell_volume', 'red_blood_cell_count',
'hypertension', 'diabetesmellitus', 'coronary_artery_disease',
'appetite', 'pedal_edema']

x.columns

x.head(10)

y = df["classification"]

from sklearn.model_selection import train_test_split

x_train,x_test,y_train, y_test = train_test_split(x,y, test_size = 0.25)

x_train.shape, x_test.shape, y_train.shape

from sklearn.ensemble import RandomForestClassifier

from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import classification_report,
confusion_matrix,accuracy_score

ran_fore = RandomForestClassifier(n_estimators=100)

ran_fore.fit(x_train, y_train)

rf_pred = ran_fore.predict(x_test)

rf_score = round(ran_fore.score(x_train, y_train)*100, 2)

rf_test_score = round(ran_fore.score(x_test, y_test)*100, 2)

print("Random forest train score = \n", rf_score)

print("Random forest test score = \n", rf_test_score)

print("accuracy  = \n", accuracy_score(y_test,rf_pred))

print("confusion Matrix : \n", confusion_matrix(y_test, rf_pred))

print(classification_report(y_test, rf_pred))

import pickle

pickle.dump(ran_fore, open("CKD.pkl",'wb'))
```