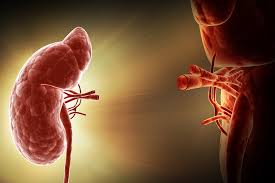
**CHRONIC KIDNEY DISEASE ANALYSIS**

**SmartBridge - Remote Summer Internship Program**

**Category: Machine Learning**

****

Team Members:

**SURISETTY SAIPHANEENDRA**

**Manda Rahul Reddy**

**Ajay Poreddy**

**Kolanu Manohar Reddy**

**Index**

1. Introduction

1.1 Overview

1.2 Purpose

2. Literature Survey

2.1 Existing Problem

2.2 Proposed Solution

3. Theoretical Analysis

3.1 Block Diagram

3.2 Hardware/Software Designing

4. Experimental Investigations

5. Flowchart

6. Result

7. Advantages & disadvantages

8. Applications

9. Conclusion

10. Future Scope

11. Bibliography

12. Appendix

**1. Introduction:-**

1.1 Overview:

Chronic Kidney Disease is a major medical problem and can be cured if treated in the early stages. people are not aware that medical tests, we take for different purposes could contain valuable information concerning kidney diseases.

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, the predicted survival of the patient after the illness, the pattern of the disease and work for curing the disease.

Treatment mostly aims to stop the kidney disease from getting worse, or at least delay the process. It is also important to treat complications such as anemia, acidosis or changes in bone metabolism. If the kidneys fail completely, dialysis or a kidney transplant can prolong the person’s life.

1.2 Purpose:

Technological development, including machine learning, has a huge impact on health through an effective analysis of various chronic diseases for more accurate diagnosis and successful treatment.

Our aim from the project is to make use of pandas, matplotlib, numpy, libraries from python to extract the libraries for machine learning for the chronic disease prediction and the results are compared across a number of classification algorithms and regression techniques. The dataset is pre-processed by completing and normalizing missing data. The most relevant features are selected from the dataset for improved accuracy and reduced training time.

**2**. **Literature Survey**

2.1 Existing Problem

CKD is linked to several complications with higher prevalence and intensity at lower levels of kidney function, which interact with each other and these complications contribute to high morbidity and mortality and poor quality of life.

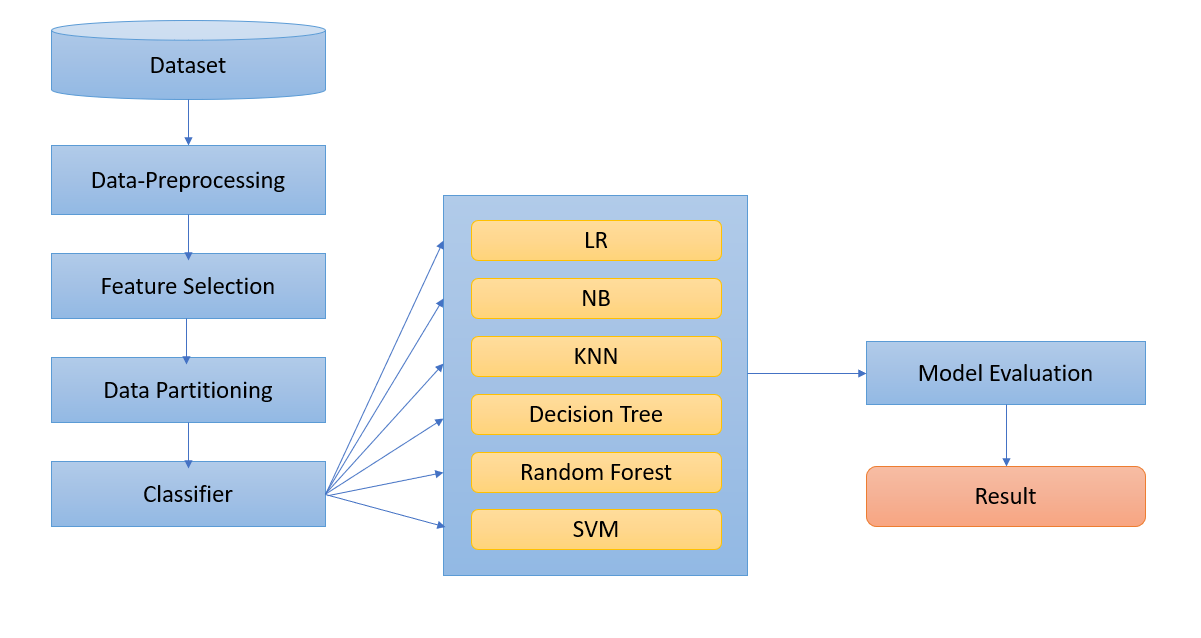
It’s a big challenge in medical canter or healthcare center to provide best quality treatment to all patients, as only affordable patients can use good quality of treatment. . The major problem faced by the CKD affected persons was not diagnosed until final stages.

2.2 Proposed Solution

With the use of Machine Learning Model, there will be no limitation of the complexity increasing number of variables. This Model and train and test the given factors which causes Chronic Kidney Disease and with the best performing machine learning model it can effortlessly predict the result of the patient with much higher accuracy than traditional methods.

**3. Theoretical Analysis**

3.1 Block Diagram



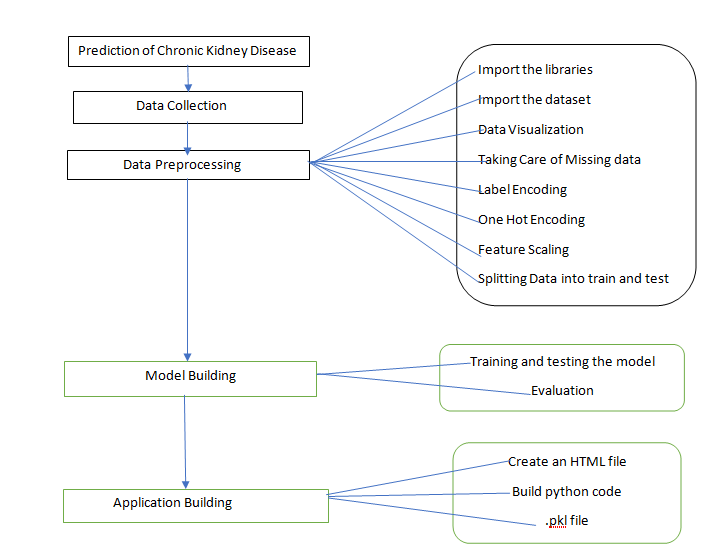
3.2 Hardware/Software Designing:

Python, Python Web Frame Works, Python for Data Analysis, Python for Data Visualization, Data Pre-processing Techniques, Machine Learning, Regression Algorithms.

**4. Experimental Investigations:**

In our project, we have used the chronic kidney disease dataset. The data that is used in this project originally comes from the kaggle machine learning dataset. We got to know all the required parameters to predict whether the person is suffering from Chronic Kidney Disease or not and we also analysed different models and concluded the best model for predicting the output.

**5. Flowchart :**

****

**6. Result:**

We have used KNN and Decision Tree Classifier and Random Forest Classifier to make predictions and compared their performance. Decision Tree Classifier has highest accuracy and is a good choice for this problem. Decision Tree Classifier trains the model with subsets of data sampled from the training data, this will make our model more accurate.

In this project, the DTC algorithm is used to predict its performance. The results show 100% accuracy.

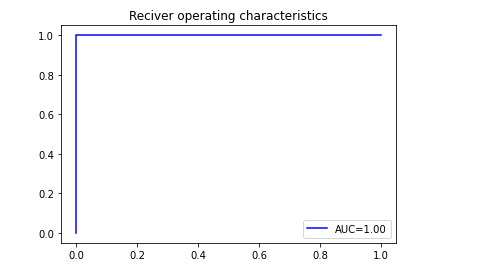
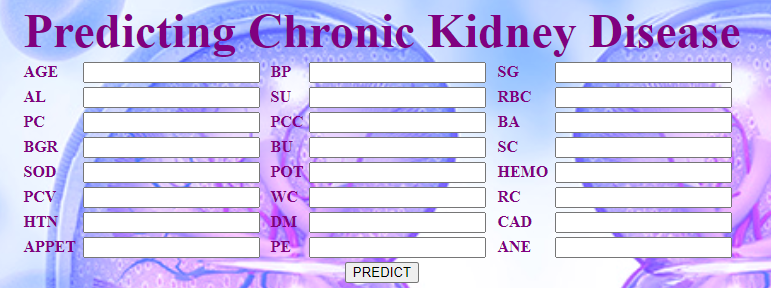


FIG: AUC – ROC Curve

**Snapshots:**



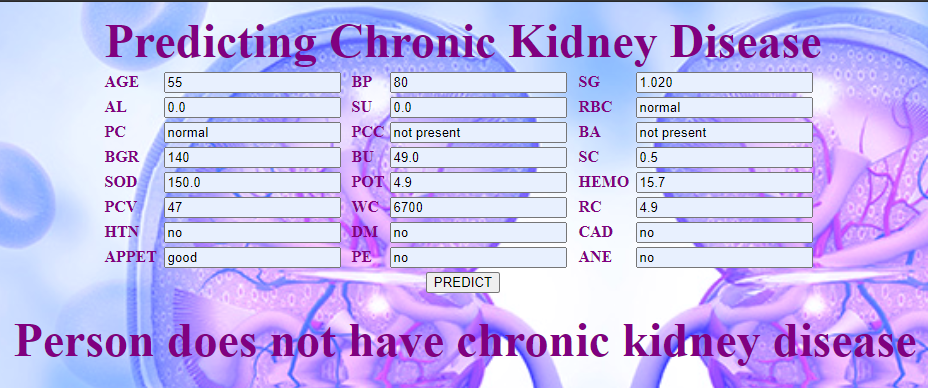


FIG- Negative Classification Example : indicates that the patient doesn’t has Chronic Kidney Disease.

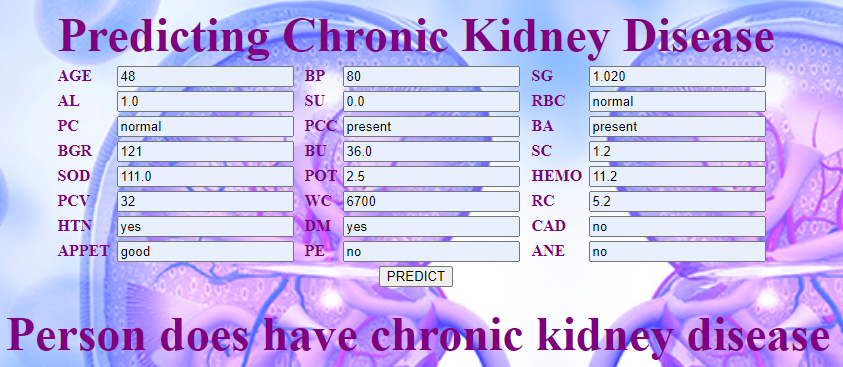


Fig: - Positive Classification Example : indicates that the patient has Chronic Kidney Disease.

**7. Advantages & disadvantages:**

**Advantages:**

The benefits of this model are:

* Easy interface
* Straight forward results
* Accurate performance calculations

**Disadvantages:**

1. A small change in the data can cause a large change in the structure of the decision tree causing instability.

2. Decision tree often involves higher time to train the model.

**8. APPLICATIONS:**

* Chronic Kidney Disease analysis using Decision Tree Classification algorithm simplifies the management process of check-up by deploying a web interface to the users.
* Fast processing and immediate results with high accuracy.
* Minimizing human effort and cost efficient databases.

**9. Conclusion:**

* Best alternative to state whether the patient has CKD positive or negative.
* Further improvement is required.

**10. Future Scope:**

Database should be expanded on which the system will be tested much better. Also, the model requires further improvement mostly regarding feature selection of the chronic kidney disease into multiple components.

**11. Biblography:-**

<https://www.kaggle.com/mansoordaku/ckdisease?select=kidney_disease.csv>

-Data from kidney\_disease.csv dataset.

**12. APPENDIX:**

1. Source Code for Data Preprocessing:-

# **import the libraries**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# **Reading the dataset**

df=pd.read\_csv("kidney\_disease.csv")

#**Taking Care of Missing Data**

df.fillna(df.mean(),inplace=True)

df['rbc']=df['rbc'].fillna(value=df['rbc'].mode().iloc[0])

df['pc']=df['pc'].fillna(value=df['pc'].mode().iloc[0])

df['htn']=df['htn'].fillna(value=df['htn'].mode().iloc[0])

df['ba']=df['ba'].fillna(value=df['ba'].mode().iloc[0])

df['pcc']=df['pcc'].fillna(value=df['pcc'].mode().iloc[0])

df['appet']=df['appet'].fillna(value=df['appet'].mode().iloc[0])

df['ane']=df['ane'].fillna(value=df['ane'].mode().iloc[0])

df['dm']=df['dm'].fillna(value=df['dm'].mode().iloc[0])

df['cad']=df['cad'].fillna(value=df['cad'].mode().iloc[0])

df['pe']=df['pe'].fillna(value=df['pe'].mode().iloc[0])

df['pcv']=df['pcv'].fillna(value=df['pcv'].mode().iloc[0])

df['wc']=df['wc'].fillna(value=df['wc'].mode().iloc[0])

df['rc']=df['rc'].fillna(value=df['rc'].mode().iloc[0])

df['wc'].replace("\t?", df['wc'].mode()[0], inplace=True)

df['rc'].replace("\t?", df['rc'].mode()[0], inplace=True)

df['pcv'].replace("\t?", df['pcv'].mode()[0], inplace=True)

#**Label Encoder**

from sklearn.preprocessing import LabelEncoder

lb=LabelEncoder()

l=[5,6,7,8,17,18,19,20,21,22,23]

for i in l:

x[:,i]=lb.fit\_transform(x[:,i])

x[0,:]

y=lb.fit\_transform(y)

y

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

x=sc.fit\_transform(x)

x

#**Data Visualization**

sns.countplot(df.classification)

plt.xlabel('Chronic Kidney Disease')

plt.title("Classification",fontsize=15)

plt.show()

plt.figure(figsize = (19,19))

sns.heatmap(df.corr(), annot = True, cmap = 'coolwarm')

1. **Source Code for Data Tree Classifier:-**

from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier(criterion='entropy', random\_state=0)

dtc.fit(x\_train,y\_train)

ydtc = dtc.predict(x\_test)

accuracy\_score(y\_test, ydtc)

cm=confusion\_matrix(y\_test,ydtc)

f,ax=plt.subplots(figsize=(5,5))

sns.heatmap(cm,annot=True,linewidths=0.5,linecolor="red",fmt=".0f",ax=ax)

plt.xlabel("y\_pred")

plt.ylabel("y\_test")

plt.title("DecisionTreeClassifier")

plt.show()

1. **Source code for HTML file:-**

<html>

<head>

<style>

body

{

background-image:url("https://drugdiscoverytrends.com/wp-content/uploads/2019/10/190416-kidney.jpg");

background-blend-mode: lighten;

height: 500px;

background-position: center;

background-repeat: no-repeat;

background-size: cover;

position: relative;

}

</style>

</head>

<body bgcolor = "blue" text = "purple">

<form action="{{ url\_for('y\_predict')}}" method = "POST">

<table border = "0" align = "center">

<tr>

<th colspan = "6" align = "center"><b><font color = "purple" size = "20"> Predicting Chronic Kidney Disease</font></b></th>

</tr>

<tr>

<td><b> AGE</b></td>

<td><input type = "text" name = "AGE"/></td>

<td><b> BP</b></td>

<td><input type = "text" name = "BP"/></td>

<td><b> SG</b></td>

<td><input type = "text" name = "SG"/></td>

</tr>

<tr>

<td><b> AL</b></td>

<td><input type = "text" name = "AL"/></td>

<td><b>SU</b></td>

<td><input type = "text" name = "SU"/></td>

<td> <b>RBC</b></td>

<td><input type = "text" name = "RBC"/></td>

</tr>

<tr>

<td><b> PC</b></td>

<td><input type = "text" name = "PC"/></td>

<td><b> PCC</b></td>

<td><input type = "text" name = "PCC"/></td>

<td><b> BA</b></td>

<td><input type = "text" name = "BA"/></td>

</tr>

<tr>

<td><b> BGR</b></td>

<td><input type = "text" name = "BGR"/></td>

<td><b> BU</b></td>

<td><input type = "text" name = "BU"/></td>

<td> <b>SC</b></td>

<td><input type = "text" name = "SC"/></td>

</tr>

<tr>

<td><b> SOD</b></td>

<td><input type = "text" name = "SOD"/></td>

<td><b> POT</b></td>

<td><input type = "text" name = "POT"/></td>

<td><b> HEMO</b></td>

<td><input type = "text" name = "HEMO"/></td>

</tr>

<tr>

<td><b> PCV</b></td>

<td><input type = "text" name = "PCV"/></td>

<td><b> WC</b></td>

<td><input type = "text" name = "WC"/></td>

<td><b> RC</b></td>

<td><input type = "text" name = "RC"/></td>

</tr>

<tr>

<td><b> HTN</b></td>

<td><input type = "text" name = "HTN"/></td>

<td><b> DM</b></td>

<td><input type = "text" name = "DM"/></td>

<td><b> CAD</b></td>

<td><input type = "text" name = "CAD"/></td>

</tr>

<tr>

<td> <b>APPET</b></td>

<td><input type = "text" name = "APPET"/></td>

<td><b>PE</b></td>

<td><input type = "text" name = "PE"/></td>

<td><b>ANE</b></td>

<td><input type = "text" name = "ANE"/></td>

</tr>

<tr>

<th colspan = 6"><input type = "submit" value= "PREDICT"/><th>

</tr>

</table>

</form>

<center><b><font size=30>{{ prediction\_text }}</font></b></center>

</body>

</html>

1. **Source code for Flask application:-**

import numpy as np

from joblib import load

from flask import Flask, request, jsonify, render\_template

import pickle

app = Flask(\_\_name\_\_)

model=load('decision.pkl')

@app.route('/')

def home():

return render\_template('Chronic\_Kidney.html')

@app.route('/y\_predict',methods=['POST'])

def y\_predict():

'''

For rendering results on HTML GUI

'''

x\_test = [x for x in request.form.values()]

print(x\_test);

if(x\_test[5]=='abnormal'):

x\_test[5]=0

else:

x\_test[5]=1

if(x\_test[6]=='abnormal'):

x\_test[6]=0

else:

x\_test[6]=1

if(x\_test[7]=='notpresent'):

x\_test[7]=0

else:

x\_test[7]=1

if(x\_test[8]=='notpresent'):

x\_test[8]=0

else:

x\_test[8]=1

if(x\_test[18]=='no'):

x\_test[18]=0

else:

x\_test[18]=1

if(x\_test[19]=='no'):

x\_test[19]=3

else:

x\_test[19]=4

if(x\_test[20]=='no'):

x\_test[20]=1

else:

x\_test[20]=2

if(x\_test[21]=='good'):

x\_test[21]=0

else:

x\_test[21]=1

if(x\_test[22]=='no'):

x\_test[22]=0

else:

x\_test[22]=1

if(x\_test[23]=='no'):

x\_test[23]=0

else:

x\_test[23]=1

for i in range(len(x\_test)):

x\_test[i]=float(x\_test[i])

print(x\_test)

l=np.array(x\_test,dtype=object)

l=l.reshape(1,-1)

prediction = model.predict(l)

print(prediction)

d={0:'have chronic kidney disease',1:'not have chronic kidney disease'}

k=prediction.tolist()

for i in k:

output=d[i]

print(output)

return render\_template('Chronic\_Kidney.html', prediction\_text='Person does {}'.format(output))

@app.route('/predict\_api',methods=['POST'])

def predict\_api():

'''

For direct API calls trought request

'''

data = request.get\_json(force=True)

prediction = model.y\_predict([np.array(list(data.values()))])

output = prediction[0]

return jsonify(output)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)