Project Name:	Project - Early Detection of Chronic Kidney Disease using Machine Learning
Team ID:	PNT2022TMID13778

FINAL CODE PDF

Collecting , Visualizing, and Preprocessing the Dataset

1.Importing the packages

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from collections import Counter as c

import seaborn as sns

import missingno as msng

from sklearn.metrics import accuracy_score,confusion_matrix

from sklearn.model_selection import train_test_split

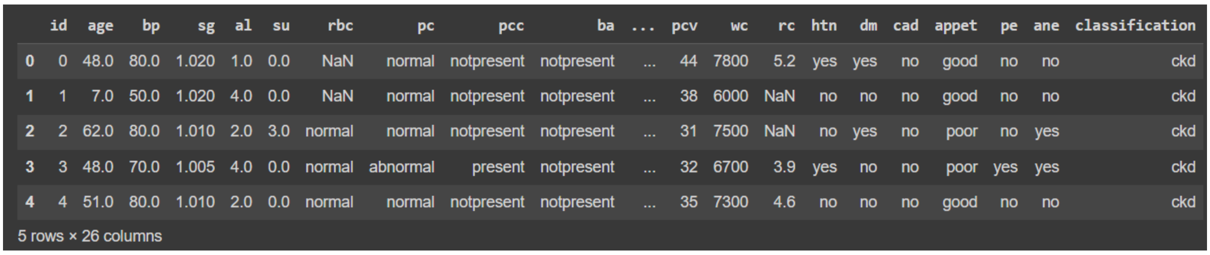
from sklearn.preprocessing import LabelEncoder

from sklearn.linear_model import LogisticRegression

#Data Collections

data=pd.read_csv("/content/drive/MyDrive/chronickidneydisease.csv")

data.head()



```
   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 rows × 26 columns
```

data.drop(['id'],axis=1,inplace=True)

data.columns

data.columns=['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu',

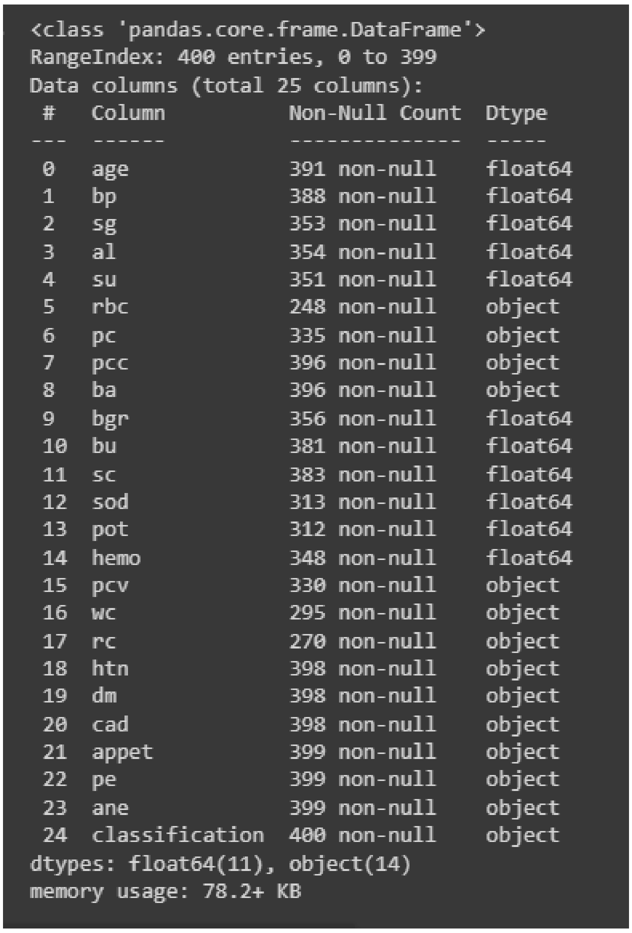
'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad',

'appet', 'pe', 'ane', 'classification']

data.columns

data['classification'].unique()

data.info()



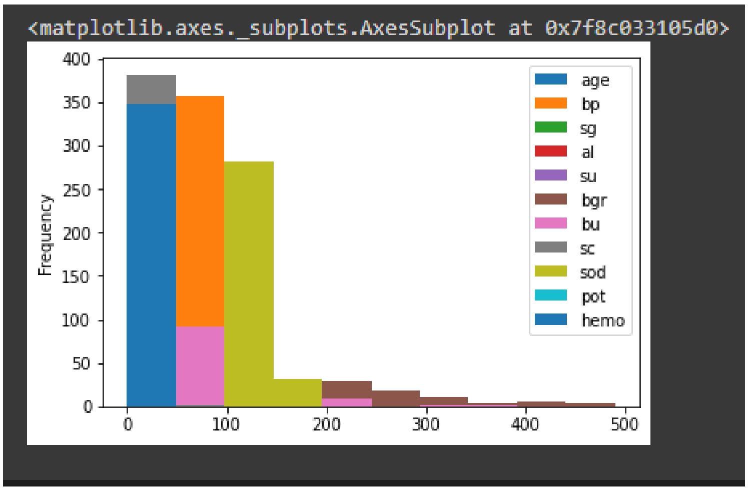
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             391 non-null    float64
 1   bp              388 non-null    float64
 2   sg              353 non-null    float64
 3   al              354 non-null    float64
 4   su              351 non-null    float64
 5   rbc             248 non-null    object 
 6   pc              335 non-null    object 
 7   pcc             396 non-null    object 
 8   ba              396 non-null    object 
 9   bgr             356 non-null    float64
 10  bu              381 non-null    float64
 11  sc              383 non-null    float64
 12  sod             313 non-null    float64
 13  pot             312 non-null    float64
 14  hemo            348 non-null    float64
 15  pcv             330 non-null    object 
 16  wc              295 non-null    object 
 17  rc              270 non-null    object 
 18  htn             398 non-null    object 
 19  dm              398 non-null    object 
 20  cad             398 non-null    object 
 21  appet           399 non-null    object 
 22  pe              399 non-null    object 
 23  ane             399 non-null    object 
 24  classification  400 non-null    object 
dtypes: float64(11), object(14)
memory usage: 78.2+ KB
```

2. Data visualization

from matplotlib import pyplot

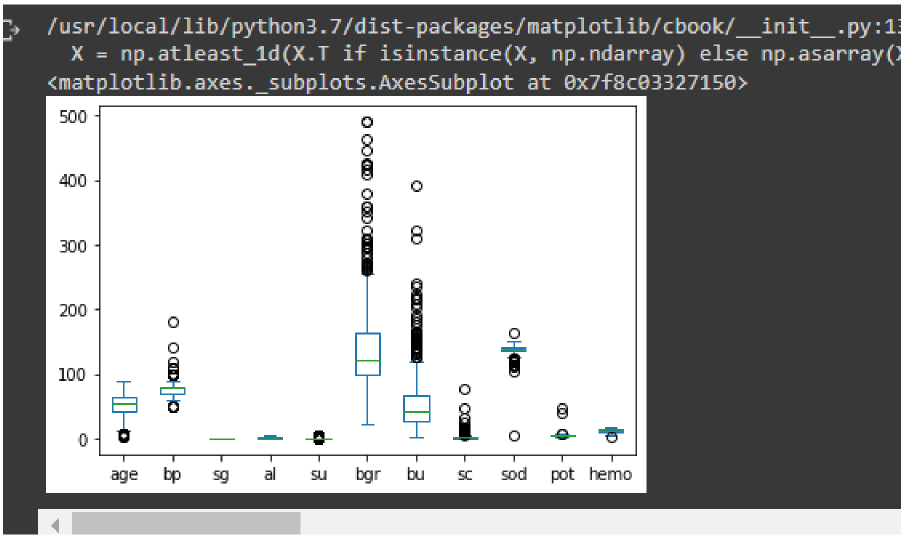
data.plot

data.plot.hist()



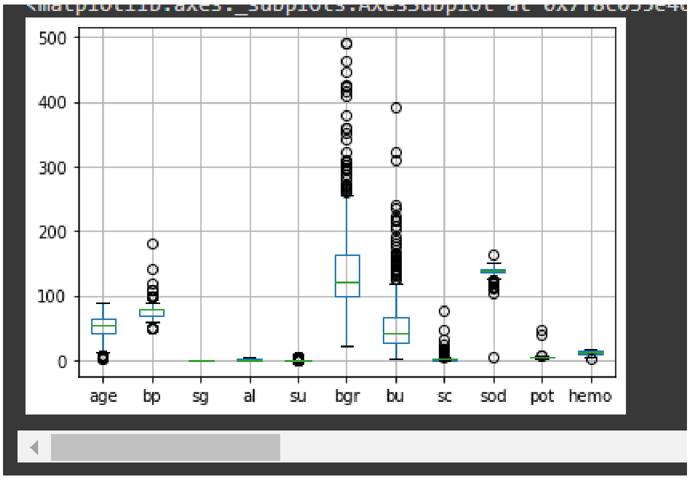
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8c033105d0>
```

data.plot.box()



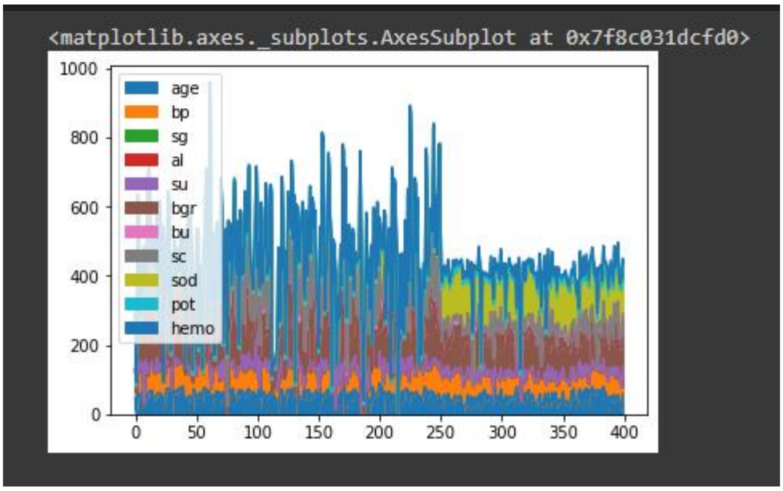
```
/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/__init__.py:13
  X = np.atleast_1d(X.T if isinstance(X, np.ndarray) else np.asarray(X
<matplotlib.axes._subplots.AxesSubplot at 0x7f8c03327150>
```

data.boxplot()



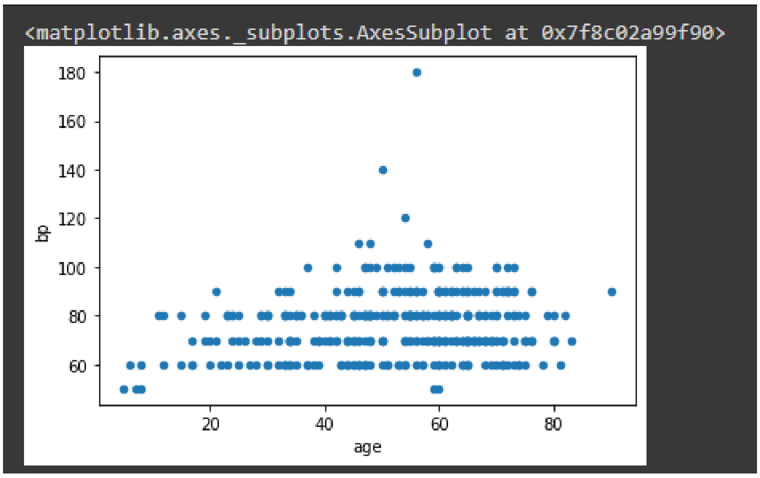
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8c035e40
```

data.plot.area()



```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8c031dcfd0>
```

data.plot.scatter(x='age',y='bp')

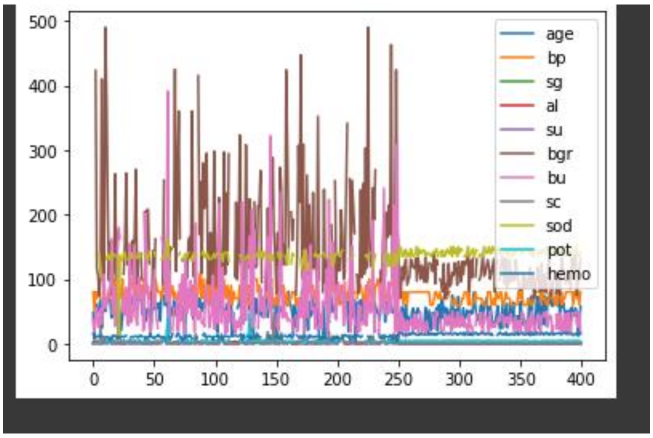


```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8c02a99f90>
```

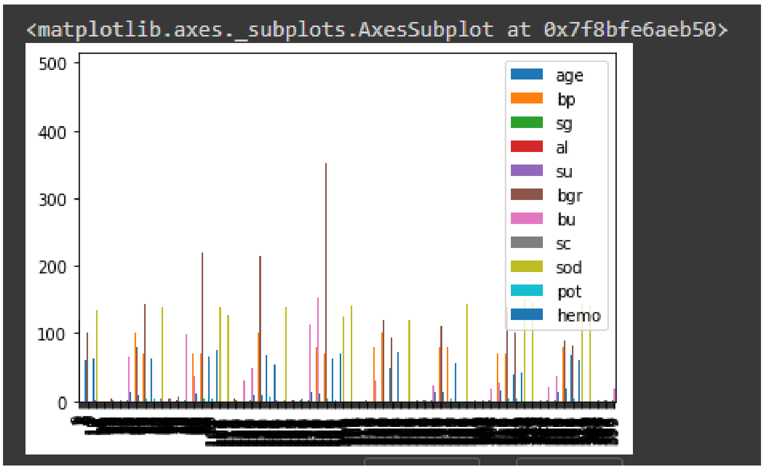
pie = data

pie

pie.plot();



data.plot.bar()



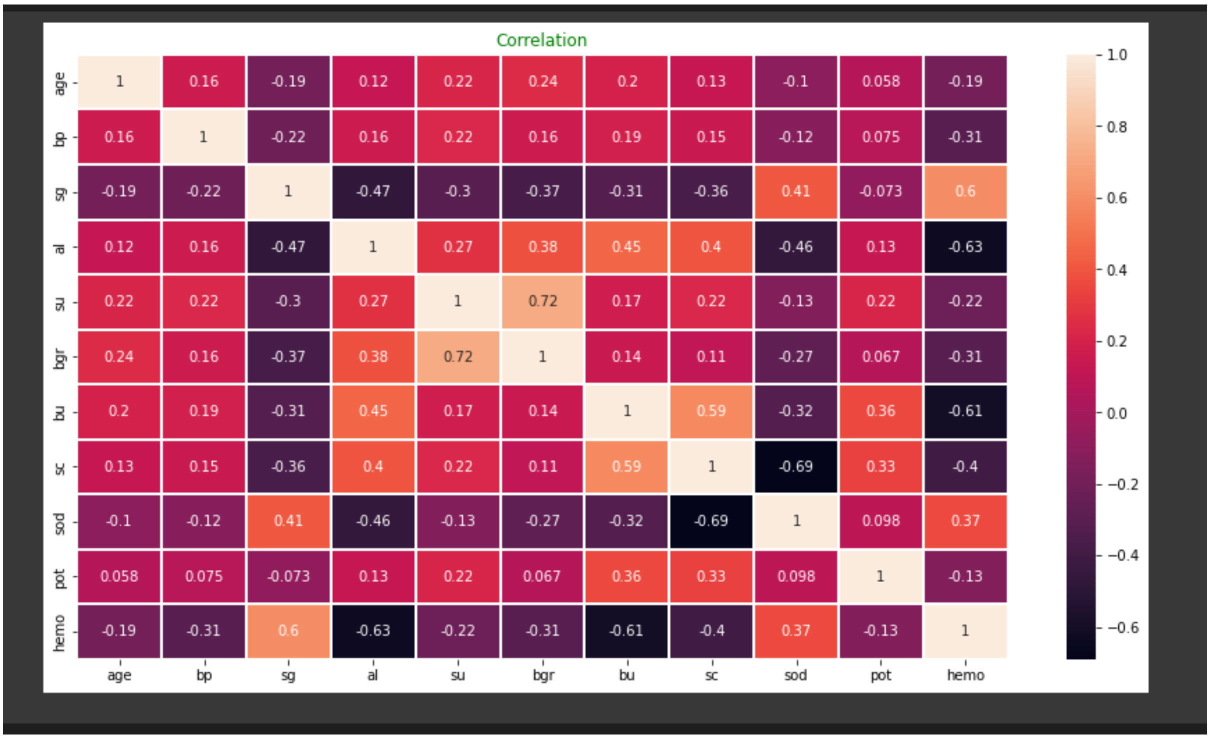
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8bfe6aeb50>
```

data.corr()

plt.figure(figsize=(15,8));

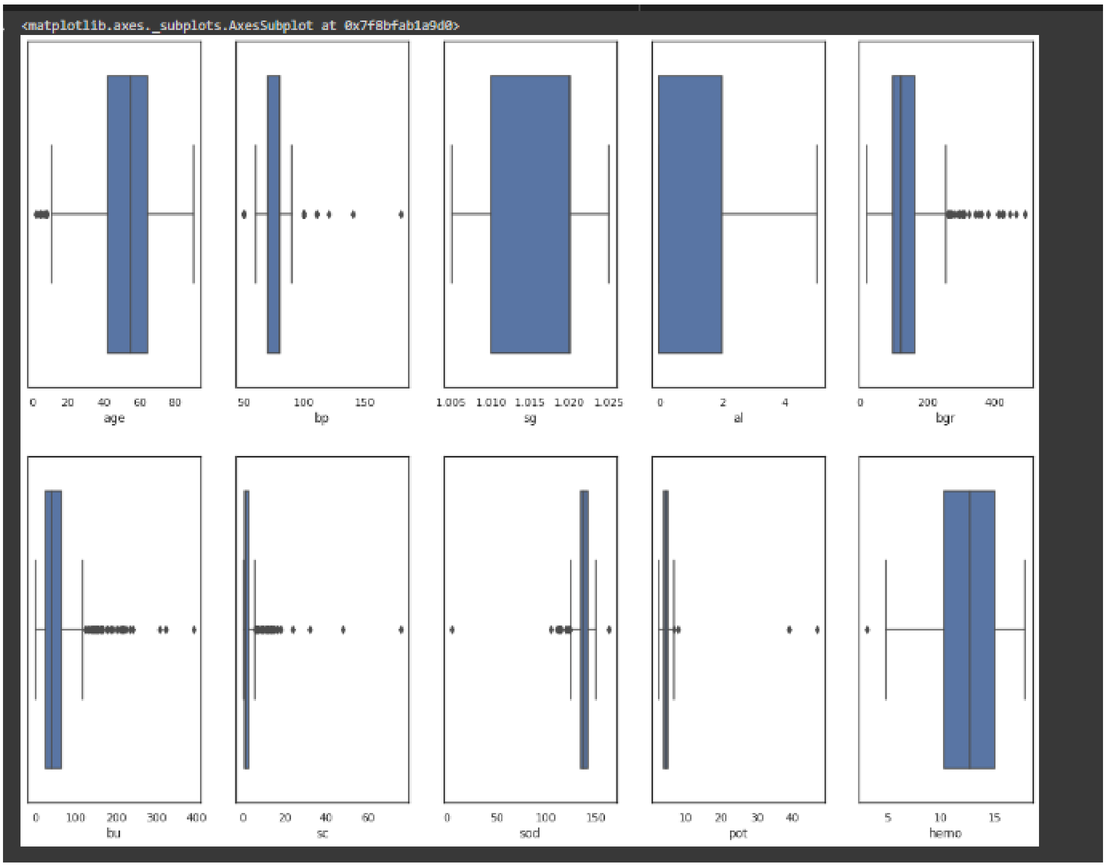
plt.title("Correlation",color="green")

```
sns.heatmap(data.corr(),linewidth=1,annot=True);
```



	age	bp	sg	al	su	bgr	bu	sc	sod	pot	hemo
age	1	0.16	-0.19	0.12	0.22	0.24	0.2	0.13	-0.1	0.058	-0.19
bp	0.16	1	-0.22	0.16	0.22	0.16	0.19	0.15	-0.12	0.075	-0.31
sg	-0.19	-0.22	1	-0.47	-0.3	-0.37	-0.31	-0.36	0.41	-0.073	0.6
al	0.12	0.16	-0.47	1	0.27	0.38	0.45	0.4	-0.46	0.13	-0.63
su	0.22	0.22	-0.3	0.27	1	0.72	0.17	0.22	-0.13	0.22	-0.22
bgr	0.24	0.16	-0.37	0.38	0.72	1	0.14	0.11	-0.27	0.067	-0.31
bu	0.2	0.19	-0.31	0.45	0.17	0.14	1	0.59	-0.32	0.36	-0.61
sc	0.13	0.15	-0.36	0.4	0.22	0.11	0.59	1	-0.69	0.33	-0.4
sod	-0.1	-0.12	0.41	-0.46	-0.13	-0.27	-0.32	-0.69	1	0.098	0.37
pot	0.058	0.075	-0.073	0.13	0.22	0.067	0.36	0.33	0.098	1	-0.13
hemo	-0.19	-0.31	0.6	-0.63	-0.22	-0.31	-0.61	-0.4	0.37	-0.13	1

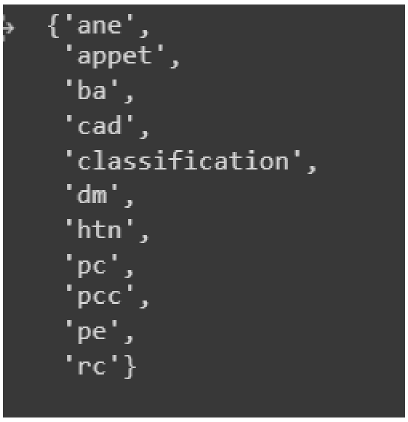
```
sns.set_theme(style="white")

fig, ((ax1, ax2,ax3,ax4,ax5), (ax6, ax7,ax8,ax9,ax10))= plt.subplots(nrows=2, ncols=5, figsize=(18,14))

sns.boxplot(data=data,x="age",ax=ax1)

sns.boxplot(data=data,x="bp",ax=ax2)

sns.boxplot(data=data,x="sg",ax=ax3)

sns.boxplot(data=data,x="al",ax=ax4)

sns.boxplot(data=data,x="bgr",ax=ax5)

sns.boxplot(data=data,x="bu",ax=ax6)

sns.boxplot(data=data,x="sc",ax=ax7)

sns.boxplot(data=data,x="sod",ax=ax8)

sns.boxplot(data=data,x="pot",ax=ax9)

sns.boxplot(data=data,x="hemo",ax=ax10)
```



```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8bfab1a9d0>
```

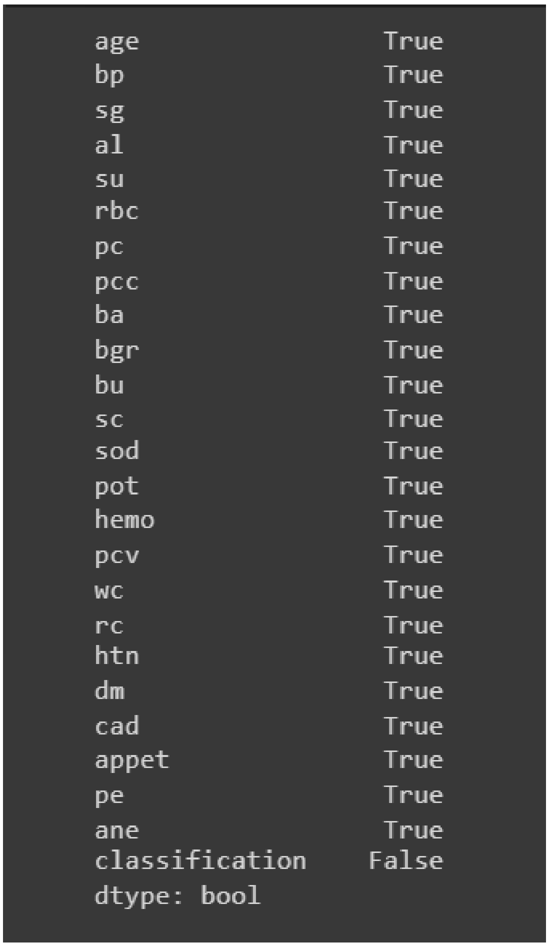
3. Data Preprocessing

```
data['classification']=data['classification'].replace("ckd\t",'ckd')

catcols=set(data.dtypes[data.dtypes=='O'].index.values)

print(catcols)

for i in catcols:

   print("columns:",i)

   print(c(data[i]))

   print('*'*120+'\n')

catcols.remove('rbc')

catcols.remove('pcv')

catcols.remove('wc')

catcols
```



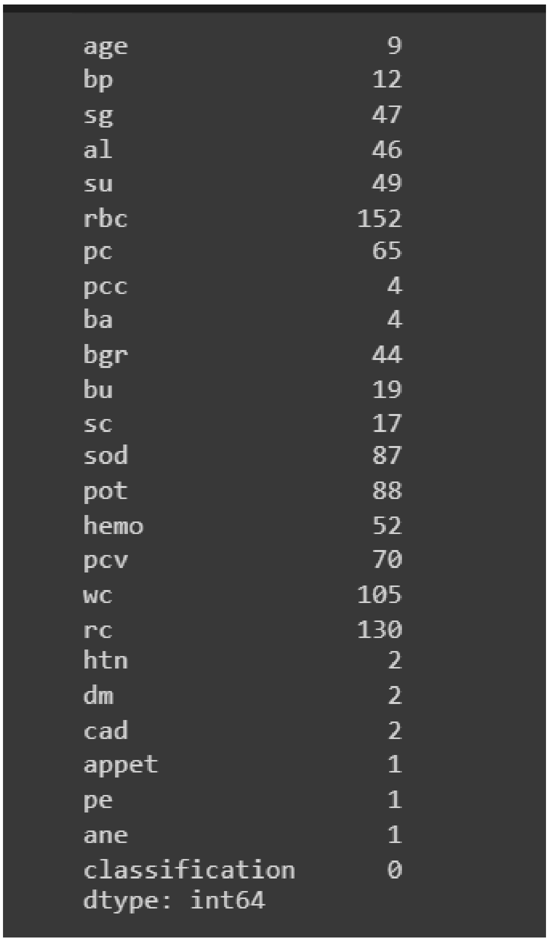
```
{'ane',
 'appet',
 'ba',
 'cad',
 'classification',
 'dm',
 'htn',
 'pc',
 'pcc',
 'pe',
 'rc'}
```

```
contcols=set(data.dtypes[data.dtypes!='O'].index.values)

contcols

for i in catcols:

   print("continuous columns :",i)

   print(c(data[i]))

   print('*'*120+'\n')

contcols.remove('sg')

contcols.remove('al')

contcols.remove('su')

print(contcols)

contcols.add('rbc')

contcols.add('pc')

contcols.add('wc')

print(contcols)

catcols.add('sg')

catcols.add('al')

catcols.add('su')

print(catcols)

data['cad']=data.cad.replace('\tno','no')

c(data['cad'])

data['dm']=data.dm.replace(to_replace={'\tno':'no','\tyes':'yes',' yes':'yes'})

c(data['dm'])

data.isna().any()
```



```
age               True
bp                True
sg                True
al                True
su                True
rbc               True
pc                True
pcc               True
ba                True
bgr               True
bu                True
sc                True
sod               True
pot               True
hemo              True
pcv               True
wc                True
rc                True
htn               True
dm                True
cad               True
appet             True
pe                True
ane               True
classification   False
dtype: bool
```

```
data.isna().sum()
```



```
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
```

```
data.pcv=pd.to_numeric(data.pcv,errors='coerce')

data.wc=pd.to_numeric(data.wc,errors='coerce')

data.rc=pd.to_numeric(data.rc,errors='coerce')

data['bgr'].fillna(data['bgr'].mean(),inplace=True)

data['bp'].fillna(data['bp'].mean(),inplace=True)

data['bu'].fillna(data['bu'].mean(),inplace=True)

data['hemo'].fillna(data['hemo'].mean(),inplace=True)

data['pcv'].fillna(data['pcv'].mean(),inplace=True)

data['pot'].fillna(data['pot'].mean(),inplace=True)

data['rc'].fillna(data['rc'].mean(),inplace=True)

data['sc'].fillna(data['sc'].mean(),inplace=True)

data['sod'].fillna(data['sod'].mean(),inplace=True)

data['wc'].fillna(data['wc'].mean(),inplace=True)

data['age'].fillna(data['age'].mode()[0],inplace=True)

data['htn'].fillna(data['htn'].mode()[0],inplace=True)
```

```
data['pcc'].fillna(data['pcc'].mode()[0],inplace=True)
data['appet'].fillna(data['appet'].mode()[0],inplace=True)
data['al'].fillna(data['al'].mode()[0],inplace=True)
data['pc'].fillna(data['pc'].mode()[0],inplace=True)
data['rbc'].fillna(data['rbc'].mode()[0],inplace=True)
data['cad'].fillna(data['cad'].mode()[0],inplace=True)
data['ba'].fillna(data['ba'].mode()[0],inplace=True)
data['ane'].fillna(data['ane'].mode()[0],inplace=True)
data['su'].fillna(data['su'].mode()[0],inplace=True)
data['dm'].fillna(data['dm'].mode()[0],inplace=True)
data['pe'].fillna(data['pe'].mode()[0],inplace=True)
data['sg'].fillna(data['sg'].mode()[0],inplace=True)
```

ML MODEL CREATION

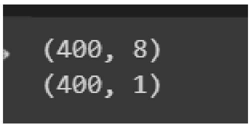
Importing the packages

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msng
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
```

1.splitting the dataset

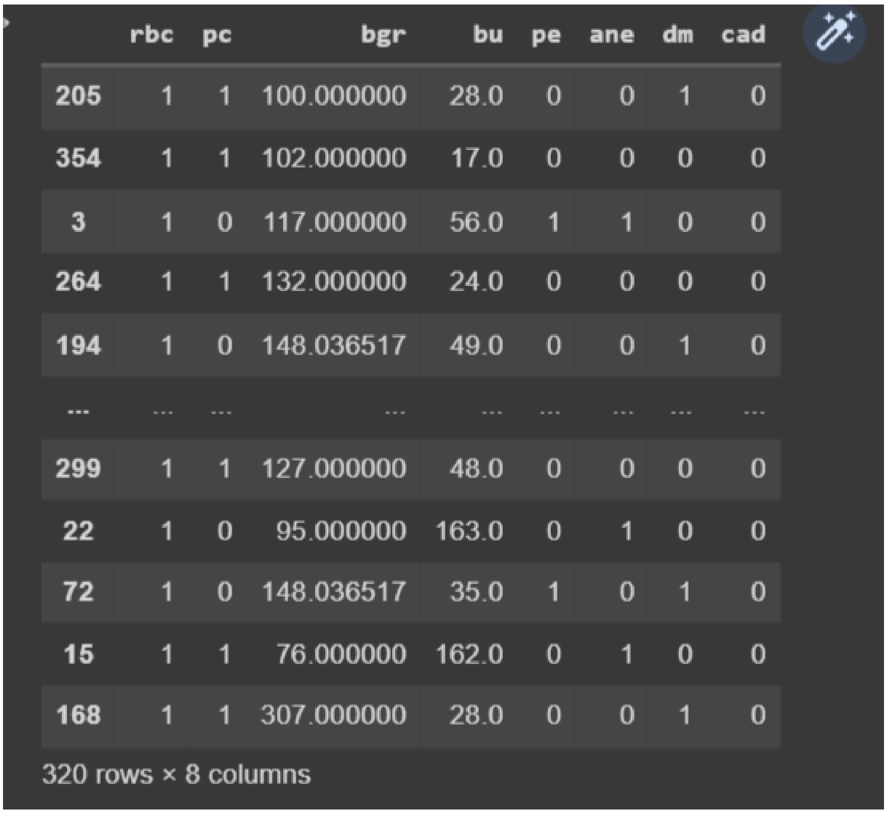
```
for i in catcols:
   print("LABEL ENCODING OF :",i)
   le=LabelEncoder()
   print(c(data[i]))
   data[i]=le.fit_transform(data[i])
   print(c(data[i]))
```

```
    print('*'*100)

data['rbc']=le.fit_transform(data['rbc'])
selcols=['rbc','pc','bgr','bu','pe','ane','dm','cad']
x=pd.DataFrame(data,columns=selcols)
y=pd.DataFrame(data,columns=['classification'])

print(x.shape)
print(y.shape)
```



```
(400, 8)
(400, 1)
```

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=2)
xtrain
```



	rbc	pc	bgr	bu	pe	ane	dm	cad
205	1	1	100.000000	28.0	0	0	1	0
354	1	1	102.000000	17.0	0	0	0	0
3	1	0	117.000000	56.0	1	1	0	0
264	1	1	132.000000	24.0	0	0	0	0
194	1	0	148.036517	49.0	0	0	1	0
...	...	...	...	...	...	...	...	...
299	1	1	127.000000	48.0	0	0	0	0
22	1	0	95.000000	163.0	0	1	0	0
72	1	0	148.036517	35.0	1	0	1	0
15	1	1	76.000000	162.0	0	1	0	0
168	1	1	307.000000	28.0	0	0	1	0

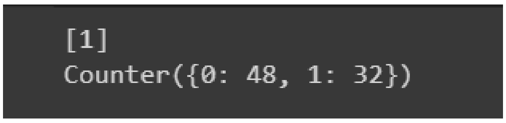
320 rows × 8 columns

2. Model creation

```
lgr=LogisticRegression()
lgr.fit(xtrain.values,ytrain.values)
```

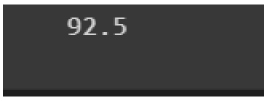
```
ypred=lgr.predict(xtest)

ypred1=lgr.predict([[129,99,1,0,0,1,0,1]])

print(ypred1)

c(ypred)
```

3. Accuracy , Confustion Matrix , Classification Report



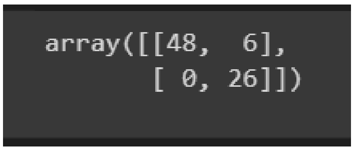
```
[1]
Counter({0: 48, 1: 32})
```

```
print(accuracy_score(ytest,ypred)*100)
```



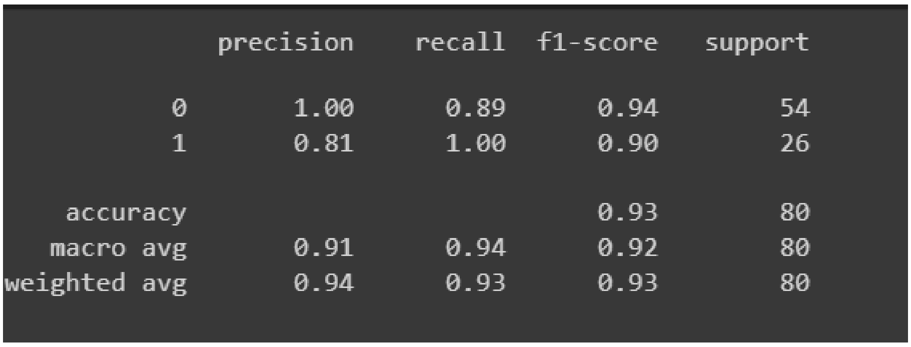
```
92.5
```

```
confmat=confusion_matrix(ytest,ypred)

confmat
```



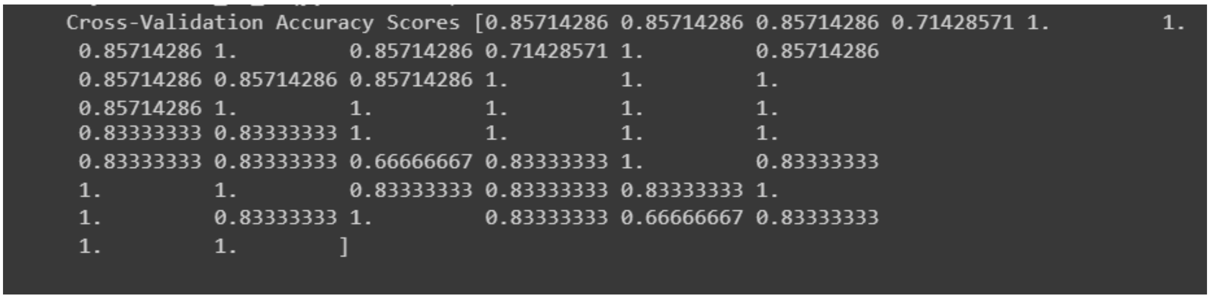
```
array([[48,  6],
       [ 0, 26]])
```

```
from sklearn.metrics import classification_report

print(classification_report(ytest, ypred))
```



```
              precision    recall  f1-score   support

           0       1.00      0.89      0.94        54
           1       0.81      1.00      0.90        26

    accuracy                           0.93        80
   macro avg       0.91      0.94      0.92        80
weighted avg       0.94      0.93      0.93        80
```

```
from sklearn.model_selection import cross_val_score


scores = cross_val_score(lgr, xtrain, ytrain, cv=50)

print('Cross-Validation Accuracy Scores', scores)
```



```
Cross-Validation Accuracy Scores [0.85714286 0.85714286 0.85714286 0.71428571 1.         1.
 0.85714286 1.         0.85714286 0.71428571 1.         0.85714286
 0.85714286 0.85714286 0.85714286 1.         1.         1.
 0.85714286 1.         1.         1.         1.         1.
 0.83333333 0.83333333 1.         1.         1.         1.
 0.83333333 0.83333333 0.66666667 0.83333333 1.         0.83333333
 1.         1.         0.83333333 0.83333333 0.83333333 1.
 1.         0.83333333 1.         0.83333333 0.66666667 0.83333333
 1.         1.        ]
```

1.FrontEnd Development

Frontend consists of 3 pages

1. Index page
2. Prediction page
3. Output page

Technology used in Frontend

HTML

CSS

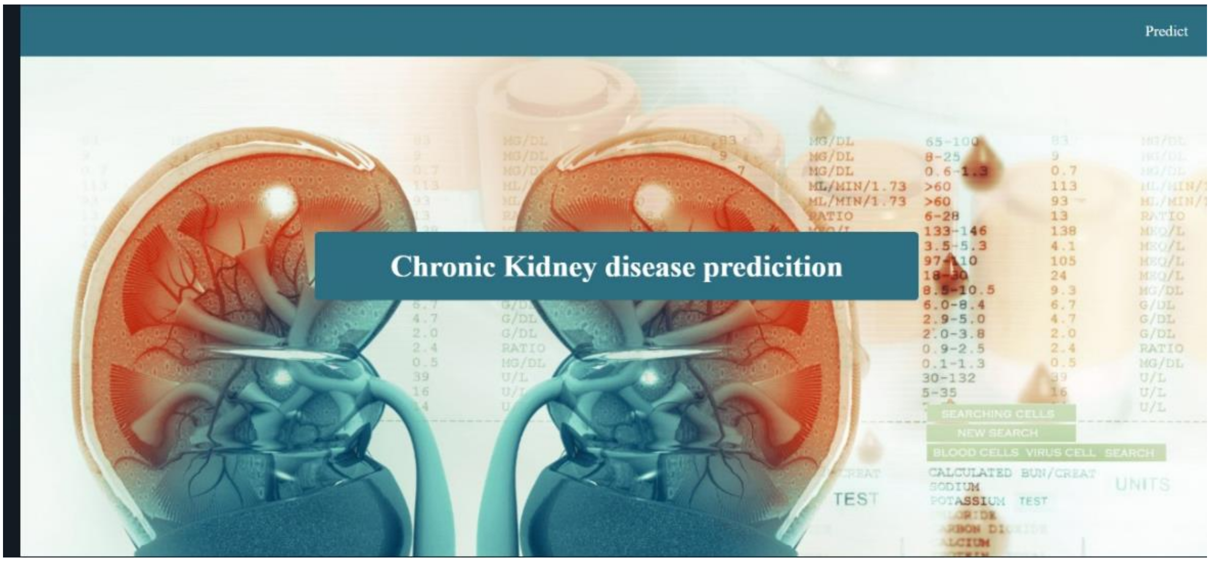
JS

1.Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <style>
      * {
        padding: 0;
        margin: 0;
      }

      .background {
        background-image: url("/images/bg4.jpeg");
        background-repeat: no-repeat;
        background-size: cover;
      }
      .header {
        display: flex;
        flex: 100%;
        flex-direction: row;
        justify-content: flex-end;
        height: 60px;
        padding-right: 30px;
        background-color: #2e6e82;
        align-items: center;
      }
     .btn{
        color: #fff;
        font-size: large;
        text-decoration: none;
     }
     .titleWrapper{
        height: 500px;
        display: flex;
        justify-content: center;
        align-items: center;
     }
     .title{
        background-color: #2e6e82;
        border-radius: 5px;
        padding: 20px 90px;
     }
    </style>
  </head>
```

```
  <body class="background">
    <div class="header">
      <a href="inputs.html" class="btn"> Predict
        </button>
    </div>
    <div class="titleWrapper">
        <h1 class="title">Chronic Kidney disease predicition</h1>
    </div>
  </body>
</html>
```

Index page



Predict

Chronic Kidney disease predicition

2.Prediction Page

Inputs.html

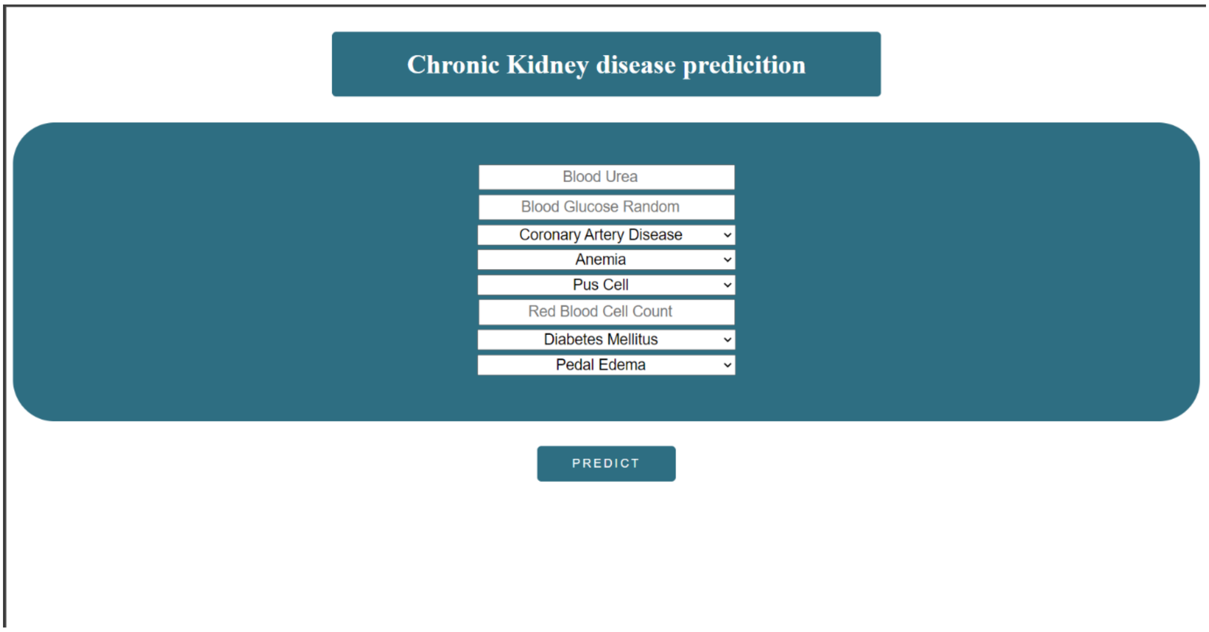
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
```

```
<style>
  .header {
    display: flex;
    justify-content: center;
    align-items: center;
  }
  .title {
    background-color: #2e6e82;
    border-radius: 5px;
    padding: 20px 90px;
    color: white;
  }
  .inputs {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    margin-top: 10px;
    background-color: #2e6e82;
    padding: 50px;
    border-radius: 50px;
  }
  input {
    width: 300px;
    height: 25px;
    text-align: center;
    margin-bottom: 5px;
    font-size: large;
  }
  select {
    width: 310px;
  height: 25px;
  text-align: center;
  margin-bottom: 5px;
  font-size: large;
}
.btn {
  display: flex;
  justify-content: center;
  align-items: center;
  margin-top: 30px;
}
button {
  position: relative;
  font-size: 14px;
  letter-spacing: 3px;
  height: 3em;
  padding: 0 3em;
  border: none;
  background-color: #2e6e82;
  color: #fff;
  text-transform: uppercase;
  overflow: hidden;
  border-radius: 5px;
}

button::before {
  content: "";
  display: block;
  position: absolute;
  z-index: 0;
  bottom: 0;
  left: 0;
  height: 0px;
  width: 100%;
  background: rgb(46, 110, 130);
  background: linear-gradient(
    90deg,
    rgba(46, 110, 130, 1) 20%,
    rgba(46, 110, 130, 1) 100%
  );
  transition: 0.2s;
}

button .label {
  position: relative;
}

button .icon {
  display: flex;
  align-items: center;
  justify-content: center;
  height: 3em;
  width: 3em;
  position: absolute;
  top: 3em;
  right: 0;
  opacity: 0;
  transition: 0.4s;
}

button:hover::before {
  height: 100%;
```

```
    }

    button:hover .icon {
      top: 0;
      opacity: 1;
    }
  </style>
</head>
<body>
  <div class="header">
    <h1 class="title">Chronic Kidney disease predicition</h1>
  </div>
  <div class="inputs">
    <input type="number" placeholder="Blood Urea" />
    <input type="number" placeholder="Blood Glucose Random" />
    <select name="Coronary Artery Disease" id="">
      <option value="Coronary Artery Disease">Coronary Artery Disease</option>
      <option value="yes">Yes</option>
      <option value="no">No</option>
    </select>
    <select name="anemia" id="">
      <option value="anemia">Anemia</option>
      <option value="yes">Yes</option>
      <option value="no">No</option>
    </select>
    <select name="pus cell" id="">
      <option value="pus cell">Pus Cell</option>
      <option value="normal">Normal</option>
      <option value="abnormal">Abnormal</option>
    </select>
    <input type="number" placeholder="Red Blood Cell Count" />
```

```html
      <select name="diabetes mellitus" id="">
        <option value="diabetes mellitus">Diabetes Mellitus</option>
        <option value="yes">Yes</option>
        <option value="no">No</option>
      </select>
      <select name="pedal edema" id="">
        <option value="pedal edema">Pedal Edema</option>
        <option value="yes">Yes</option>
        <option value="no">No</option>
      </select>
    </div>
    <div class="btn">
      <a href="results.html">
        <button>
          <span class="label">Predict</span>
          <span class="icon">
            <svg
              xmlns="http://www.w3.org/2000/svg"
              viewBox="0 0 24 24"
              width="24"
              height="24"
            >
              <path fill="none" d="M0 0h24v24H0z"></path>
              <path
                fill="currentColor"
                d="M16.172 11l-5.364-5.364 1.414-1.414L20 12l-7.778 7.778-1.414-1.414L16.172 13H4v-2z"
              ></path>
            </svg>
          </span>
        </button>
    </a>
   </div>
  </body>
</html>
```

Inputs page Or Prediction page



Chronic Kidney disease predicition

FrontEnd and Backend connection

1.FrontEnd Development

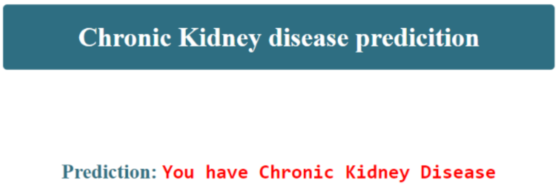
OUTPUT PAGE

Result.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
```

```
<style>
 .header {
   display: flex;
   justify-content: center;
   align-items: center;
 }
 .title {
   background-color: #2e6e82;
   border-radius: 5px;
   padding: 20px 90px;
   color: white;
 }
 .resultWrapper {
   display: flex;
   height: 200px;
   justify-content: center;
   align-items: center;
 }
 .result {
   border-radius: 10px;
   padding: 10px 30px;
 }
 .result-positive {
   color: red;
   font-size: larger;
 }
 .result-negative {
   color: blue;
   font-size: larger;
 }
 h2 {
    color: #2e6e82;
   }
  </style>
 </head>
 <body>
  <div class="header">
   <h1 class="title">Chronic Kidney disease predicition</h1>
  </div>
  <div class="resultWrapper">
   <div class="result">
    <h2>
     Prediction:
     <samp class="result-positive">You have Chronic Kidney Disease</samp>
    </h2>
    <!-- <h2>
     Prediction:
     <samp class="result-negative"> You Don't Chronic Kidney Disease</samp>
    </h2> -->
   </div>
  </div>
 </body>
</html>
```

Results Page

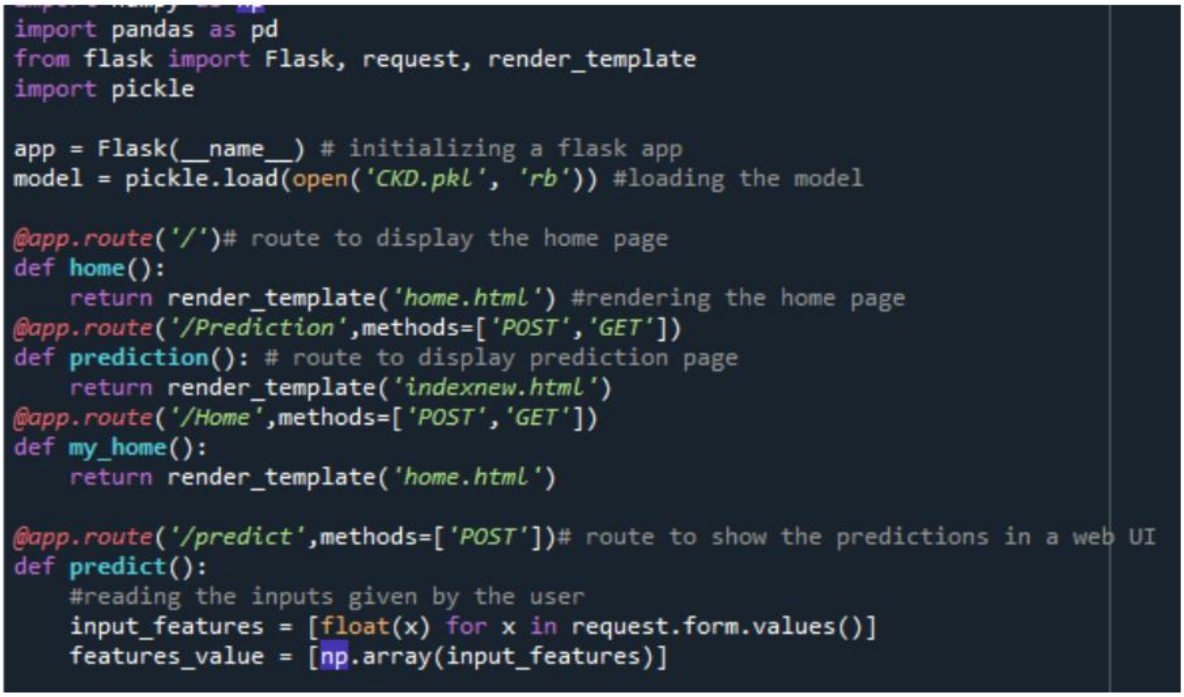


Chronic Kidney disease predicition

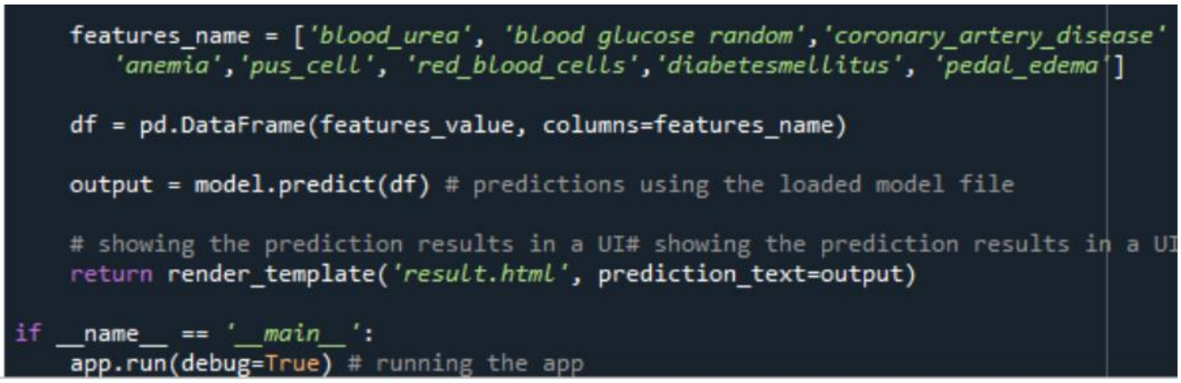
Prediction: You have Chronic Kidney Disease

2.Backend development

Flask



```
import numpy as np
import pandas as pd
from flask import Flask, request, render_template
import pickle

app = Flask(__name__) # initializing a flask app
model = pickle.load(open('CKD.pkl', 'rb')) #loading the model

@app.route('/')# route to display the home page
def home():
    return render_template('home.html') #rendering the home page
@app.route('/Prediction',methods=['POST','GET'])
def prediction(): # route to display prediction page
    return render_template('indexnew.html')
@app.route('/Home',methods=['POST','GET'])
def my_home():
    return render_template('home.html')

@app.route('/predict',methods=['POST'])# route to show the predictions in a web UI
def predict():
    #reading the inputs given by the user
    input_features = [float(x) for x in request.form.values()]
    features_value = [np.array(input_features)]
```



```python
    features_name = ['blood_urea', 'blood glucose random','coronary_artery_disease'
       'anemia','pus_cell', 'red_blood_cells','diabetesmellitus', 'pedal_edema']

    df = pd.DataFrame(features_value, columns=features_name)

    output = model.predict(df) # predictions using the loaded model file

    # showing the prediction results in a UI# showing the prediction results in a UI
    return render_template('result.html', prediction_text=output)

if __name__ == '__main__':
    app.run(debug=True) # running the app
```