Mlmodel.py

```
import numpy as np
import pandas as pd
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split


import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")

df=pd.read_csv('kidney_disease.csv')
df
df.isnull().sum()
df['id'].value_counts()
del df["id"]
df['age'].value_counts()
df["bp"].value_counts()
df['bp']=df['bp'].fillna(df['bp'].mean())
df['sg'].value_counts()
df['sg']=df['sg'].fillna(df['sg'].mean())
df['al'].value_counts()
df['su'].value_counts()
df['su']=df['su'].fillna(df['su'].mean())
df['al']=df['al'].fillna(df['al'].mean())
df['rbc'].value_counts()
df.rbc.replace(np.nan,'normal',inplace=True)
df['rbc'].value_counts()
df['pc'].value_counts()
df.pc.replace(np.nan,'normal',inplace=True)
df['pc'].value_counts()
df['pcc'].value_counts()
df['ba'].value_counts()
df['bgr'].value_counts()
df['bgr']=df['bgr'].fillna(df['bgr'].mean())
df['bu'].value_counts()
df['sc'].value_counts()
df['bu']=df['bu'].fillna(df['bu'].mean())
df['sc']=df['sc'].fillna(df['sc'].mean())
df['sod'].value_counts()

df['pot'].value_counts()
df['sod']=df['sod'].fillna(df['sod'].mean())
```

```
df['pot']=df['pot'].fillna(df['pot'].mean())
df['hemo'].value_counts()
df['hemo']=df['hemo'].fillna(df['hemo'].mean())
df['pcv'].value_counts()
df.pcv.replace('\t43',np.nan,inplace=True)
df.pcv.replace('\t?',np.nan,inplace=True)
df.pcv.isnull().sum()
df['pcv']=pd.to_numeric(df['pcv'])
df['pcv']=df['pcv'].fillna(df['pcv'].mean())
df['wc'].value_counts()
df['rc'].value_counts()
df.rc.replace('\t?',np.nan,inplace=True)
df.wc.replace('\t?',np.nan,inplace=True)
df['rc']=pd.to_numeric(df['rc'])
df['wc']=pd.to_numeric(df['wc'])
df['rc']=df['rc'].fillna(df['rc'].mean())
df['wc']=df['wc'].fillna(df['wc'].mean())
df['htn'].value_counts()

df['dm'].value_counts()
df.dm.replace('\tno','no',inplace=True)
df.dm.replace('\tyes','yes',inplace=True)
df.dm.replace(' yes','yes',inplace=True)

df['dm'].value_counts()
df['cad'].value_counts()

df.cad.replace('\tno','no',inplace=True)
df['cad'].value_counts()
df['appet'].value_counts()
df['pe'].value_counts()
df['ane'].value_counts()
df["classification"].value_counts()
df.classification.replace('ckd\t','ckd',inplace=True)
df["classification"].value_counts()
df.dropna(inplace=True)
df.shape
Numcol=[]
for i in df.dtypes.index:
    if df.dtypes[i]!='object':
        Numcol.append(i)
Numcol
catcol=[]
for i in df.dtypes.index:
    if df.dtypes[i]=='object':
        catcol.append(i)
catcol
```

```
f=df[['age','bp','sg','su','bgr','bu','sc','sod','pot','hemo','pcv','wc','rc']
]
from scipy.stats import zscore
z=abs(zscore(f))
z
newdf=df[(z<3).all(axis=1)]
newdf
from sklearn.preprocessing import OrdinalEncoder
oe=OrdinalEncoder()
newdf[catcol]=oe.fit_transform(newdf[catcol])
newdf.head()

newdf.skew()

s=['age','su','rbc','pc','pcc','ba','bu','sc','sod','cad','appet','pe','ane']
from sklearn.preprocessing import PowerTransformer
scaler=PowerTransformer(method='yeo-johnson')
newdf[s]=scaler.fit_transform(newdf[s].values)
x=newdf[['age','bp','rbc','bu','hemo']]
x
y=newdf["classification"]
y
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=0)
from sklearn.metrics import classification_report,
accuracy_score,confusion_matrix
from sklearn.ensemble import AdaBoostClassifier
adb=(AdaBoostClassifier(random_state=1))
adb.fit(xtrain,ytrain)
ypred=adb.predict(xtest)
from sklearn.metrics import
confusion_matrix,classification_report,accuracy_score
cm=confusion_matrix(ytest,ypred)
cr=classification_report(ytest,ypred)
ac=accuracy_score(ytest,ypred)
print(f"Accuracy score:{ac}\n{cm}\n{cr}")
train=adb.score(xtrain,ytrain)
test=adb.score(xtest,ytest)
print(f"Training Score:{train}\n Testing Score:{test}")


def mymodel(model):
    model.fit(xtrain,ytrain)
    return model


def makepredict():
    adb=AdaBoostClassifier()
    model=mymodel(adb)
```

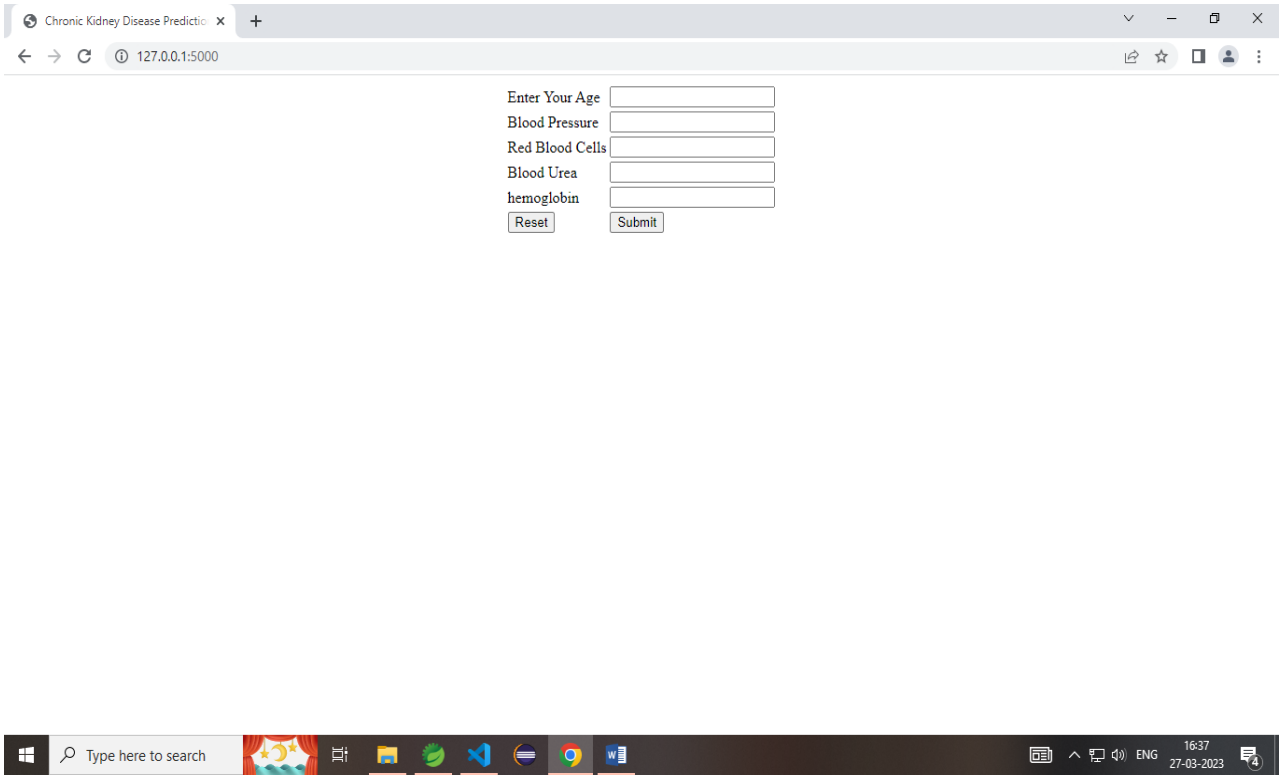
```
    return model
```

myapp.py

```
from flask import Flask,render_template,request
from mlmodel import *


f1=Flask(__name__)


@f1.route("/")
def home():
    return render_template("page1.html")


@f1.route("/getpredict",methods=['GET','POST'])
def prediction():
    if request.method=='POST':
        age=request.form['age']
        bp=request.form['bp']
        rbc=request.form['rbc']
        bu=request.form['bu']
        hemo=request.form['hemo']
        print(age)
        print(bp)
        print(rbc)
        print(bu)
        print(hemo)

        newobs=np.array([[age,bp,rbc,bu,hemo]],dtype=float)
        print(newobs)
        model=makepredict()
        yp=model.predict(newobs)[0]

        return render_template("page2.html",data=yp)

if __name__=="__main__":
    f1.run(debug=True)
```

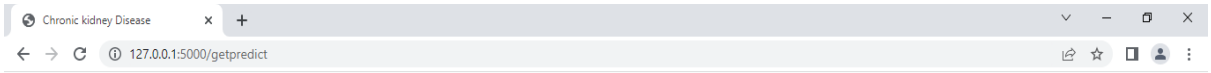
Page1.html

```
<html>
    <head>
        <title>Chronic Kidney Disease Prediction</title>
    </head>
<body>
    <div align="center">
        <form method="post" action="getpredict">
            <table>
                <tr> <td>Enter Your Age</td>
                     <td><input type="text" name="age"></td>
                </tr>
                <tr>
                    <td>Blood Pressure</td>
                    <td><input type="text" name="bp"></td>
                </tr>
                <tr><td>Red Blood Cells</td>
                    <td><input type="text" name="rbc"></td>
                </tr>
                <tr><td>Blood Urea</td>
                    <td><input type="text" name="bu"></td>
                </tr>
                <tr><td>hemoglobin</td>
                    <td><input type="text" name="hemo"></td>
                </tr>
                <tr><td><input type="reset">
                    <td><input type="submit">
                </tr>
            </table>
        </form>
    </div>
</body>
```

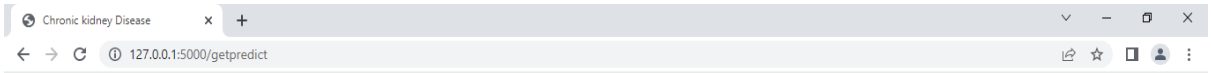


Page2.html

```
<html>
    <head>
        <title>Chronic kidney Disease</title>
    </head>
    <h1>The Patient is having {{data}}</h1>
```



The Patient is having 0.0



The Patient is having 1.0