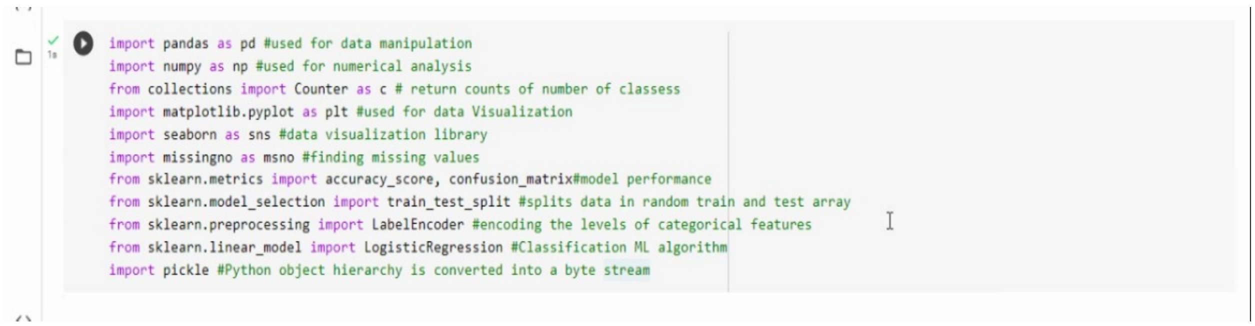
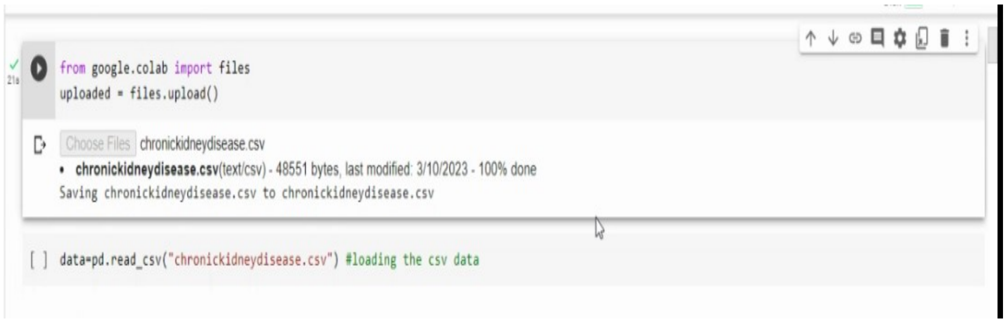


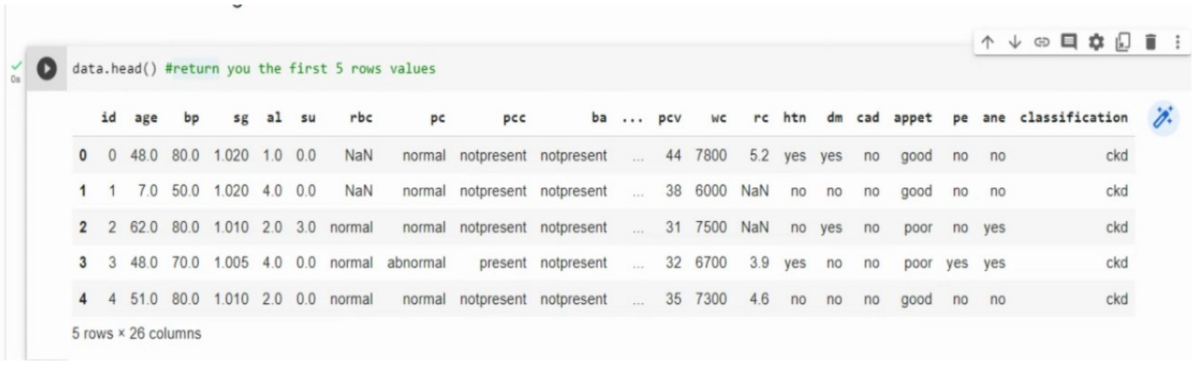
```python
import pandas as pd #used for data manipulation
import numpy as np #used for numerical analysis
from collections import Counter as c # return counts of number of classess
import matplotlib.pyplot as plt #used for data Visualization
import seaborn as sns #data visualization library
import missingno as msno #finding missing values
from sklearn.metrics import accuracy_score, confusion_matrix#model performance
from sklearn.model_selection import train_test_split #splits data in random train and test array
from sklearn.preprocessing import LabelEncoder #encoding the levels of categorical features
from sklearn.linear_model import LogisticRegression #Classification ML algorithm
import pickle #Python object hierarchy is converted into a byte stream
```



```python
from google.colab import files
uploaded = files.upload()
```

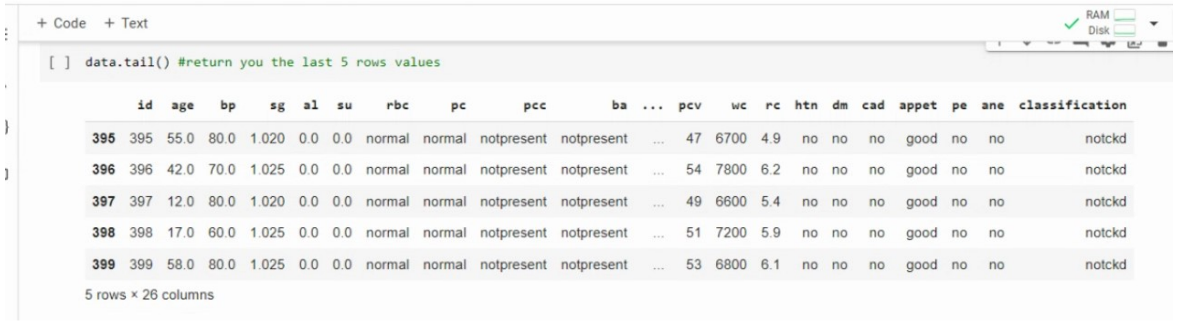
```
Choose Files chronickidneydisease.csv
• chronickidneydisease.csv(text/csv) - 48551 bytes, last modified: 3/10/2023 - 100% done
Saving chronickidneydisease.csv to chronickidneydisease.csv
```

```python
data=pd.read_csv("chronickidneydisease.csv") #loading the csv data
```



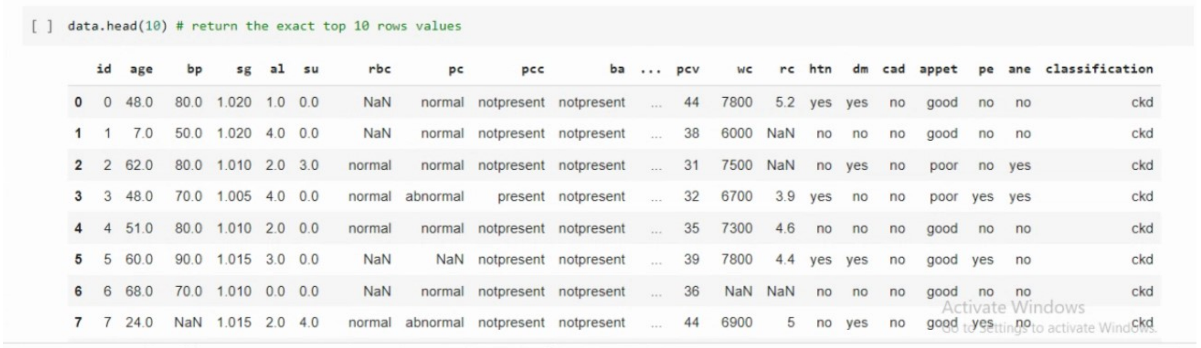
```python
data.head() #return you the first 5 rows values
```

```
   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
```

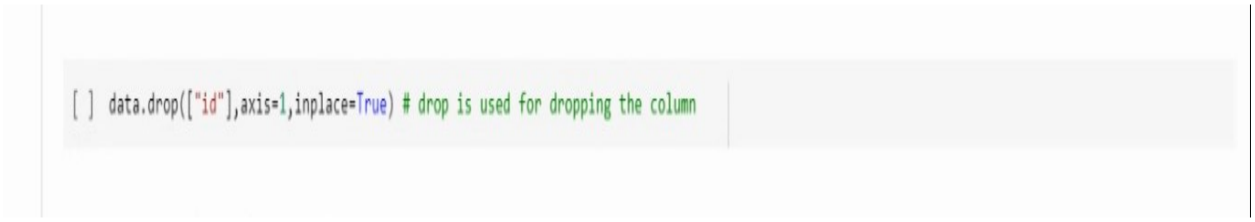


```python
data.tail() #return you the last 5 rows values
```

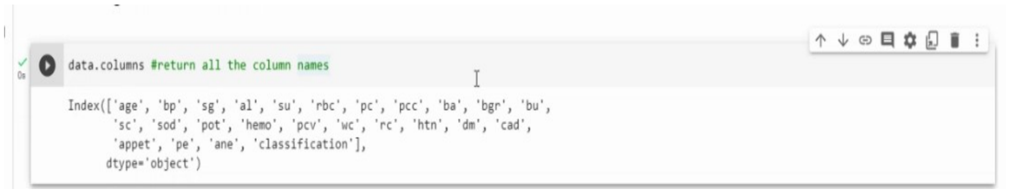
```
      id   age    bp     sg   al   su     rbc      pc         pcc          ba  ...  pcv    wc   rc  htn  dm  cad  appet  pe  ane  classification
395  395  55.0  80.0  1.020  0.0  0.0  normal  normal  notpresent  notpresent  ...   47  6700  4.9   no  no   no   good  no   no          notckd
396  396  42.0  70.0  1.025  0.0  0.0  normal  normal  notpresent  notpresent  ...   54  7800  6.2   no  no   no   good  no   no          notckd
397  397  12.0  80.0  1.020  0.0  0.0  normal  normal  notpresent  notpresent  ...   49  6600  5.4   no  no   no   good  no   no          notckd
398  398  17.0  60.0  1.025  0.0  0.0  normal  normal  notpresent  notpresent  ...   51  7200  5.9   no  no   no   good  no   no          notckd
399  399  58.0  80.0  1.025  0.0  0.0  normal  normal  notpresent  notpresent  ...   53  6800  6.1   no  no   no   good  no   no          notckd
5 rows × 26 columns
```



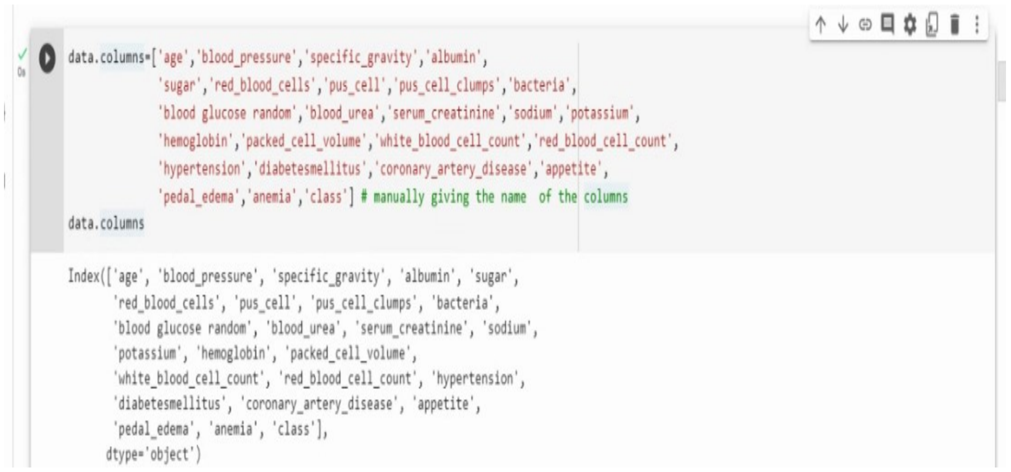
```
[ ] data.head(10) # return the exact top 10 rows values

   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   5  60.0  90.0  1.015  3.0  0.0     NaN       NaN  notpresent  notpresent  ...   39  7800  4.4  yes  yes   no   good  yes   no             ckd
6   6  68.0  70.0  1.010  0.0  0.0     NaN    normal  notpresent  notpresent  ...   36   NaN  NaN   no   no   no   good   no   no             ckd
7   7  24.0   NaN  1.015  2.0  4.0  normal  abnormal  notpresent  notpresent  ...   44  6900    5   no  yes   no   good  yes   no             ckd
```



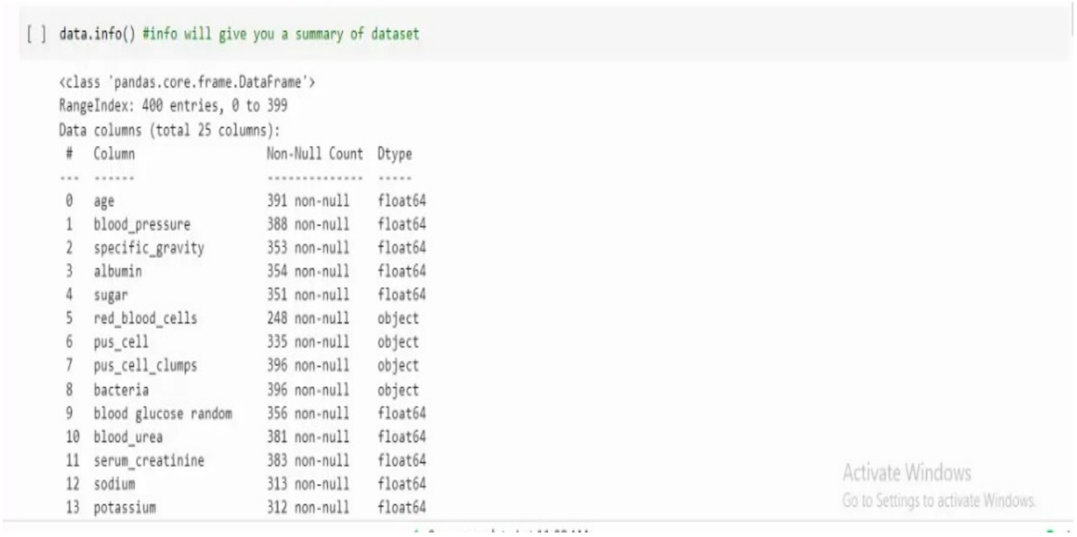
```
[ ] data.drop(["id"],axis=1,inplace=True) # drop is used for dropping the column
```



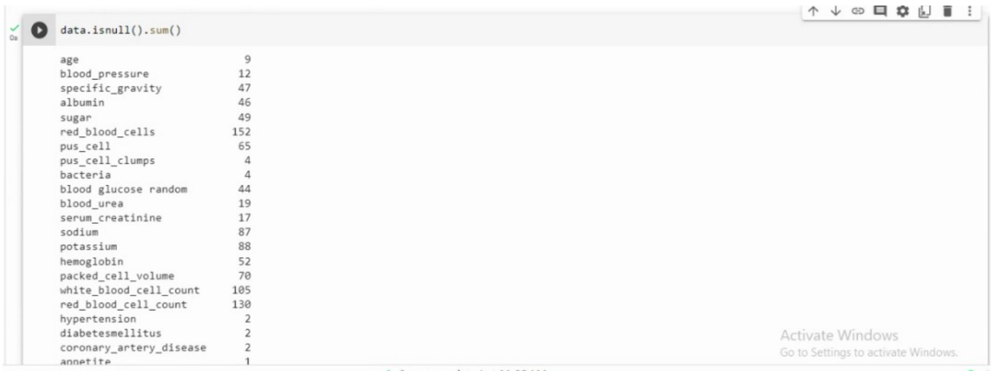
```
data.columns #return all the column names

Index(['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'],
      dtype='object')
```



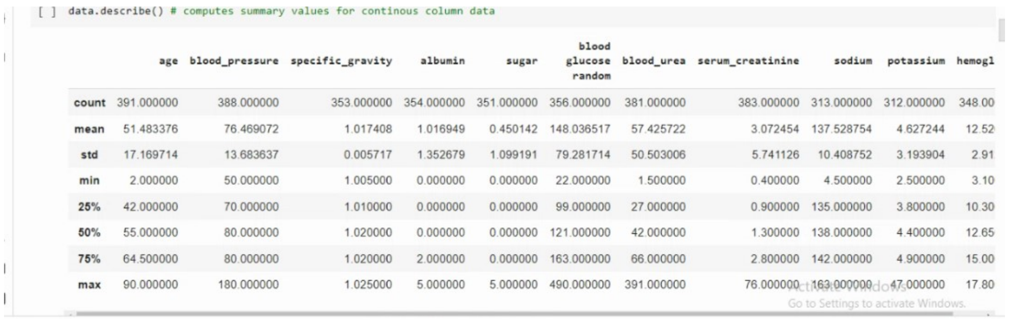
```
data.columns=['age','blood_pressure','specific_gravity','albumin',
              'sugar','red_blood_cells','pus_cell','pus_cell_clumps','bacteria',
              'blood glucose random','blood_urea','serum_creatinine','sodium','potassium',
              'hemoglobin','packed_cell_volume','white_blood_cell_count','red_blood_cell_count',
              'hypertension','diabetesmellitus','coronary_artery_disease','appetite',
              'pedal_edema','anemia','class'] # manually giving the name  of the columns
data.columns

Index(['age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar',
       'red_blood_cells', 'pus_cell', 'pus_cell_clumps', 'bacteria',
       'blood glucose random', 'blood_urea', 'serum_creatinine', 'sodium',
       'potassium', 'hemoglobin', 'packed_cell_volume',
       'white_blood_cell_count', 'red_blood_cell_count', 'hypertension',
       'diabetesmellitus', 'coronary_artery_disease', 'appetite',
       'pedal_edema', 'anemia', 'class'],
      dtype='object')
```



```
[ ] data.info() #info will give you a summary of dataset

<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   blood_pressure        388 non-null    float64
 2   specific_gravity      353 non-null    float64
 3   albumin               354 non-null    float64
 4   sugar                 351 non-null    float64
 5   red_blood_cells       248 non-null    object
 6   pus_cell              335 non-null    object
 7   pus_cell_clumps       396 non-null    object
 8   bacteria              396 non-null    object
 9   blood glucose random  356 non-null    float64
 10  blood_urea            381 non-null    float64
 11  serum_creatinine      383 non-null    float64
 12  sodium                313 non-null    float64
 13  potassium             312 non-null    float64
```

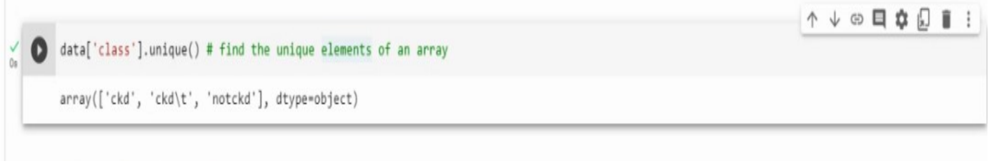


```
data.isnull().sum()

age                        9
blood_pressure            12
specific_gravity          47
albumin                   46
sugar                     49
red_blood_cells          152
pus_cell                  65
pus_cell_clumps            4
bacteria                   4
blood glucose random      44
blood_urea                19
serum_creatinine          17
sodium                    87
potassium                 88
hemoglobin                52
packed_cell_volume        70
white_blood_cell_count   105
red_blood_cell_count     130
hypertension               2
diabetesmellitus           2
coronary_artery_disease    2
appetite                   1
```

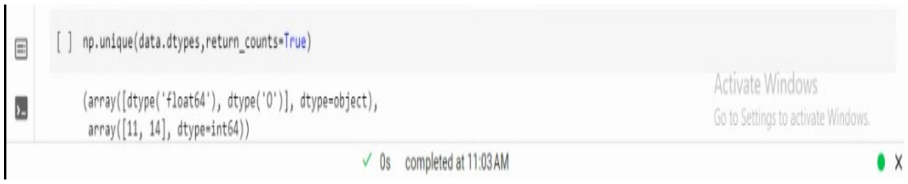


```
[ ] data.describe() # computes summary values for continous column data
```

	age	blood_pressure	specific_gravity	albumin	sugar	blood glucose random	blood_urea	serum_creatinine	sodium	potassium	hemogl
count	391.000000	388.000000	353.000000	354.000000	351.000000	356.000000	381.000000	383.000000	313.000000	312.000000	348.00
mean	51.483376	76.469072	1.017408	1.016949	0.450142	148.036517	57.425722	3.072454	137.528754	4.627244	12.52
std	17.169714	13.683637	0.005717	1.352679	1.099191	79.281714	50.503006	5.741126	10.408752	3.193904	2.91
min	2.000000	50.000000	1.005000	0.000000	0.000000	22.000000	1.500000	0.400000	4.500000	2.500000	3.10
25%	42.000000	70.000000	1.010000	0.000000	0.000000	99.000000	27.000000	0.900000	135.000000	3.800000	10.30
50%	55.000000	80.000000	1.020000	0.000000	0.000000	121.000000	42.000000	1.300000	138.000000	4.400000	12.65
75%	64.500000	80.000000	1.020000	2.000000	0.000000	163.000000	66.000000	2.800000	142.000000	4.900000	15.00
max	90.000000	180.000000	1.025000	5.000000	5.000000	490.000000	391.000000	76.000000	163.000000	47.000000	17.80

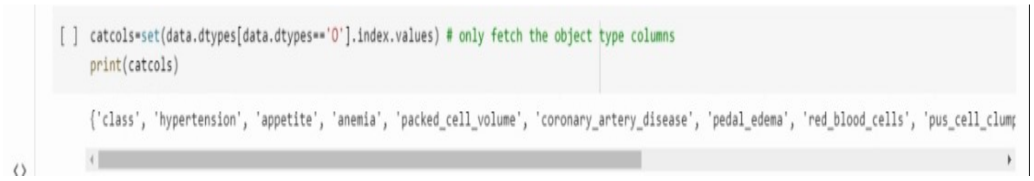


```
data['class'].unique() # find the unique elements of an array

array(['ckd', 'ckd\t', 'notckd'], dtype=object)
```

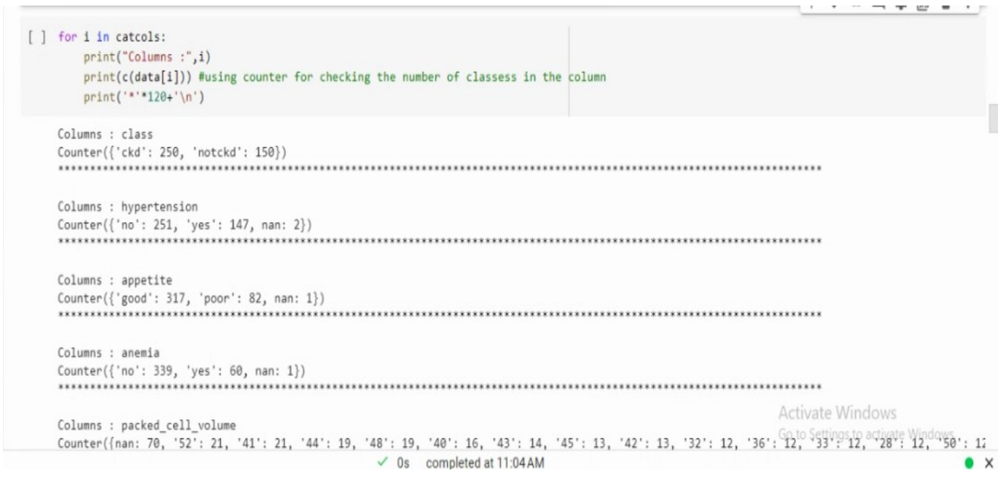


```
[ ] np.unique(data.dtypes,return_counts=True)

(array([dtype('float64'), dtype('O')], dtype=object),
 array([11, 14], dtype=int64))
```

✓ 0s completed at 11:03 AM

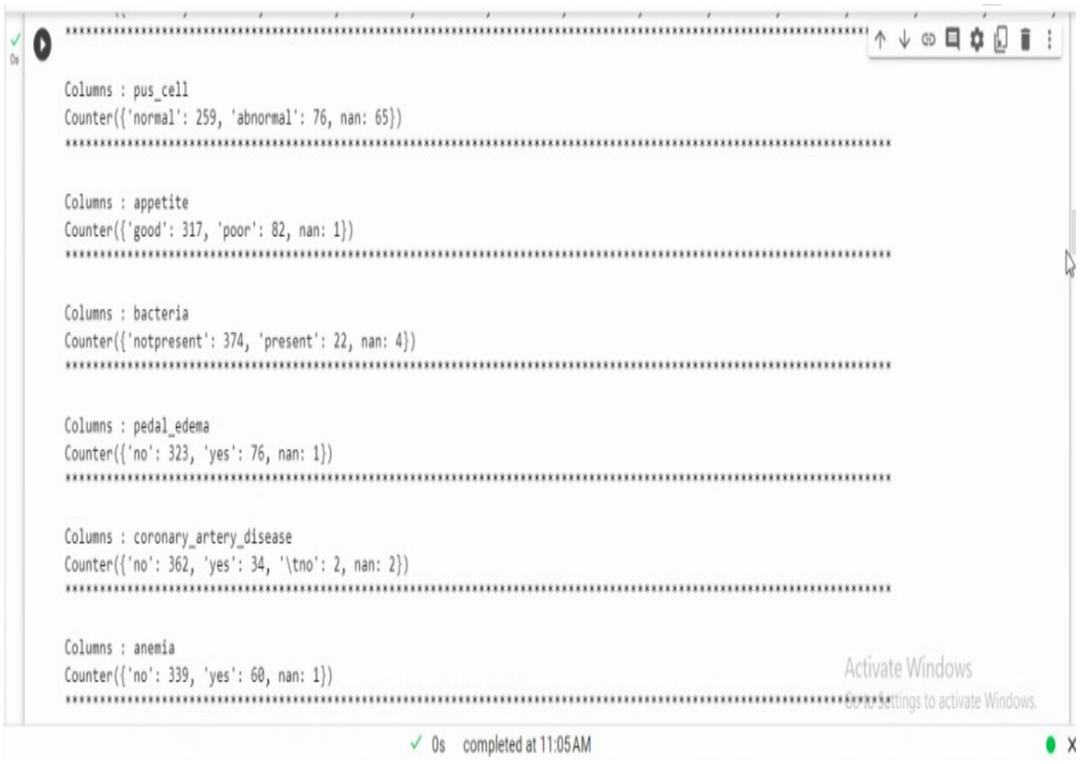


```
[ ] catcols=set(data.dtypes[data.dtypes=='O'].index.values) # only fetch the object type columns
    print(catcols)

{'class', 'hypertension', 'appetite', 'anemia', 'packed_cell_volume', 'coronary_artery_disease', 'pedal_edema', 'red_blood_cells', 'pus_cell_clumr
```



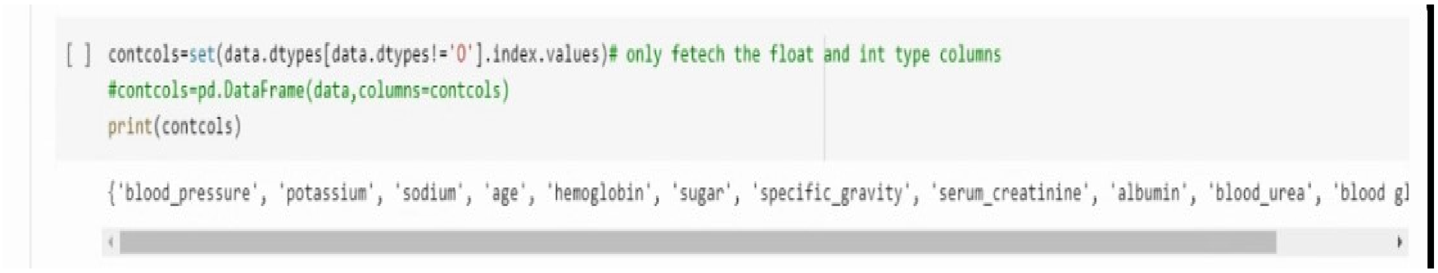
```
[ ] for i in catcols:
        print("Columns :",i)
        print(c(data[i])) #using counter for checking the number of classess in the column
        print('*'*120+'\n')

    Columns : class
    Counter({'ckd': 250, 'notckd': 150})
    ************************************************************************************************************************

    Columns : hypertension
    Counter({'no': 251, 'yes': 147, nan: 2})
    ************************************************************************************************************************

    Columns : appetite
    Counter({'good': 317, 'poor': 82, nan: 1})
    ************************************************************************************************************************

    Columns : anemia
    Counter({'no': 339, 'yes': 60, nan: 1})
    ************************************************************************************************************************

    Columns : packed_cell_volume
    Counter({nan: 70, '52': 21, '41': 21, '44': 19, '48': 19, '40': 16, '43': 14, '45': 13, '42': 13, '32': 12, '36': 12, '33': 12, '28': 12, '50': 12
```

✓ 0s completed at 11:04 AM

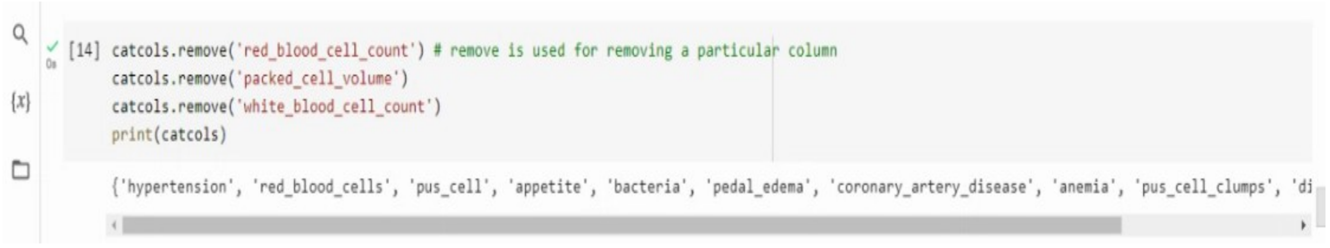


```
****************************************************************************************************

Columns : pus_cell
Counter({'normal': 259, 'abnormal': 76, nan: 65})
****************************************************************************************************

Columns : appetite
Counter({'good': 317, 'poor': 82, nan: 1})
****************************************************************************************************

Columns : bacteria
Counter({'notpresent': 374, 'present': 22, nan: 4})
****************************************************************************************************

Columns : pedal_edema
Counter({'no': 323, 'yes': 76, nan: 1})
****************************************************************************************************

Columns : coronary_artery_disease
Counter({'no': 362, 'yes': 34, '\tno': 2, nan: 2})
****************************************************************************************************

Columns : anemia
Counter({'no': 339, 'yes': 60, nan: 1})
****************************************************************************************************
```

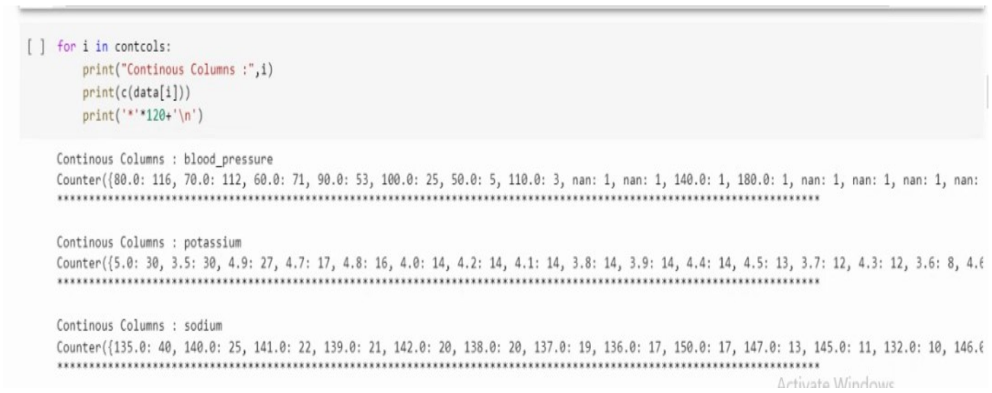
✓ 0s completed at 11:05 AM



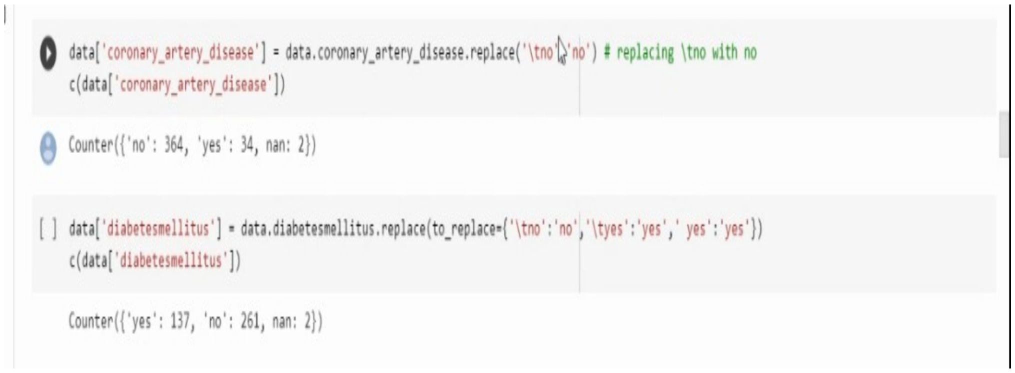
```
[ ] contcols=set(data.dtypes[data.dtypes!='O'].index.values)# only fetech the float and int type columns
    #contcols=pd.DataFrame(data,columns=contcols)
    print(contcols)

    {'blood_pressure', 'potassium', 'sodium', 'age', 'hemoglobin', 'sugar', 'specific_gravity', 'serum_creatinine', 'albumin', 'blood_urea', 'blood gl
```



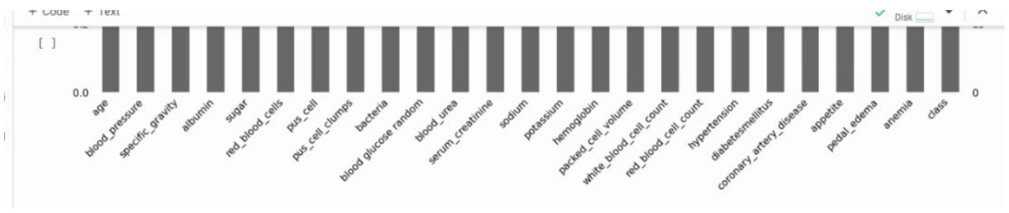
```
[14] catcols.remove('red_blood_cell_count') # remove is used for removing a particular column
     catcols.remove('packed_cell_volume')
     catcols.remove('white_blood_cell_count')
     print(catcols)

     {'hypertension', 'red_blood_cells', 'pus_cell', 'appetite', 'bacteria', 'pedal_edema', 'coronary_artery_disease', 'anemia', 'pus_cell_clumps', 'di
```

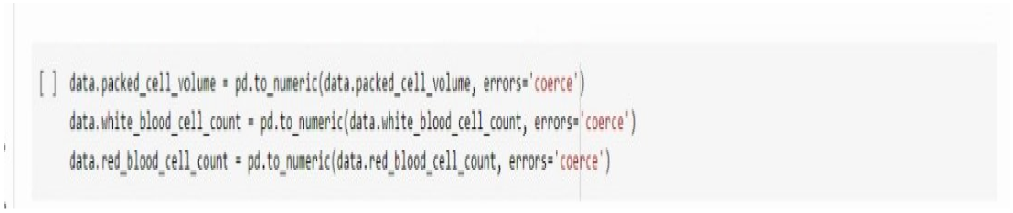


```
[ ] for i in contcols:
        print("Continous Columns :",i)
        print(c(data[i]))
        print('*'*120+'\n')

    Continous Columns : blood_pressure
    Counter({80.0: 116, 70.0: 112, 60.0: 71, 90.0: 53, 100.0: 25, 50.0: 5, 110.0: 3, nan: 1, nan: 1, 140.0: 1, 180.0: 1, nan: 1, nan: 1, nan: 1, nan:
    ************************************************************************************************************************

    Continous Columns : potassium
    Counter({5.0: 30, 3.5: 30, 4.9: 27, 4.7: 17, 4.8: 16, 4.0: 14, 4.2: 14, 4.1: 14, 3.8: 14, 3.9: 14, 4.4: 14, 4.5: 13, 3.7: 12, 4.3: 12, 3.6: 8, 4.6
    ************************************************************************************************************************

    Continous Columns : sodium
    Counter({135.0: 40, 140.0: 25, 141.0: 22, 139.0: 21, 142.0: 20, 138.0: 20, 137.0: 19, 136.0: 17, 150.0: 17, 147.0: 13, 145.0: 11, 132.0: 10, 146.6
    ************************************************************************************************************************
```

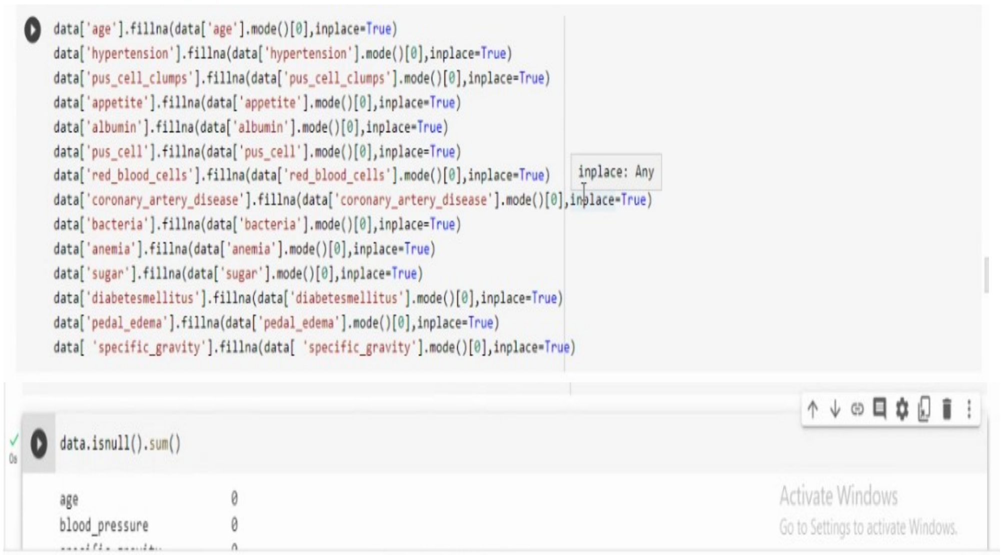


```
data['coronary_artery_disease'] = data.coronary_artery_disease.replace('\tno','no') # replacing \tno with no
c(data['coronary_artery_disease'])

Counter({'no': 364, 'yes': 34, nan: 2})

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

    Counter({'yes': 137, 'no': 261, nan: 2})
```





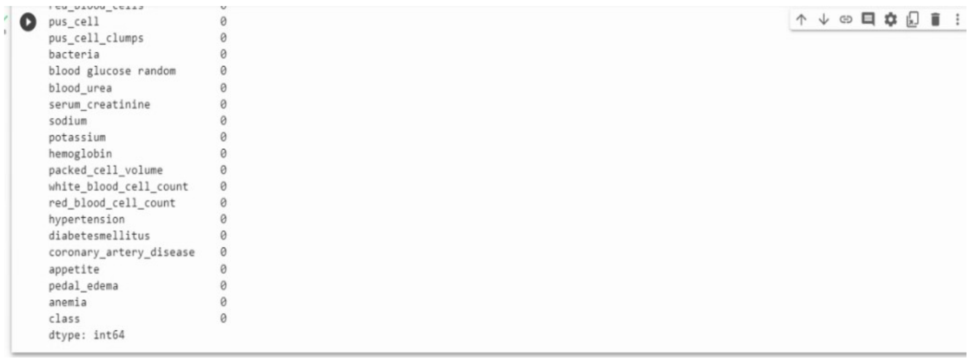
```
[ ] data.packed_cell_volume = pd.to_numeric(data.packed_cell_volume, errors='coerce')
    data.white_blood_cell_count = pd.to_numeric(data.white_blood_cell_count, errors='coerce')
    data.red_blood_cell_count = pd.to_numeric(data.red_blood_cell_count, errors='coerce')
```



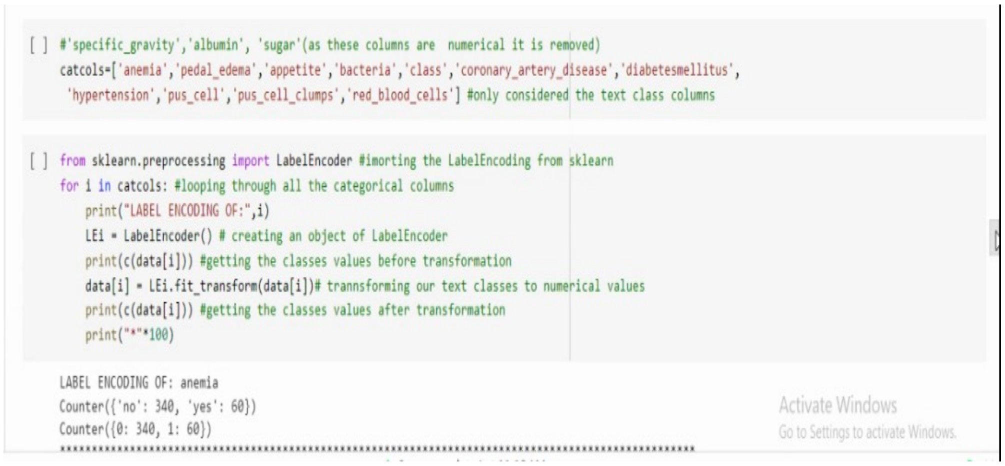
```
data['age'].fillna(data['age'].mode()[0],inplace=True)
data['hypertension'].fillna(data['hypertension'].mode()[0],inplace=True)
data['pus_cell_clumps'].fillna(data['pus_cell_clumps'].mode()[0],inplace=True)
data['appetite'].fillna(data['appetite'].mode()[0],inplace=True)
data['albumin'].fillna(data['albumin'].mode()[0],inplace=True)
data['pus_cell'].fillna(data['pus_cell'].mode()[0],inplace=True)
data['red_blood_cells'].fillna(data['red_blood_cells'].mode()[0],inplace=True)
data['coronary_artery_disease'].fillna(data['coronary_artery_disease'].mode()[0],inplace=True)
data['bacteria'].fillna(data['bacteria'].mode()[0],inplace=True)
data['anemia'].fillna(data['anemia'].mode()[0],inplace=True)
data['sugar'].fillna(data['sugar'].mode()[0],inplace=True)
data['diabetesmellitus'].fillna(data['diabetesmellitus'].mode()[0],inplace=True)
data['pedal_edema'].fillna(data['pedal_edema'].mode()[0],inplace=True)
data[ 'specific_gravity'].fillna(data[ 'specific_gravity'].mode()[0],inplace=True)
```

```
data.isnull().sum()
```

```
age                  0
blood_pressure       0
```



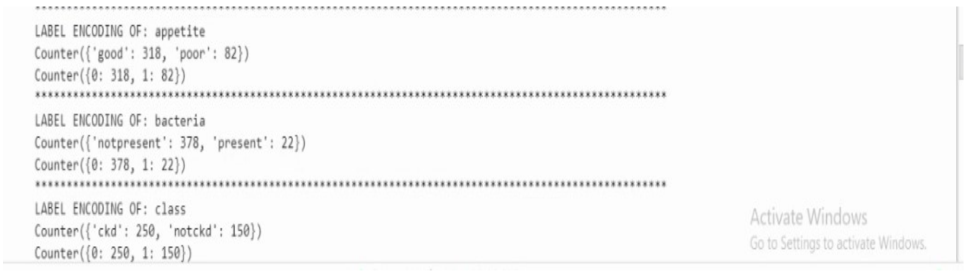
```
pus_cell                   0
pus_cell_clumps            0
bacteria                   0
blood glucose random       0
blood_urea                 0
serum_creatinine           0
sodium                     0
potassium                  0
hemoglobin                 0
packed_cell_volume         0
white_blood_cell_count     0
red_blood_cell_count       0
hypertension               0
diabetesmellitus           0
coronary_artery_disease    0
appetite                   0
pedal_edema                0
anemia                     0
class                      0
dtype: int64
```



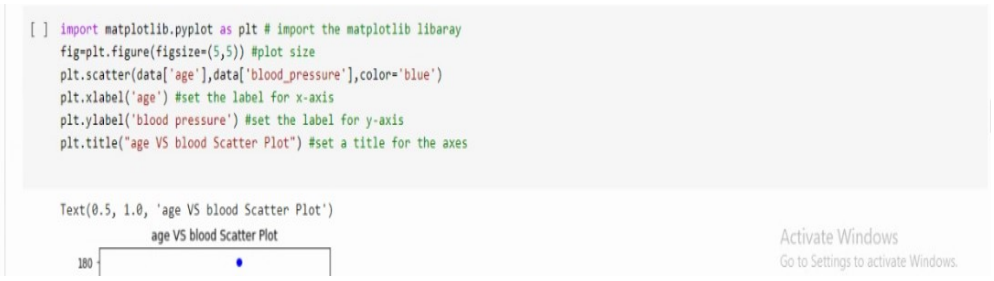
```
#'specific_gravity','albumin', 'sugar'(as these columns are  numerical it is removed)
catcols=['anemia','pedal_edema','appetite','bacteria','class','coronary_artery_disease','diabetesmellitus',
 'hypertension','pus_cell','pus_cell_clumps','red_blood_cells'] #only considered the text class columns
```

```
from sklearn.preprocessing import LabelEncoder #imorting the LabelEncoding from sklearn
for i in catcols: #looping through all the categorical columns
    print("LABEL ENCODING OF:",i)
    LEi = LabelEncoder() # creating an object of LabelEncoder
    print(c(data[i])) #getting the classes values before transformation
    data[i] = LEi.fit_transform(data[i])# trannsforming our text classes to numerical values
    print(c(data[i])) #getting the classes values after transformation
    print("*"*100)
```

```
LABEL ENCODING OF: anemia
Counter({'no': 340, 'yes': 60})
Counter({0: 340, 1: 60})
****************************************************************************************************
```

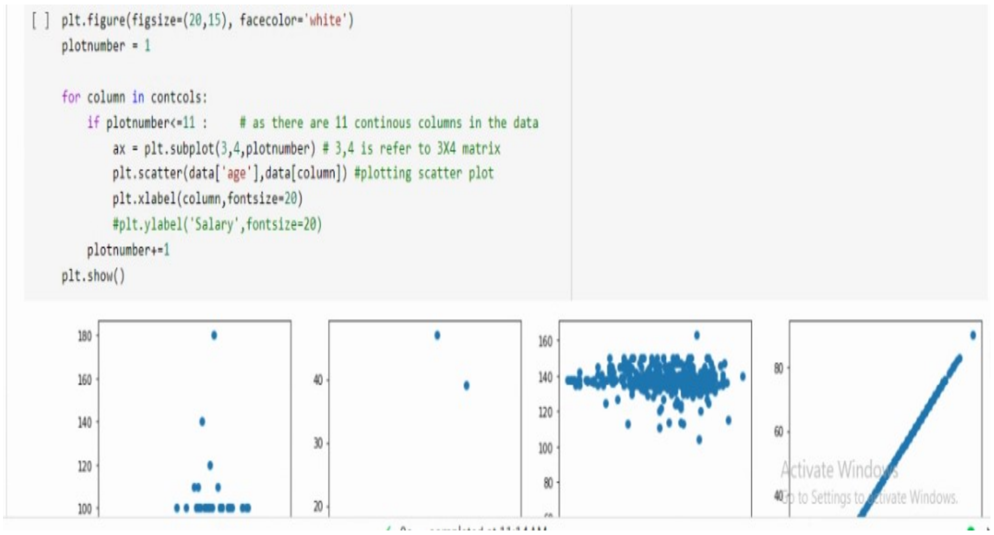


```
----------------------------------------------------------------------------------------------------
LABEL ENCODING OF: appetite
Counter({'good': 318, 'poor': 82})
Counter({0: 318, 1: 82})
****************************************************************************************************
LABEL ENCODING OF: bacteria
Counter({'notpresent': 378, 'present': 22})
Counter({0: 378, 1: 22})
****************************************************************************************************
LABEL ENCODING OF: class
Counter({'ckd': 250, 'notckd': 150})
Counter({0: 250, 1: 150})
```

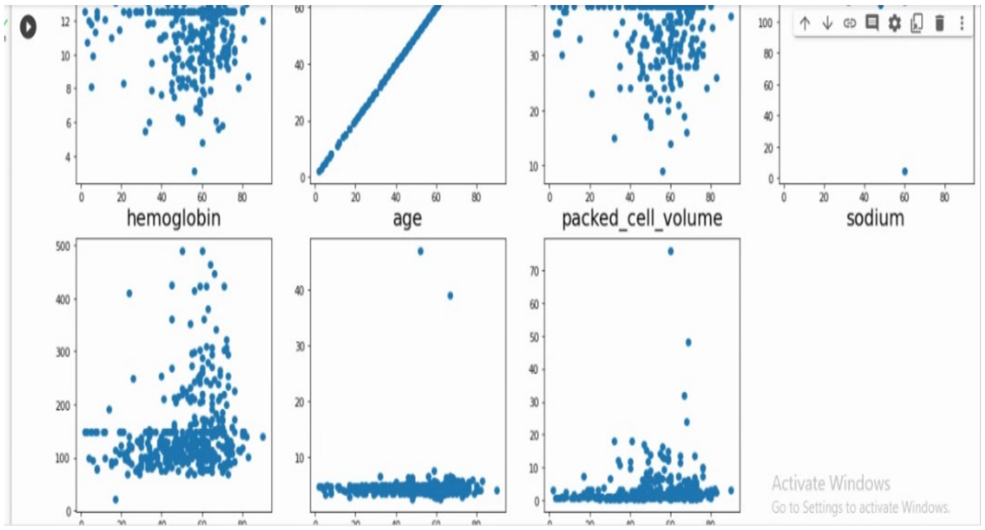


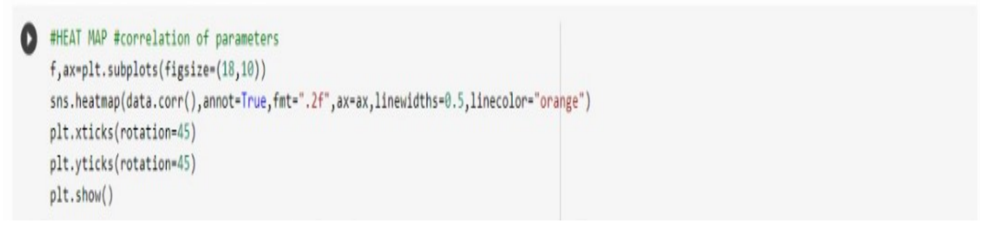
```
[ ] import matplotlib.pyplot as plt # import the matplotlib libaray
    fig=plt.figure(figsize=(5,5)) #plot size
    plt.scatter(data['age'],data['blood_pressure'],color='blue')
    plt.xlabel('age') #set the label for x-axis
    plt.ylabel('blood pressure') #set the label for y-axis
    plt.title("age VS blood Scatter Plot") #set a title for the axes
```

```
Text(0.5, 1.0, 'age VS blood Scatter Plot')
```

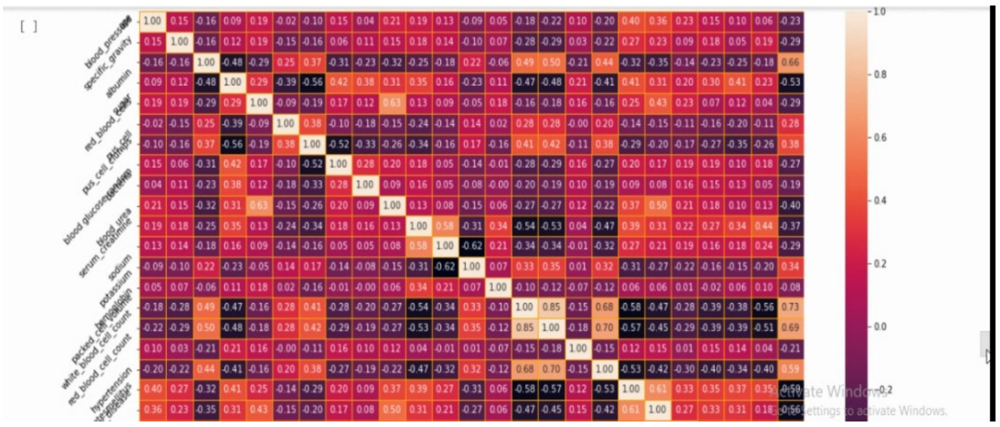


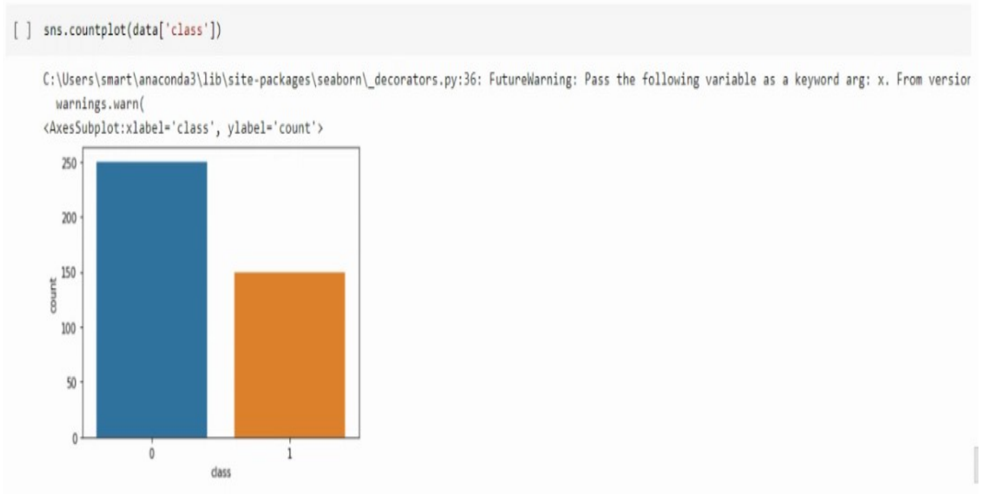
```
[ ] plt.figure(figsize=(20,15), facecolor='white')
    plotnumber = 1

    for column in contcols:
        if plotnumber<=11 :     # as there are 11 continous columns in the data
            ax = plt.subplot(3,4,plotnumber) # 3,4 is refer to 3X4 matrix
            plt.scatter(data['age'],data[column]) #plotting scatter plot
            plt.xlabel(column,fontsize=20)
            #plt.ylabel('Salary',fontsize=20)
        plotnumber+=1
    plt.show()
```





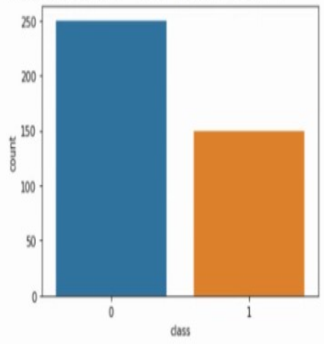
```
#HEAT MAP #correlation of parameters
f,ax=plt.subplots(figsize=(18,10))
sns.heatmap(data.corr(),annot=True,fmt=".2f",ax=ax,linewidths=0.5,linecolor="orange")
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.show()
```

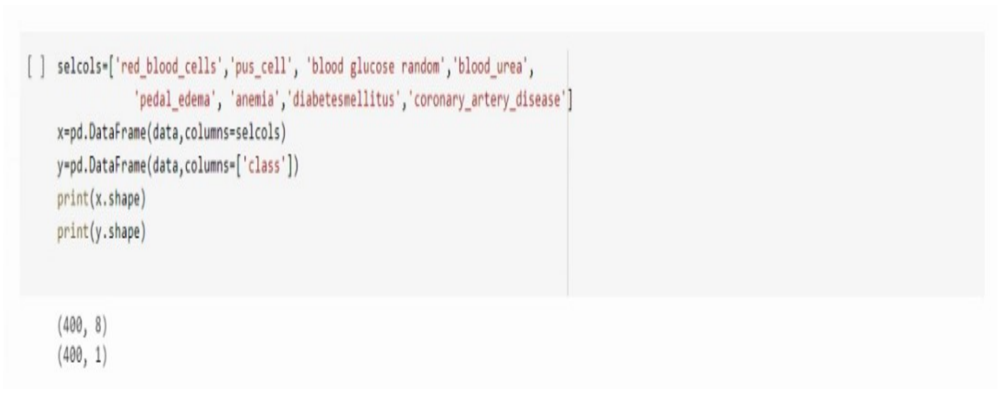




```
[ ] sns.countplot(data['class'])
```

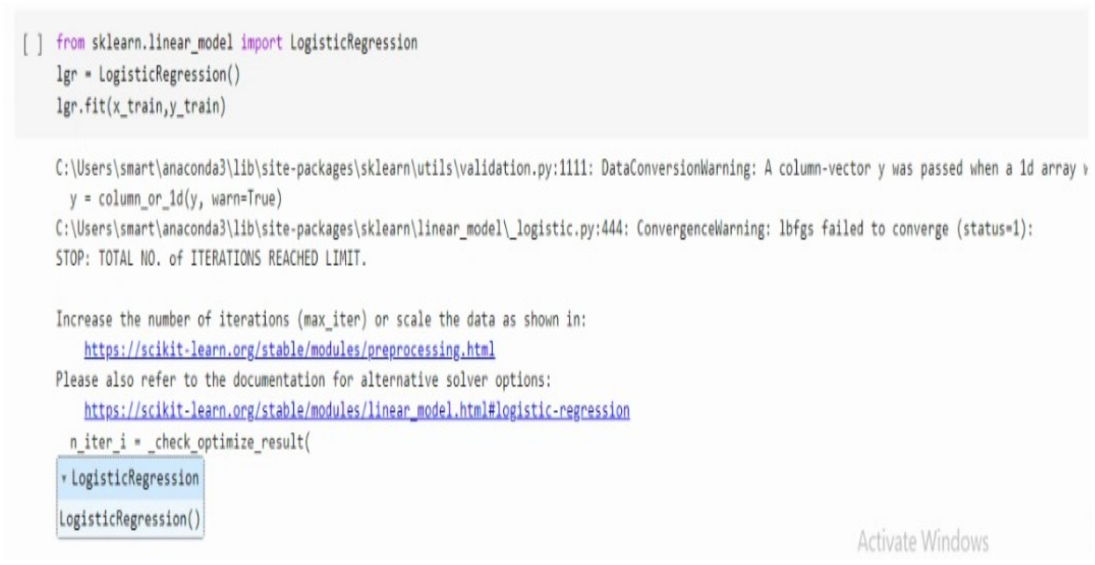
```
C:\Users\smart\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From versior
  warnings.warn(
<AxesSubplot:xlabel='class', ylabel='count'>
```





```
[ ] selcols=['red_blood_cells','pus_cell', 'blood glucose random','blood_urea',
            'pedal_edema', 'anemia','diabetesmellitus','coronary_artery_disease']
    x=pd.DataFrame(data,columns=selcols)
    y=pd.DataFrame(data,columns=['class'])
    print(x.shape)
    print(y.shape)
```

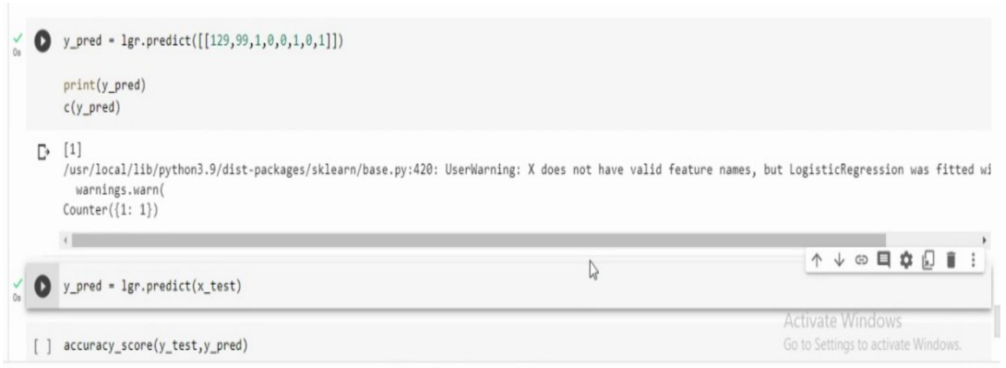
```
(400, 8)
(400, 1)
```



```
[ ] from sklearn.linear_model import LogisticRegression
    lgr = LogisticRegression()
    lgr.fit(x_train,y_train)
```

```
C:\Users\smart\anaconda3\lib\site-packages\sklearn\utils\validation.py:1111: DataConversionWarning: A column-vector y was passed when a 1d array w
  y = column_or_1d(y, warn=True)
C:\Users\smart\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:444: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

Activate Windows



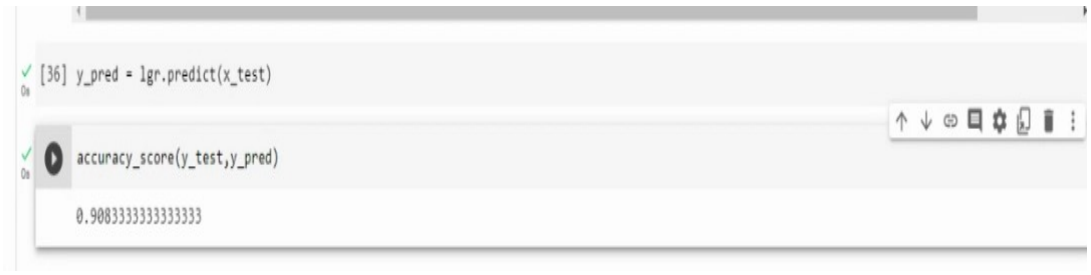
```
y_pred = lgr.predict([[129,99,1,0,0,1,0,1]])

print(y_pred)
c(y_pred)
```

```
[1]
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:420: UserWarning: X does not have valid feature names, but LogisticRegression was fitted wi
  warnings.warn(
Counter({1: 1})
```

```
y_pred = lgr.predict(x_test)
```

```
[ ] accuracy_score(y_test,y_pred)
```

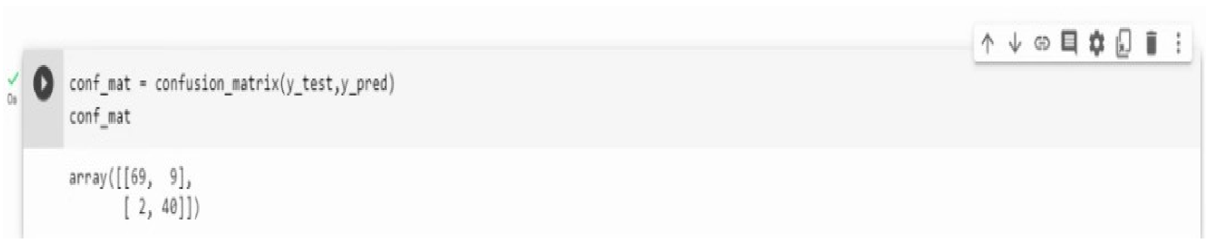
Activate Windows
Go to Settings to activate Windows.



```
[36] y_pred = lgr.predict(x_test)
```

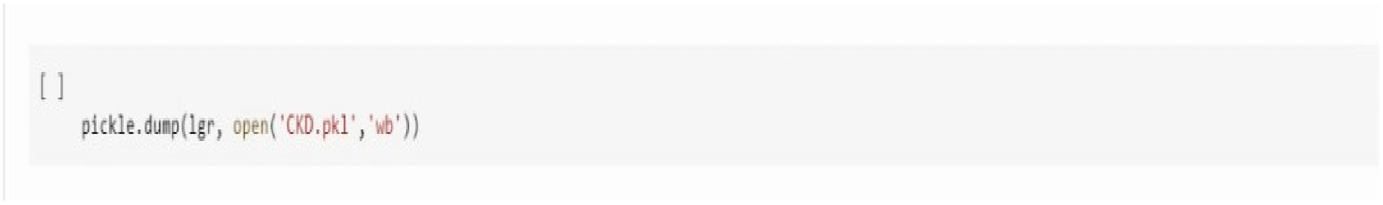
```
accuracy_score(y_test,y_pred)
```

```
0.9083333333333333
```

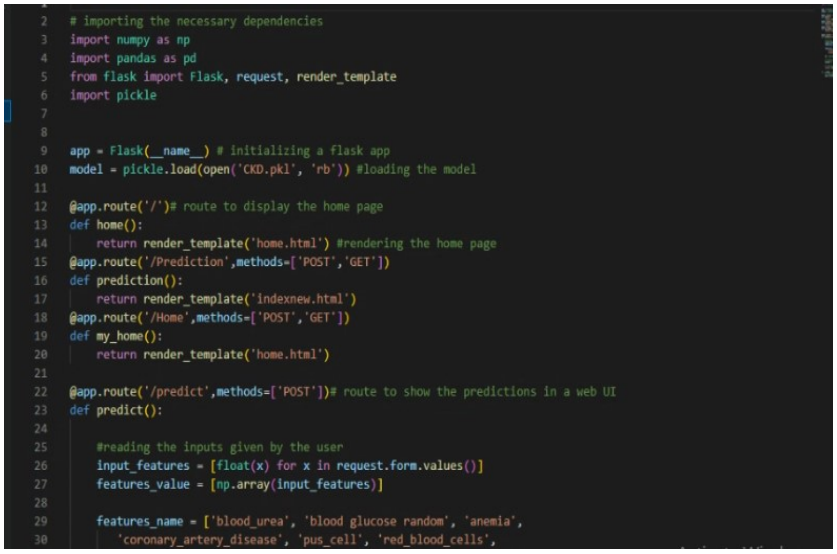


```
conf_mat = confusion_matrix(y_test,y_pred)
conf_mat
```

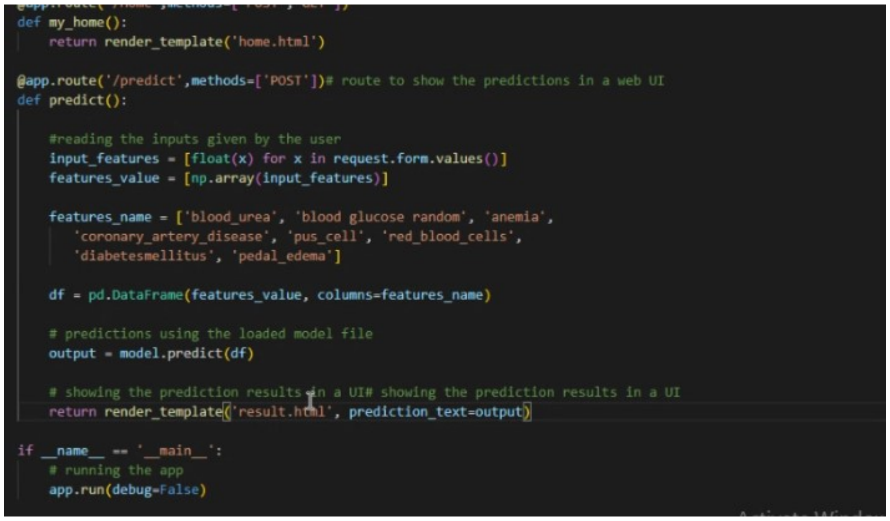
```
array([[69,  9],
       [ 2, 40]])
```



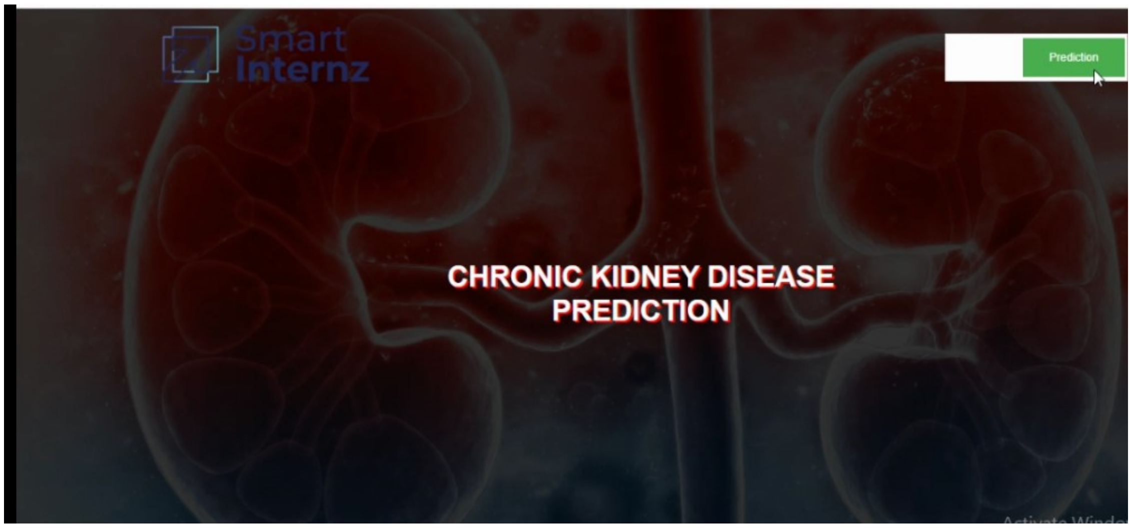
```
[ ]
pickle.dump(lgr, open('CKD.pkl','wb'))
```

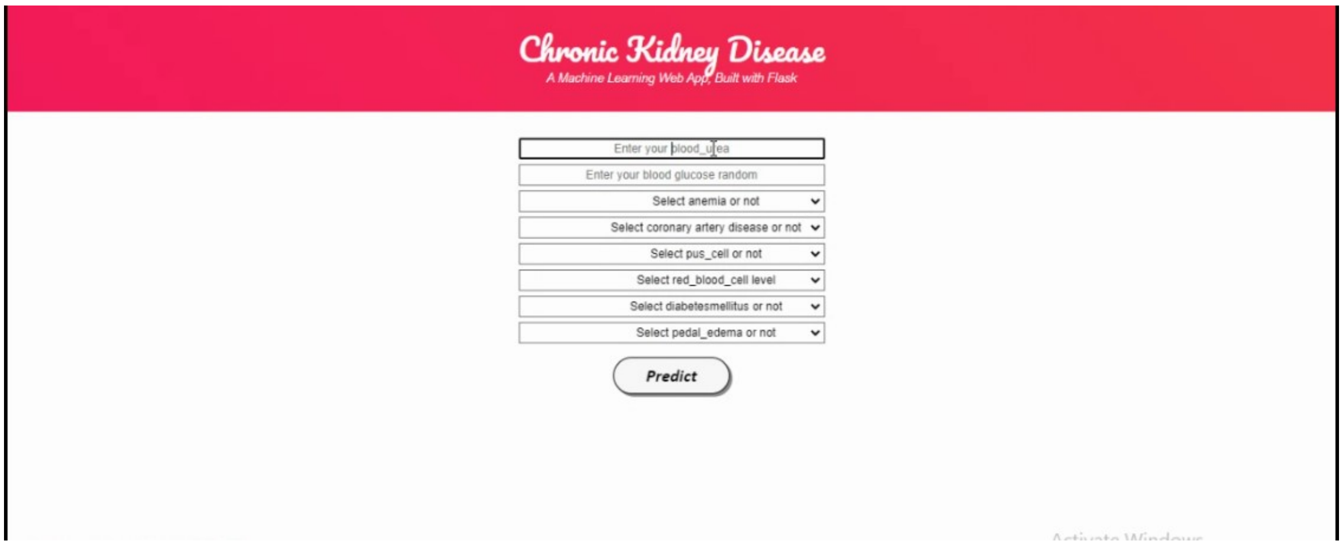


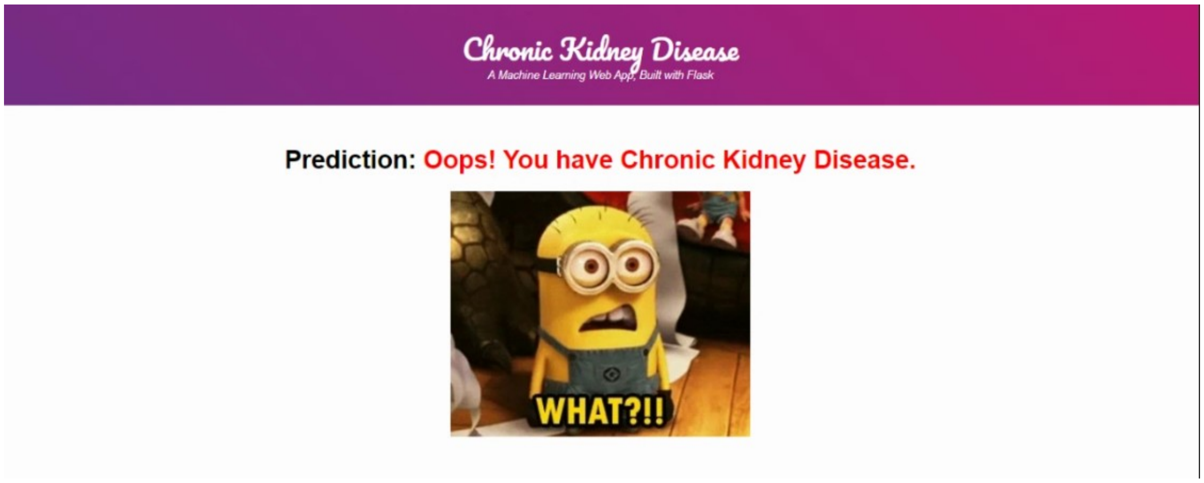
```
# importing the necessary dependencies
import numpy as np
import pandas as pd
from flask import Flask, request, render_template
import pickle


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():
    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)]

    features_name = ['blood_urea', 'blood glucose random', 'anemia',
        'coronary_artery_disease', 'pus_cell', 'red_blood_cells',
```



```python
[illegible]
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)]

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

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

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

    # 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__':
    # running the app
    app.run(debug=False)
```


Smart Internz
Prediction
CHRONIC KIDNEY DISEASE PREDICTION


Chronic Kidney Disease
A Machine Learning Web App, Built with Flask
Enter your blood_urea
Enter your blood glucose random
Select anemia or not
Select coronary artery disease or not
Select pus_cell or not
Select red_blood_cell level
Select diabetesmellitus or not
Select pedal_edema or not
Predict



Chronic Kidney Disease

A Machine Learning Web App, Built with Flask

Prediction: Oops! You have Chronic Kidney Disease.