DRUG CLASSIFICATION USING IBM WATSON STUDIO WITH MACHINE LEARNING

Category: Machine Learning

PROJECT DESCRIPTION

INTRODUCTION

Nowadays our lifestyle has been changing. Per family, at least one person has Motorcycles or cars, etc. In the same way, we all have health issues. An earlier generation has proved “Health is Wealth”. But, for our generation, this slogan is quite challenging.

We have completely moved with hybrid veggies, junk foods, etc. Due to these foods, we are not getting sufcient nutrition and suffering from health issues. To overcome this, we are consulting doctors and taking some drugs as medicines. In this project, we have some characteristics of the patients as a dataset. The target variable of this dataset is Drugs. The drug names are confdential. So, those names are replaced as DrugX, DrugY, DrugA , DrugB, and DrugC . By consulting a doctor each time, you have to pay a doctor fee and additional charges. For saving money and time, you can use this web application to predict your drug type. The main purpose of the Drug Classifcation system is to predict the suitable drug type confdently for the patients based on their characteristics. The main problem here is not just the feature sets and target sets but also the approach that is taken in solving these types of problems.

We will be using classifcation algorithms such as Decision tree, Random forest, KNN, and xgboost. We will train and test the data with these algorithms. From this best model is selected and saved in pkl format. We will be doing fask integration and IBM deployment.

PRE-REQUISITES

In order to develop this project we need to install the following software/packages:

Step 1: Anaconda Navigator

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning-related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform, package management system. Anaconda comes with great tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code.

For this project, we will be using a Jupyter notebook and Spyder

Step 2: Python packages

To build Machine learning models you must require the following packages

1.Sklearn: Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms.

2.NumPy: NumPy is a Python package that stands for 'Numerical Python. It is the core library for scientifc computing, which contains a powerful n-dimensional array of object

3.Pandas: pandas is a fast, powerful, fexible, and easy-to-use open-source data analysis and manipulation tool, built on top of the Python programming language.

4.Matplotlib: It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits

Step 3: Flask - Web framework used for building Web applications.

If you are using anaconda navigator, follow the below steps to download the required packages:

Open anaconda prompt as administrator

- Type “pip install numpy ” and click enter
- Type “pip install pandas ” and click enter
- Type “pip install scikit-learn” and click enter.
- Type ”pip install matplotlib” and click enter.
- Type ”pip install scipy" and click enter.
- Type ”pip install pickle-mixin” and click enter.
- Type ”pip install seaborn” and click enter.
- Type “pip install Flask ” and click enter.

PROJECT OBJECTIVES

By the end of this project:

1.You’ll be able to understand the problem to classify if it is a regression or a classifcation kind of problem.

2.You will be able to know how to pre-process/clean the data using different data preprocessing techniques.

3.You will be able to analyze or get insights into data through visualization.

4. Applying different algorithms according to the dataset and based on visualization.

5.You will be able to know how to build a web application using the Flask framework

PROJECT FLOW

- The user interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once the model analyses the input the prediction is showcased on the

UI To accomplish this, we have to complete all the activities listed below,
Data collection

- Collect the dataset or create the dataset

- Univariate analysis
- Bivariate analysis
- Multivariate analysis
- Descriptive analysis

Data pre-processing

- Checking for null values
- Handling outlier
- Handling categorical data
- Splitting data into train and test

Model building

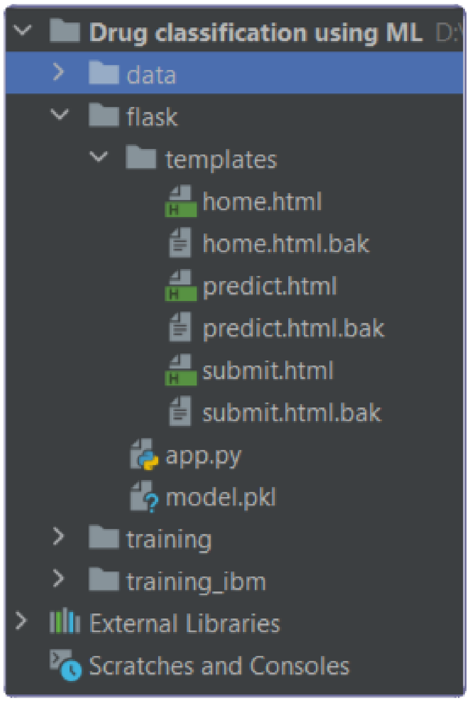
- Import the model building libraries

- Initializing the model
- Training and testing the model
- Evaluating the performance of the model
- Save the model

- Create an HTML file
- Build python code
-

PROJECT STRUCTURE

Create a Project folder that contains files as shown below



```
Drug classification using ML D:
  data
  flask
    templates
      home.html
      home.html.bak
      predict.html
      predict.html.bak
      submit.html
      submit.html.bak
    app.py
    model.pkl
  training
  training_ibm
External Libraries
Scratches and Consoles
```

We are building a flask application that needs HTML pages stored in

- the templates folder and a python script app.py for scripting.
- Drug Classification.ipynb is the python file where the ML algorithm is applied to the dataset for testing and training. Finally, the model is saved for future use.

- flask integration.
- The data folder contains the CSV file dataset for training our model.
- The training folder contains model training files and the training_ibm folder

contains IBM deployment fles.

DATA COLLECTION

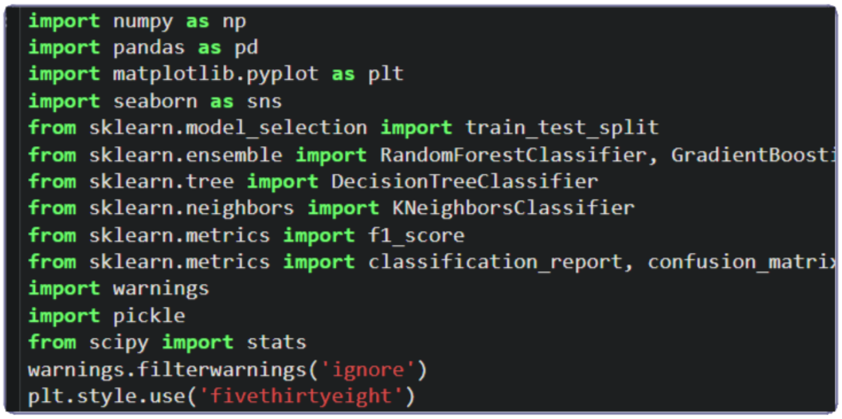
ML depends heavily on data, It is most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

VISUALIZING AND ANALYSING THE DATA

Importing The Libraries

Import the necessary libraries as shown in the image.
Import the required libraries for the model to run. The frst step is usually importing the libraries that will be needed in the program.

- Numpy- It is an open-source numerical Python library. It contains a multi-dimensional array and matrix data structures. It can be used to perform mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.
- Pandas- It is a fast, powerful, fexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
- Seaborn- Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- Matplotlib- Visualisation with python. It is a comprehensive library for creating static,animated, and interactive visualizations in Python
- Sklearn – which contains all the modules required for model building

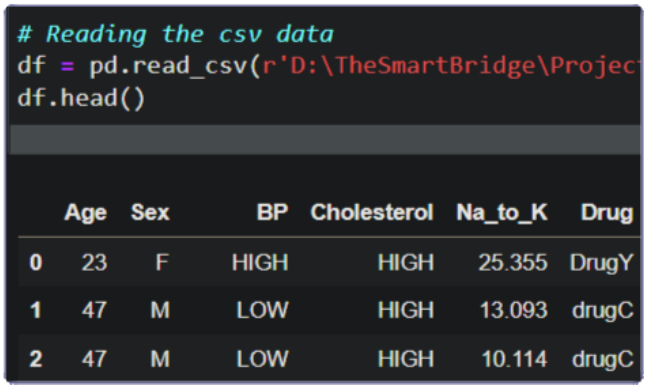


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoosti
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report, confusion_matri
import warnings
import pickle
from scipy import stats
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')
```

Read The Dataset

Our dataset format might be in .csv, excel fles, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of csv fle.

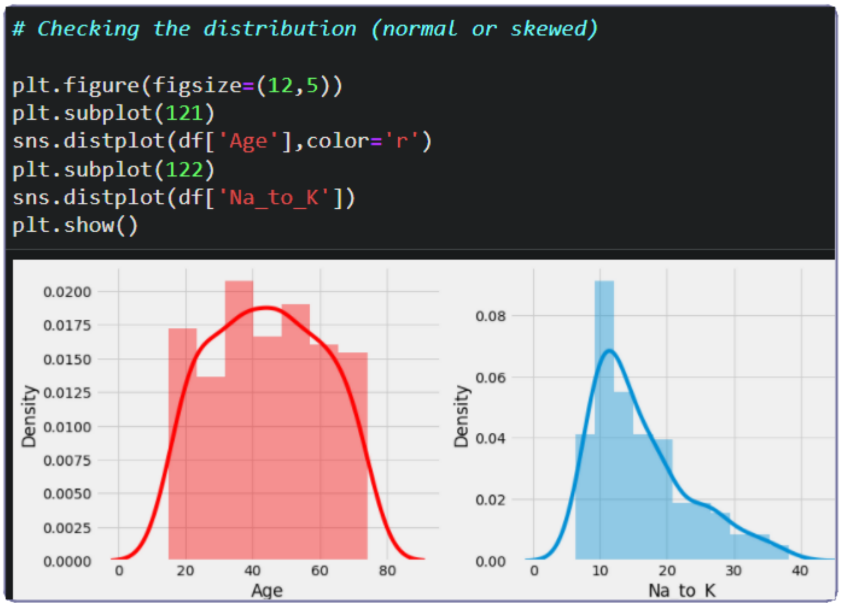


```
# Reading the csv data
df = pd.read_csv(r'D:\TheSmartBridge\Projec
df.head()
```

```
   Age  Sex    BP   Cholesterol  Na_to_K   Drug
0   23    F   HIGH        HIGH    25.355  DrugY
1   47    M    LOW        HIGH    13.093  drugC
2   47    M    LOW        HIGH    10.114  drugC
```

Univariate Analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot and countplot. Seaborn package provides a wonderful function distplot. With the help of distplot, we can fnd the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

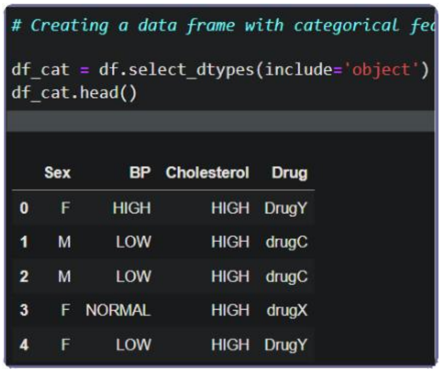


```
# Checking the distribution (normal or skewed)

plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(df['Age'],color='r')
plt.subplot(122)
sns.distplot(df['Na_to_K'])
plt.show()
```

In our dataset we have some categorical features. With the countplot function, we are going to count the unique category in those features. We have created a dummy data frame with categorical features. With for loop and subplot we have plotted this below graph.

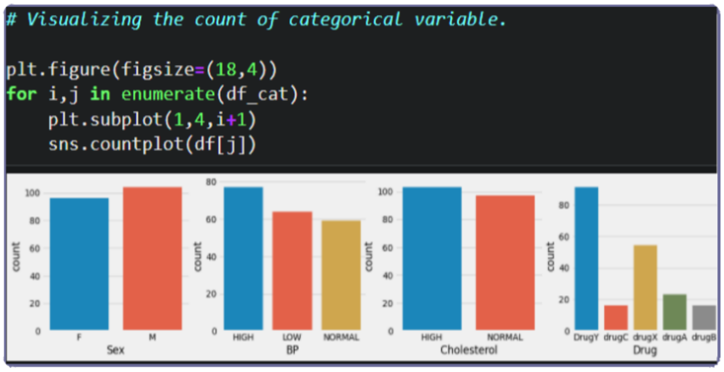
From the plot we came to know, Most of the patients are using drugY and drugX.

And most of the patients have high BP and high Cholesterol.



```
# Creating a data frame with categorical fe

df_cat = df.select_dtypes(include='object')
df_cat.head()
```

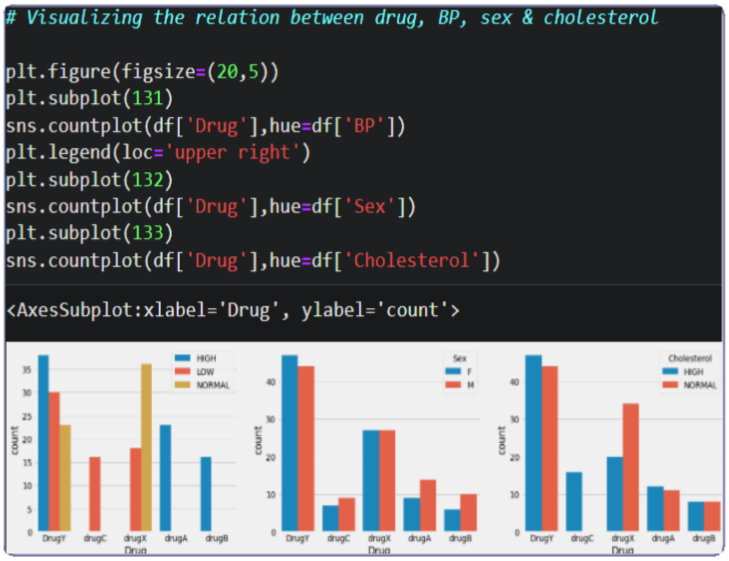
	Sex	BP	Cholesterol	Drug
0	F	HIGH	HIGH	DrugY
1	M	LOW	HIGH	drugC
2	M	LOW	HIGH	drugC
3	F	NORMAL	HIGH	drugX
4	F	LOW	HIGH	DrugY



```
# Visualizing the count of categorical variable.

plt.figure(figsize=(18,4))
for i,j in enumerate(df_cat):
    plt.subplot(1,4,i+1)
    sns.countplot(df[j])
```

Bivariate Analysis

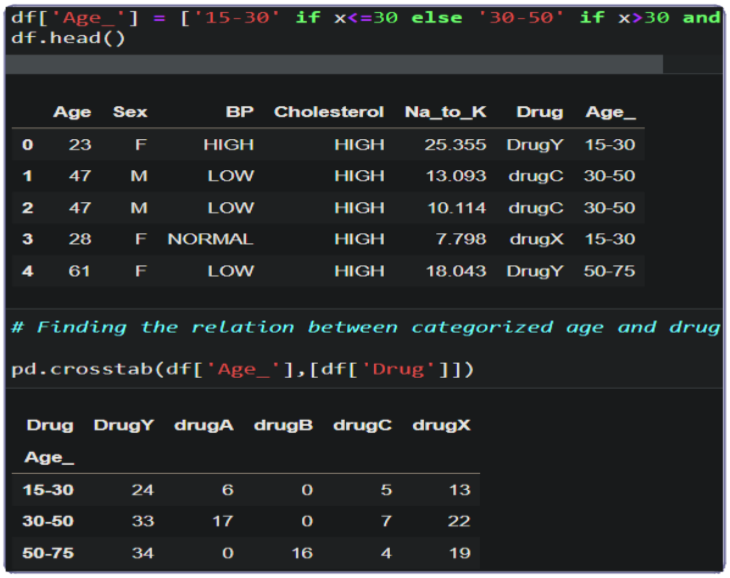
To fnd the relation between two features we use bivariate analysis. Here we are visualizing the relationship between drug & BP, drug & sex and drug & cholesterol. Countplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.
From the below plot you can understand that drugA and drugB is not preferred to low and normal BP patients. DrugC is preferred only to low BP patients.
By third graph we can understand, drugC is not preferred to normal cholesterol patients.



```
# Visualizing the relation between drug, BP, sex & cholesterol

plt.figure(figsize=(20,5))
plt.subplot(131)
sns.countplot(df['Drug'],hue=df['BP'])
plt.legend(loc='upper right')
plt.subplot(132)
sns.countplot(df['Drug'],hue=df['Sex'])
plt.subplot(133)
sns.countplot(df['Drug'],hue=df['Cholesterol'])
```

```
<AxesSubplot:xlabel='Drug', ylabel='count'>
```

With the help of age feature we are creating an age interval and fnding the relation between drug feature and age interval feature. Function crosstab is used to fnd the relationship. From the below image we get a clear understanding, DrugB is preferred only for patients above age 50 years. And drugA is not preferred for patients above age 50 years.



```
df['Age_'] = ['15-30' if x<=30 else '30-50' if x>30 and
df.head()
```

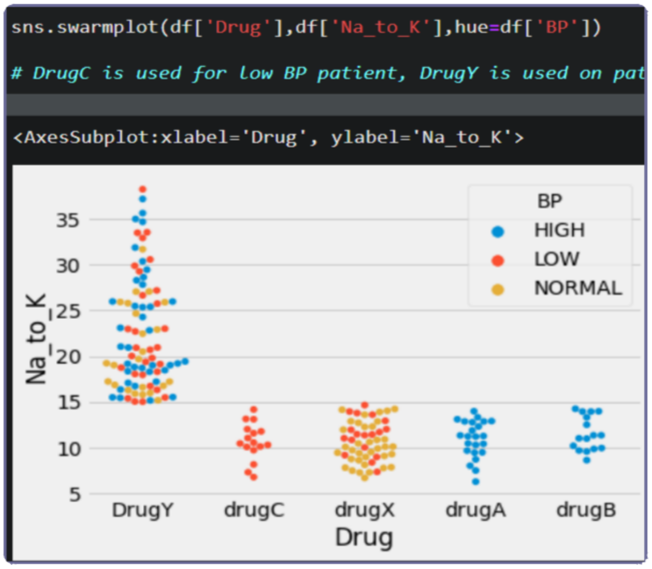
	Age	Sex	BP	Cholesterol	Na_to_K	Drug	Age_
0	23	F	HIGH	HIGH	25.355	DrugY	15-30
1	47	M	LOW	HIGH	13.093	drugC	30-50
2	47	M	LOW	HIGH	10.114	drugC	30-50
3	28	F	NORMAL	HIGH	7.798	drugX	15-30
4	61	F	LOW	HIGH	18.043	DrugY	50-75

```
# Finding the relation between categorized age and drug

pd.crosstab(df['Age_'],[df['Drug']])
```

Drug	DrugY	drugA	drugB	drugC	drugX
Age_					
15-30	24	6	0	5	13
30-50	33	17	0	7	22
50-75	34	0	16	4	19

Multivariate Analysis

In simple words, multivariate analysis is to fnd the relation between multiple features. Here we have used swarmplot from seaborn package.
From the below image, we came to a conclusion that DrugY is used by most of patients who has different BP levels. But It is preferred only for patients having Na_to_K > 15 (Na_to_K – Sodium to potassium ratio on blood).

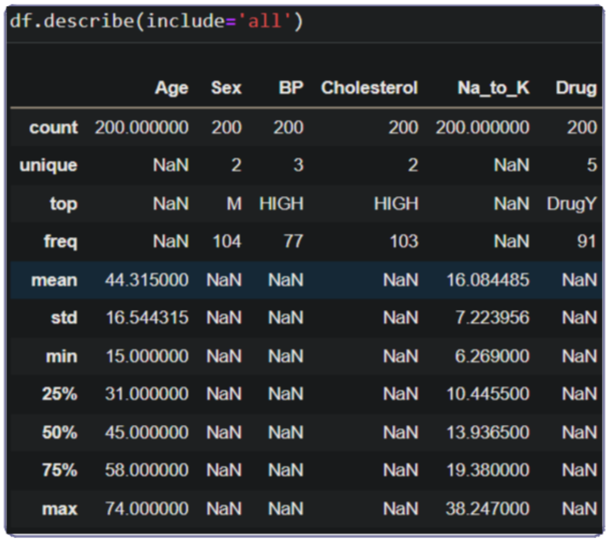


```
sns.swarmplot(df['Drug'],df['Na_to_K'],hue=df['BP'])
# DrugC is used for low BP patient, DrugY is used on pat
```

```
<AxesSubplot:xlabel='Drug', ylabel='Na_to_K'>
```

Descriptive Analysis

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can fnd mean, std, min, max and percentile values of continuous features.



```
df.describe(include='all')
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
count	200.000000	200	200	200	200.000000	200
unique	NaN	2	3	2	NaN	5
top	NaN	M	HIGH	HIGH	NaN	DrugY
freq	NaN	104	77	103	NaN	91
mean	44.315000	NaN	NaN	NaN	16.084485	NaN
std	16.544315	NaN	NaN	NaN	7.223956	NaN
min	15.000000	NaN	NaN	NaN	6.269000	NaN
25%	31.000000	NaN	NaN	NaN	10.445500	NaN
50%	45.000000	NaN	NaN	NaN	13.936500	NaN
75%	58.000000	NaN	NaN	NaN	19.380000	NaN
max	74.000000	NaN	NaN	NaN	38.247000	NaN

DATA PRE-PROCESSING

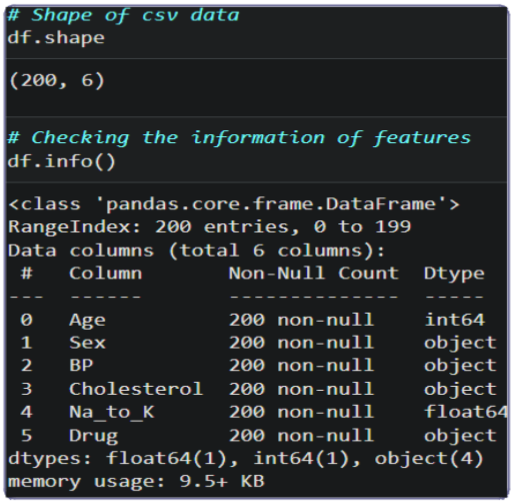
As we have understood how the data is lets pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling outliers
- Splitting dataset into training and test set

Checking For Null Values

Let’s fnd the shape of our dataset frst, To fnd the shape of our data, df.shape method is used. To fnd the data type, df.info() function is used.



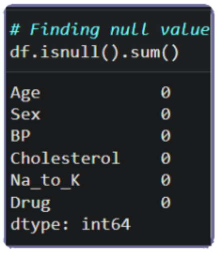
```
# Shape of csv data
df.shape
```

```
(200, 6)
```

```
# Checking the information of features
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Age          200 non-null    int64
 1   Sex          200 non-null    object
 2   BP           200 non-null    object
 3   Cholesterol  200 non-null    object
 4   Na_to_K      200 non-null    float64
 5   Drug         200 non-null    object
dtypes: float64(1), int64(1), object(4)
memory usage: 9.5+ KB
```

For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function to it. From the below image we found that there are no null values present in our dataset. So we can skip handling of missing values step.



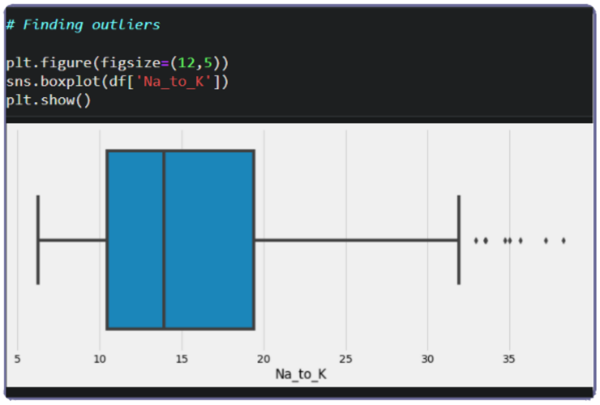
```
# Finding null value
df.isnull().sum()
```

```
Age            0
Sex            0
BP             0
Cholesterol    0
Na_to_K        0
Drug           0
dtype: int64
```

Let’s look for any outliers in the dataset

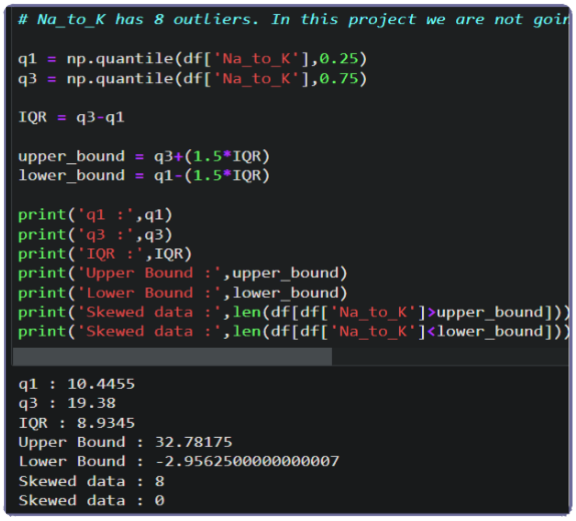
Handling Outliers

With the help of boxplot, outliers are visualized. And here we are going to fnd upper bound and lower bound of Na_to_K feature with some mathematical formula.
From the below diagram, we could visualize that Na_to_K feature has outliers.
Boxplot from seaborn library is used here.



```
# Finding outliers

plt.figure(figsize=(12,5))
sns.boxplot(df['Na_to_K'])
plt.show()
```

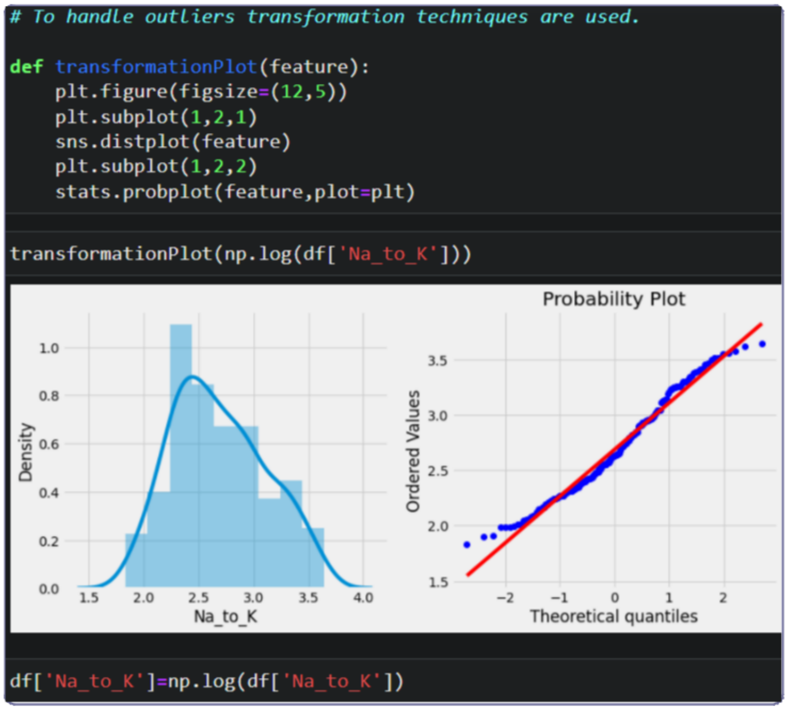
To fnd upper bound we have to multiply IQR (Interquartile range) with 1.5 and add it with 3rd quantile. To fnd lower bound instead of adding, subtract it with 1st quantile. Take image attached below as your reference.



```
# Na_to_K has 8 outliers. In this project we are not goir

q1 = np.quantile(df['Na_to_K'],0.25)
q3 = np.quantile(df['Na_to_K'],0.75)

IQR = q3-q1

upper_bound = q3+(1.5*IQR)
lower_bound = q1-(1.5*IQR)

print('q1 :',q1)
print('q3 :',q3)
print('IQR :',IQR)
print('Upper Bound :',upper_bound)
print('Lower Bound :',lower_bound)
print('Skewed data :',len(df[df['Na_to_K']>upper_bound]))
print('Skewed data :',len(df[df['Na_to_K']<lower_bound]))
```

```
q1 : 10.4455
q3 : 19.38
IQR : 8.9345
Upper Bound : 32.78175
Lower Bound : -2.9562500000000007
Skewed data : 8
Skewed data : 0
```

To handle the outliers transformation technique is used. Here log transformation is used. We have created a function to visualize the distribution and probability plot of Na_to_K feature.



```
# To handle outliers transformation techniques are used.

def transformationPlot(feature):
    plt.figure(figsize=(12,5))
    plt.subplot(1,2,1)
    sns.distplot(feature)
    plt.subplot(1,2,2)
    stats.probplot(feature,plot=plt)
```

```
transformationPlot(np.log(df['Na_to_K']))
```

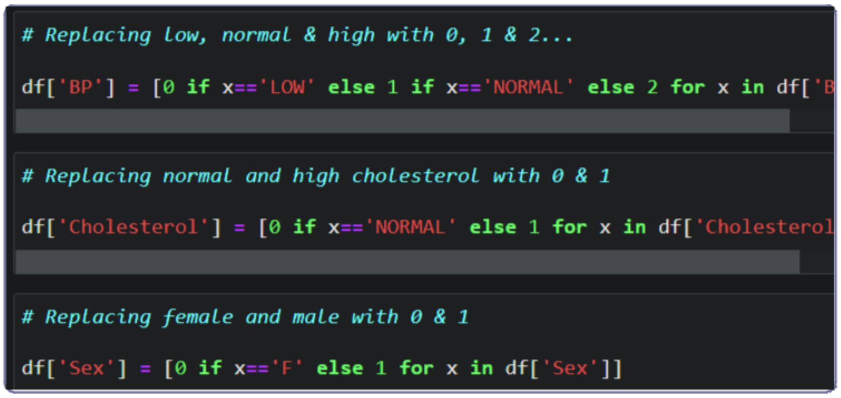
```
df['Na_to_K']=np.log(df['Na_to_K'])
```

Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using manual encoding with the help of list comprehension.

In our project, categorical features are BP, Cholesterol and sex. With list comprehension encoding is done.



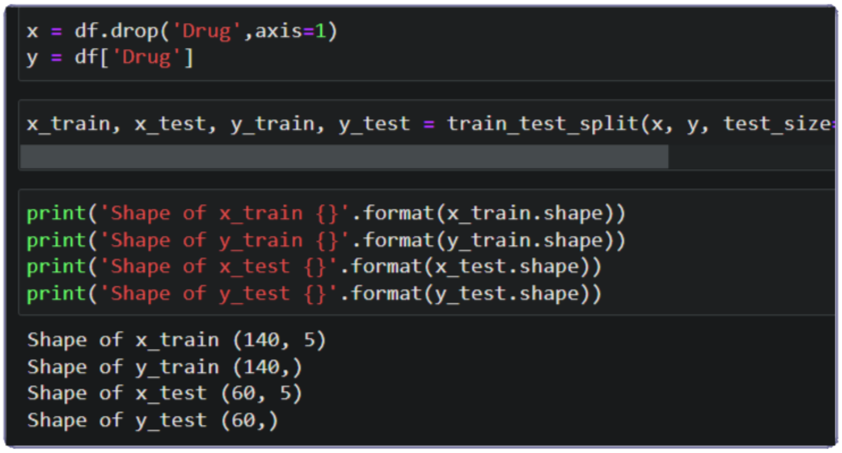
```
# Replacing low, normal & high with 0, 1 & 2...

df['BP'] = [0 if x=='LOW' else 1 if x=='NORMAL' else 2 for x in df['B
```

```
# Replacing normal and high cholesterol with 0 & 1

df['Cholesterol'] = [0 if x=='NORMAL' else 1 for x in df['Cholesterol
```

```
# Replacing female and male with 0 & 1

df['Sex'] = [0 if x=='F' else 1 for x in df['Sex']]
```

Splitting Data Into Train And Test

Now let’s split the Dataset into train and test sets
Changes: frst split the dataset into x and y and then split the data set
Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.



```
x = df.drop('Drug',axis=1)
y = df['Drug']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size
```

```
print('Shape of x_train {}'.format(x_train.shape))
print('Shape of y_train {}'.format(y_train.shape))
print('Shape of x_test {}'.format(x_test.shape))
print('Shape of y_test {}'.format(y_test.shape))
```

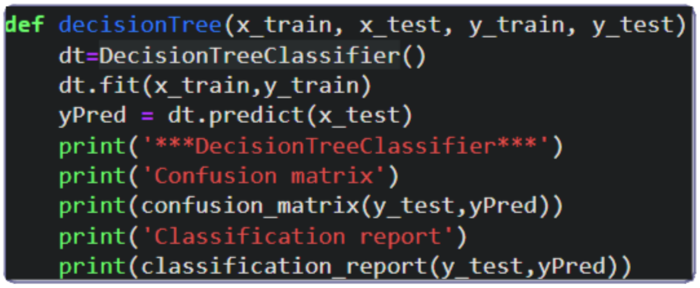
```
Shape of x_train (140, 5)
Shape of y_train (140,)
Shape of x_test (60, 5)
Shape of y_test (60,)
```

MODEL BUILDING

Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project we are applying four classifcation algorithms. The best model is saved based on its performance.

Decision Tree Model

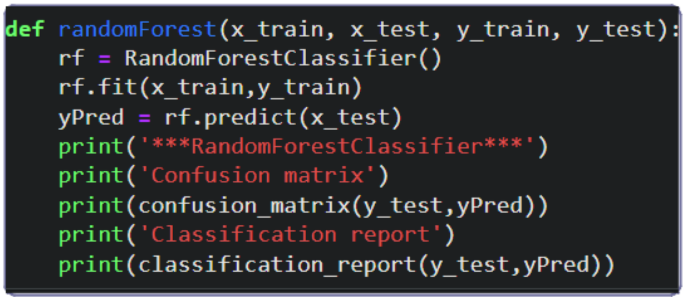
A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifer algorithm is initialized and training data is passed to the model with .ft() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classifcation report is done.



```
def decisionTree(x_train, x_test, y_train, y_test)
    dt=DecisionTreeClassifier()
    dt.fit(x_train,y_train)
    yPred = dt.predict(x_test)
    print('***DecisionTreeClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
```

Random Forest Model

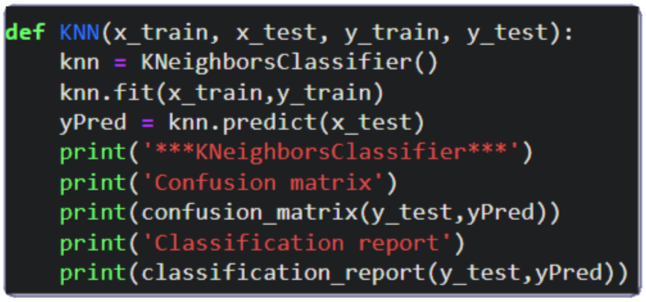
A function named randomForest is created and train and test data are passed as the parameters. Inside the function, RandomForestClassifer algorithm is initialized and training data is passed to the model with .ft() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classifcation report is done.



```
def randomForest(x_train, x_test, y_train, y_test):
    rf = RandomForestClassifier()
    rf.fit(x_train,y_train)
    yPred = rf.predict(x_test)
    print('***RandomForestClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
```

KNN Model

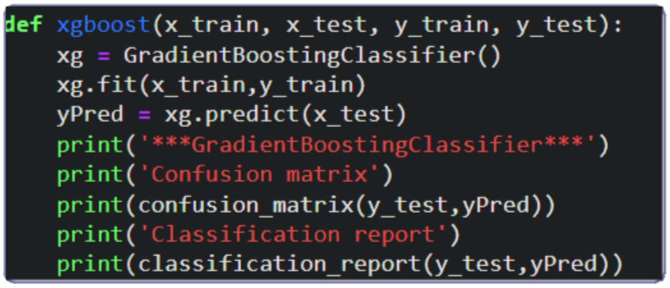
A function named KNN is created and train and test data are passed as the parameters. Inside the function, KNeighborsClassifer algorithm is initialized and training data is passed to the model with .ft() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classifcation report is done.



```python
def KNN(x_train, x_test, y_train, y_test):
    knn = KNeighborsClassifier()
    knn.fit(x_train,y_train)
    yPred = knn.predict(x_test)
    print('***KNeighborsClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
```

Xgboost Model

A function named xgboost is created and train and test data are passed as the parameters. Inside the function, GradientBoostingClassifer algorithm is initialized and training data is passed to the model with .ft() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classifcation report is done.

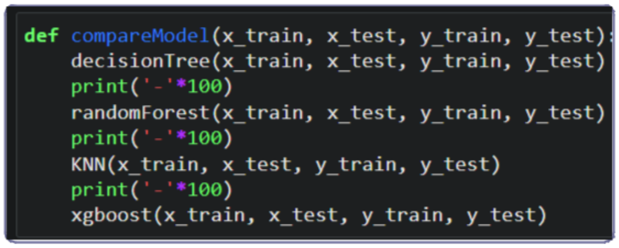


```python
def xgboost(x_train, x_test, y_train, y_test):
    xg = GradientBoostingClassifier()
    xg.fit(x_train,y_train)
    yPred = xg.predict(x_test)
    print('***GradientBoostingClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
```

Now let's see the performance of all the models and save the best model

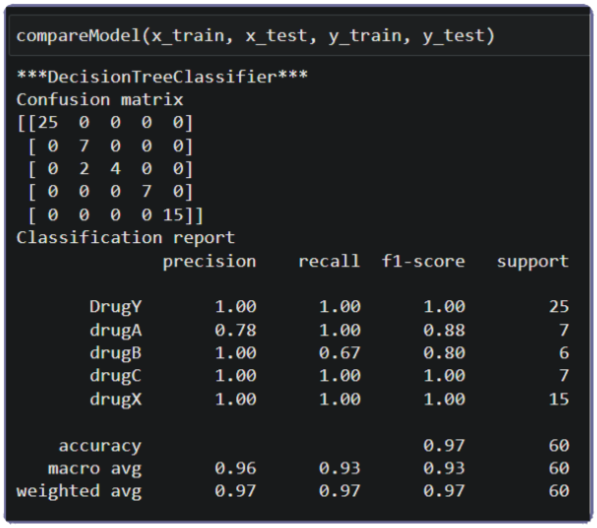
Compare The Model

For comparing the above four models compareModel function is defned.

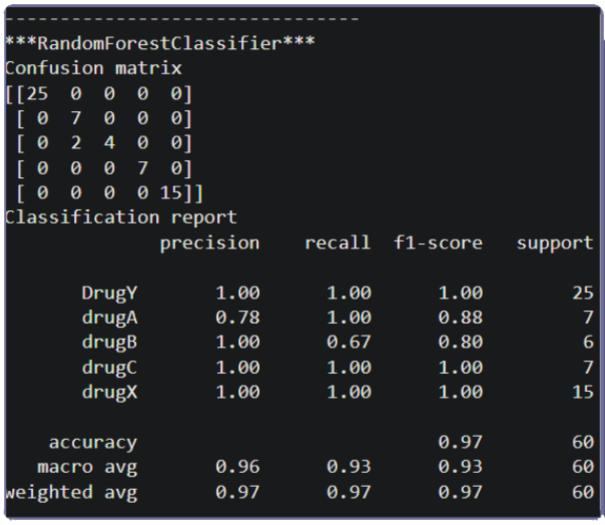


```python
def compareModel(x_train, x_test, y_train, y_test):
    decisionTree(x_train, x_test, y_train, y_test)
    print('-'*100)
    randomForest(x_train, x_test, y_train, y_test)
    print('-'*100)
    KNN(x_train, x_test, y_train, y_test)
    print('-'*100)
    xgboost(x_train, x_test, y_train, y_test)
```

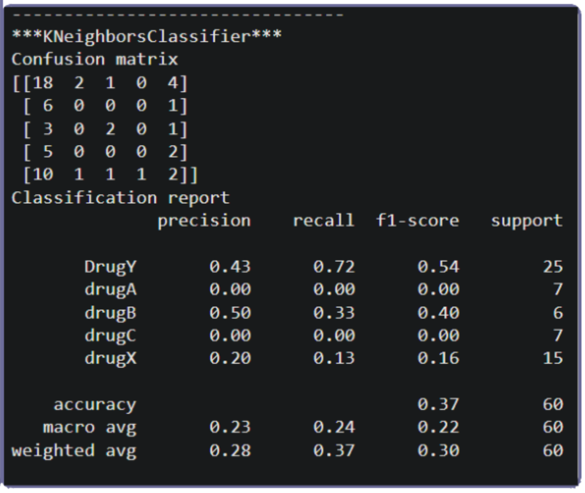
After calling the function, the results of models are displayed as output. From the four model random forest and decision tree is performing well. From the below image, We can see the accuracy of the model. Both models have 97% accuracy. Even confusion matrix also have same results. Training time of decision tree is faster than random forest. In such case we have to select decision tree model (time saving & cost wise proftable). But, here random forest is selected and evaluated with cross validation. Additionally, we can tune the model with hyper parameter tuning techniques.



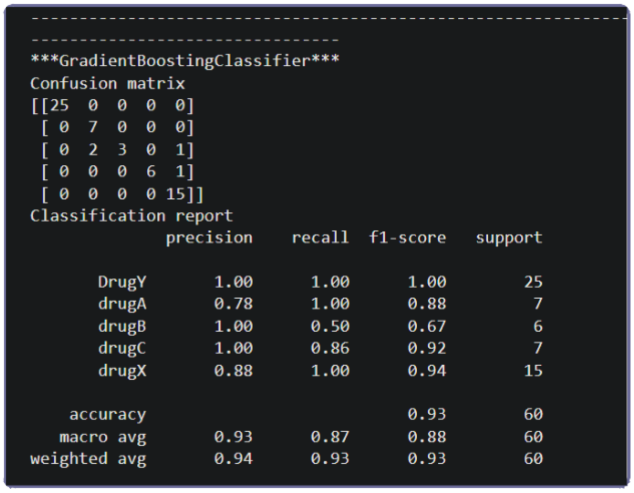
```
compareModel(x_train, x_test, y_train, y_test)

***DecisionTreeClassifier***
Confusion matrix
[[25  0  0  0  0]
 [ 0  7  0  0  0]
 [ 0  2  4  0  0]
 [ 0  0  0  7  0]
 [ 0  0  0  0 15]]
Classification report
              precision    recall  f1-score   support

       DrugY       1.00      1.00      1.00        25
       drugA       0.78      1.00      0.88         7
       drugB       1.00      0.67      0.80         6
       drugC       1.00      1.00      1.00         7
       drugX       1.00      1.00      1.00        15

    accuracy                           0.97        60
   macro avg       0.96      0.93      0.93        60
weighted avg       0.97      0.97      0.97        60
```



```
-------------------------------
***RandomForestClassifier***
Confusion matrix
[[25  0  0  0  0]
 [ 0  7  0  0  0]
 [ 0  2  4  0  0]
 [ 0  0  0  7  0]
 [ 0  0  0  0 15]]
Classification report
              precision    recall  f1-score   support

       DrugY       1.00      1.00      1.00        25
       drugA       0.78      1.00      0.88         7
       drugB       1.00      0.67      0.80         6
       drugC       1.00      1.00      1.00         7
       drugX       1.00      1.00      1.00        15

    accuracy                           0.97        60
   macro avg       0.96      0.93      0.93        60
weighted avg       0.97      0.97      0.97        60
```



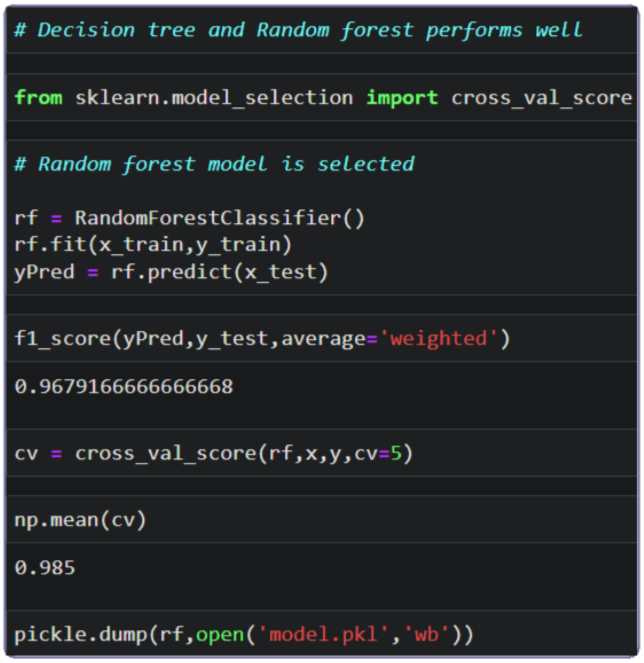
```
-------------------------------
***KNeighborsClassifier***
Confusion matrix
[[18  2  1  0  4]
 [ 6  0  0  0  1]
 [ 3  0  2  0  1]
 [ 5  0  0  0  2]
 [10  1  1  1  2]]
Classification report
              precision    recall  f1-score   support

       DrugY       0.43      0.72      0.54        25
       drugA       0.00      0.00      0.00         7
       drugB       0.50      0.33      0.40         6
       drugC       0.00      0.00      0.00         7
       drugX       0.20      0.13      0.16        15

    accuracy                           0.37        60
   macro avg       0.23      0.24      0.22        60
weighted avg       0.28      0.37      0.30        60
```



```
------------------------------------------------------------
--------------------------------
***GradientBoostingClassifier***
Confusion matrix
[[25  0  0  0  0]
 [ 0  7  0  0  0]
 [ 0  2  3  0  1]
 [ 0  0  0  6  1]
 [ 0  0  0  0 15]]
Classification report
              precision    recall  f1-score   support

       DrugY       1.00      1.00      1.00        25
       drugA       0.78      1.00      0.88         7
       drugB       1.00      0.50      0.67         6
       drugC       1.00      0.86      0.92         7
       drugX       0.88      1.00      0.94        15

    accuracy                           0.93        60
   macro avg       0.93      0.87      0.88        60
weighted avg       0.94      0.93      0.93        60
```

Evaluating Performance Of The Model And Saving The Model

From sklearn, cross_val_score is used to evaluate the score of the model. On the parameters, we have given rf (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model by pickle.dump(). Note: To understand cross validation



```
# Decision tree and Random forest performs well

from sklearn.model_selection import cross_val_score

# Random forest model is selected

rf = RandomForestClassifier()
rf.fit(x_train,y_train)
yPred = rf.predict(x_test)

f1_score(yPred,y_test,average='weighted')

0.9679166666666668

cv = cross_val_score(rf,x,y,cv=5)

np.mean(cv)

0.985

pickle.dump(rf,open('model.pkl','wb'))
```

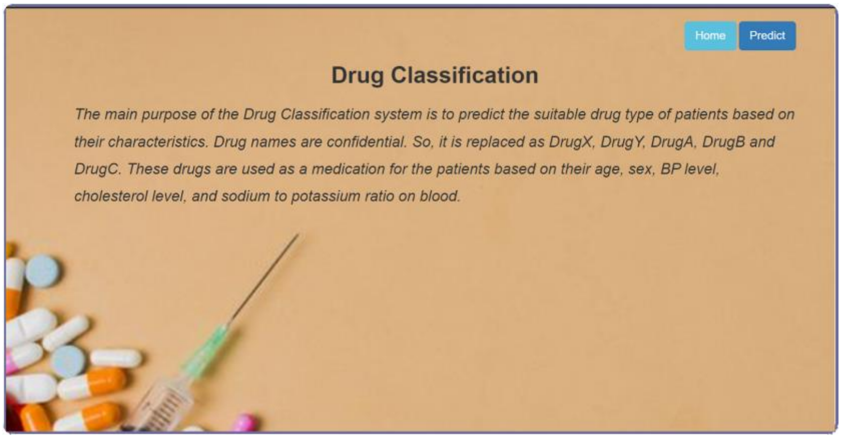
APPLICATION BUILDING

Building Html Pages

For this project create three HTML fles namely

- home.html
- predict.html
- submit.html

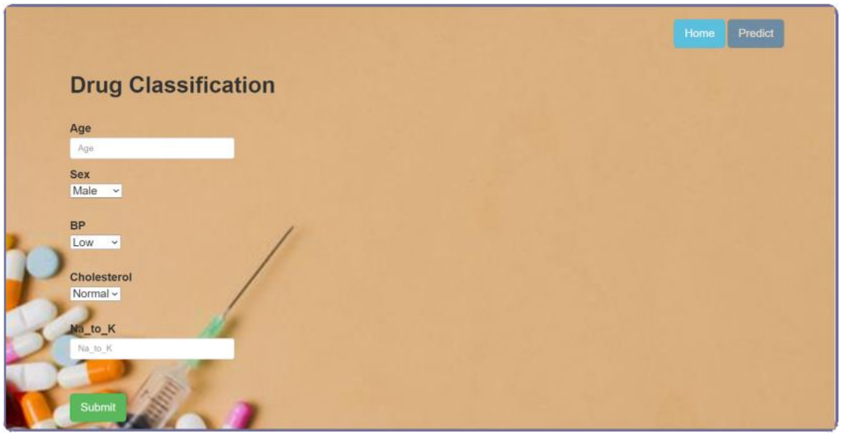
and save them in templates folder.
Let’s see how our home.html page looks like:



Drug Classification

The main purpose of the Drug Classification system is to predict the suitable drug type of patients based on their characteristics. Drug names are confidential. So, it is replaced as DrugX, DrugY, DrugA, DrugB and DrugC. These drugs are used as a medication for the patients based on their age, sex, BP level, cholesterol level, and sodium to potassium ratio on blood.

Now when you click on predict button from top right corner you will get redirected to predict.html
Lets look how our predict.html fle looks like:



Drug Classification

Age
Age

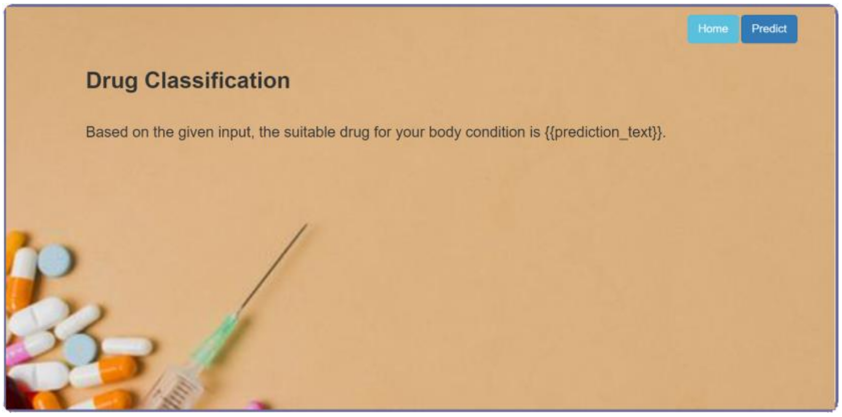
Sex
Male

BP
Low

Cholesterol
Normal

Na_to_K
Na_to_K

Now when you click on submit button from left bottom corner you will get redirected to submit.html
Lets look how our submit.html fle looks like:

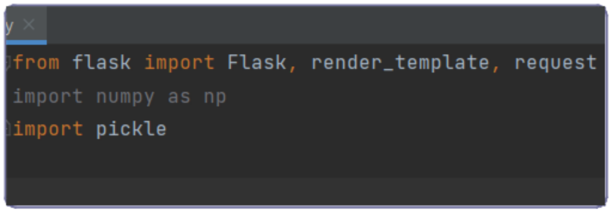


Drug Classification

Based on the given input, the suitable drug for your body condition is {{prediction_text}}.

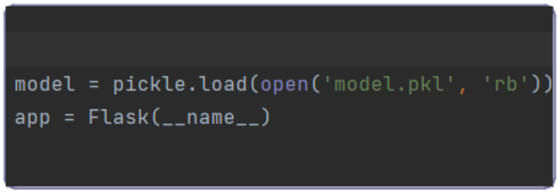
Build Python Code

Import the libraries



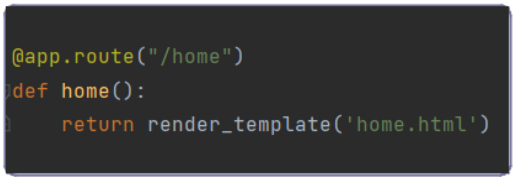
```
from flask import Flask, render_template, request
import numpy as np
import pickle
```

Load the saved model. Importing fask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.



```
model = pickle.load(open('model.pkl', 'rb'))
app = Flask(__name__)
```

Render HTML page:

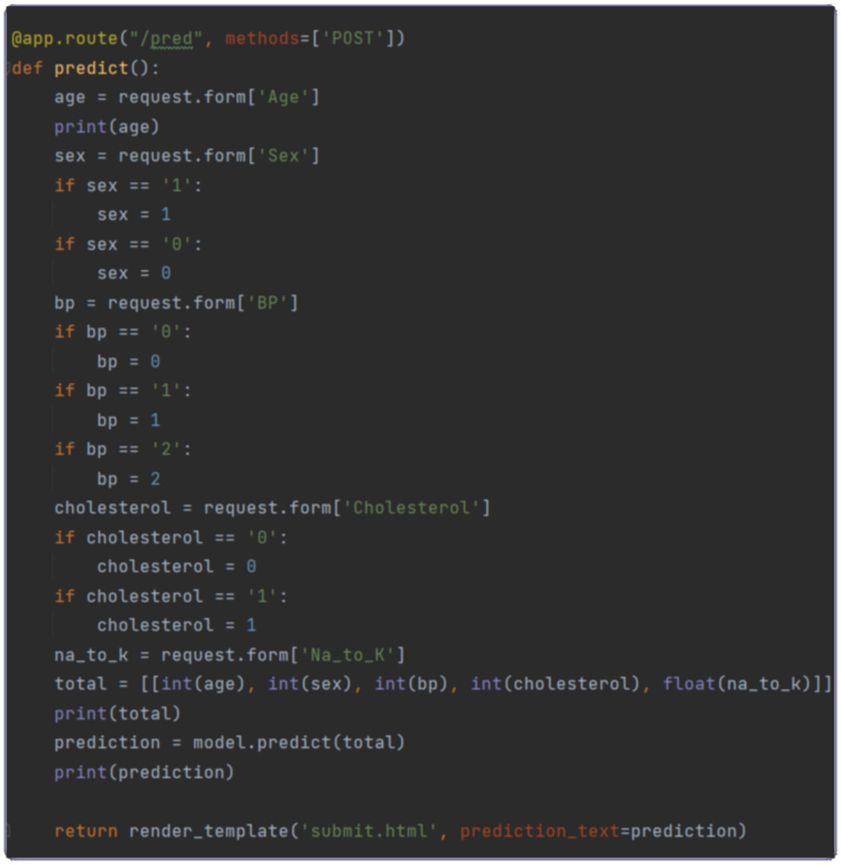


```
@app.route("/home")
def home():
    return render_template('home.html')
```

Here we will be using declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with home.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

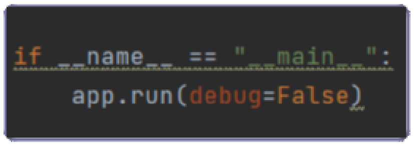
Retrieves the value from UI:



```
@app.route("/pred", methods=['POST'])
def predict():
    age = request.form['Age']
    print(age)
    sex = request.form['Sex']
    if sex == '1':
        sex = 1
    if sex == '0':
        sex = 0
    bp = request.form['BP']
    if bp == '0':
        bp = 0
    if bp == '1':
        bp = 1
    if bp == '2':
        bp = 2
    cholesterol = request.form['Cholesterol']
    if cholesterol == '0':
        cholesterol = 0
    if cholesterol == '1':
        cholesterol = 1
    na_to_k = request.form['Na_to_K']
    total = [[int(age), int(sex), int(bp), int(cholesterol), float(na_to_k)]]
    print(total)
    prediction = model.predict(total)
    print(prediction)

    return render_template('submit.html', prediction_text=prediction)
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:

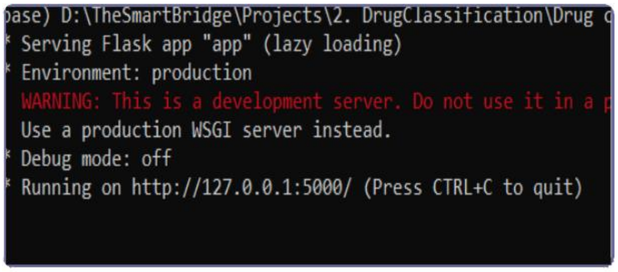


```
if __name__ == "__main__":
    app.run(debug=False)
```

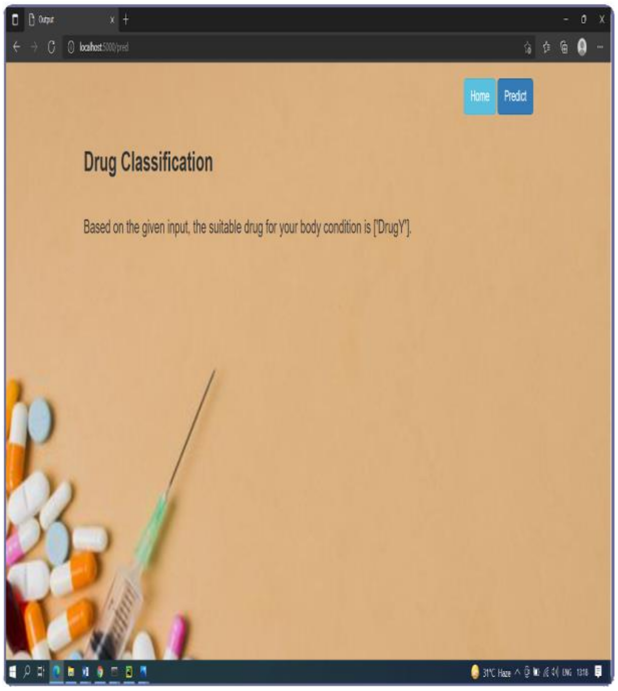
Run The Application

Open anaconda prompt from the start menu
Navigate to the folder where your python script is.
Now type “python app.py” command
Navigate to the localhost where you can view your web page.

Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



```
base) D:\TheSmartBridge\Projects\2. DrugClassification\Drug c
 Serving Flask app "app" (lazy loading)
 Environment: production
  WARNING: This is a development server. Do not use it in a p
  Use a production WSGI server instead.
 Debug mode: off
 Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



Drug Classification

Based on the given input, the suitable drug for your body condition is ['DrugY'].