Drug Classification Using ML

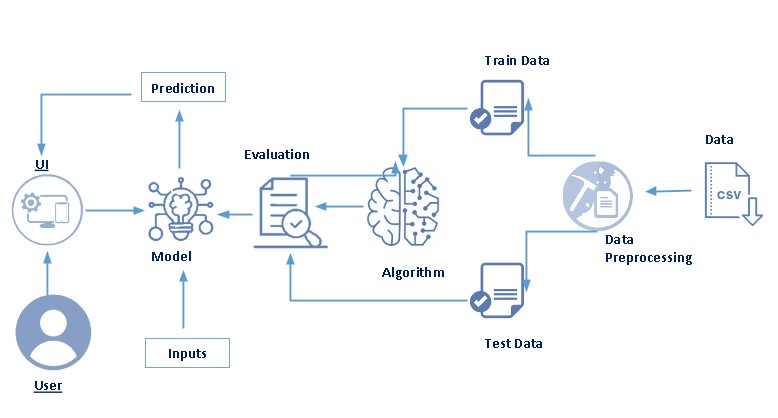
# Project Description:

Nowadays our lifestyle has been changing. Per family, at least one person has Motorcycles or cars, etc. 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 sufficient nutrition and suffering from health issues. To overcome, 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 confidential. 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 Classification system is to predict the suitable drug type confidently 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 classification 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 flask integration and IBM deployment.

# Technical Architecture:



# Pre requisites:

**To complete this project, you must required following software’s, concepts and packages**

* **Anaconda navigator and pycharm:**
  + Refer the link below to download anaconda navigator
  + Link : <https://youtu.be/1ra4zH2G4o0>
* **Python 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.

# Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

* **ML Concepts**
  + Supervised learning: <https://www.javatpoint.com/supervised-machine-learning>
  + Unsupervised learning: <https://www.javatpoint.com/unsupervised-machine-learning>
  + Regression and classification
  + Decision tree: <https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm>
  + Random forest: <https://www.javatpoint.com/machine-learning-random-forest-algorithm>
  + KNN: <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
  + Xgboost: <https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/>
  + Evaluation metrics: <https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/>
* **Flask Basics** : <https://www.youtube.com/watch?v=lj4I_CvBnt0>

# Project Objectives:

By the end of this project you will:

* Know fundamental concepts and techniques used for machine learning.
* Gain a broad understanding about data.
* Have knowledge on pre-processing the data/transformation techniques on outlier and some visualization concepts.

# Project Flow:

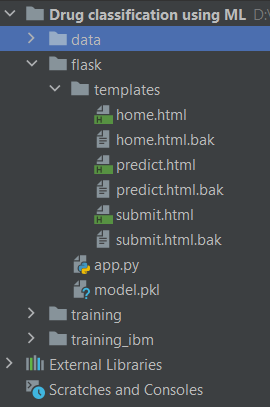
* User interacts with the UI to enter the input.
* Entered input is analyzed by the model which is integrated.
* Once model analyses the input the prediction is showcased on the UI

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

* Data collection
  + Collect the dataset or create the dataset
* Visualizing and analyzing data
  + 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 performance of model
  + Save the model
* Application Building
  + Create an HTML file
  + Build python code

# Project Structure:

Create the Project folder which contains files as shown below



* We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
* Model.pkl is our saved model. Further we will use this model for flask integration.
* Training folder contains model training files and training\_ibm folder contains IBM deployment files.

# Milestone 1: 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.

**Activity 1: Download the dataset**

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used drug200.csv data. This data is downloaded from kaggle.com. Please refer the link given below to download the dataset.

Link: <https://www.kaggle.com/prathamtripathi/drug-classification>

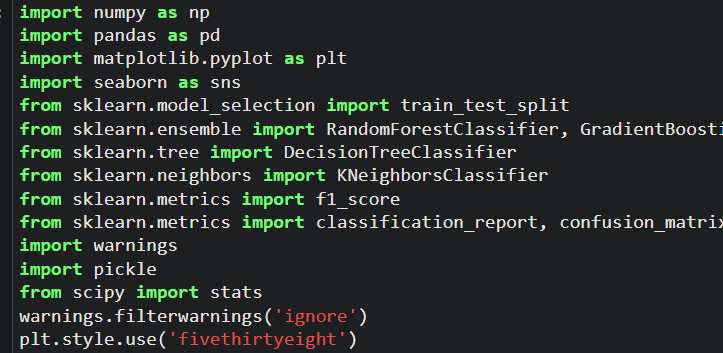
# Milestone 2: Visualizing and analysing the data

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analysing techniques.

**Note: There is n number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.**

**Activity 1: Importing the libraries**

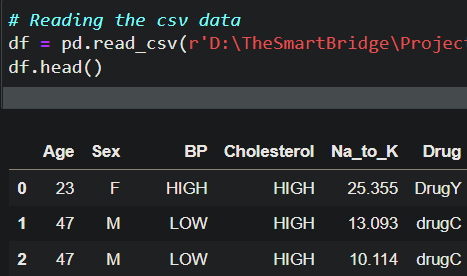
Import the necessary libraries as shown in the image. (optional) Here we have used visualization style as fivethirtyeight.



**Activity 2: Read the Dataset**

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

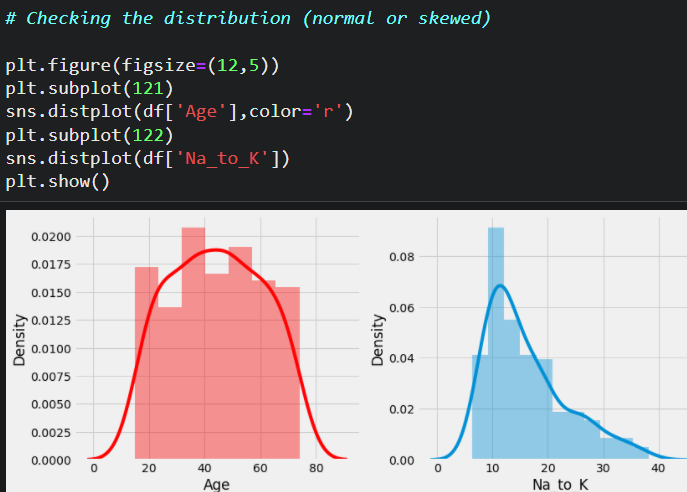
In pandas we have a function called read\_csv() to read the dataset. As a parameter we have to give the directory of csv file.



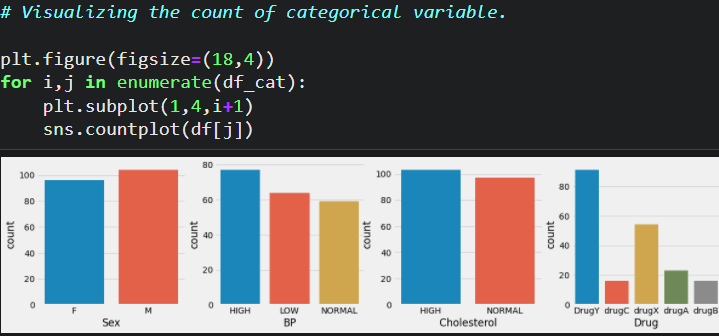
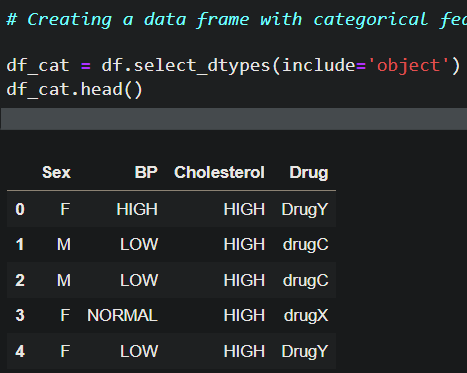
**Activity 3: Univariate analysis**

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 find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.



* 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.



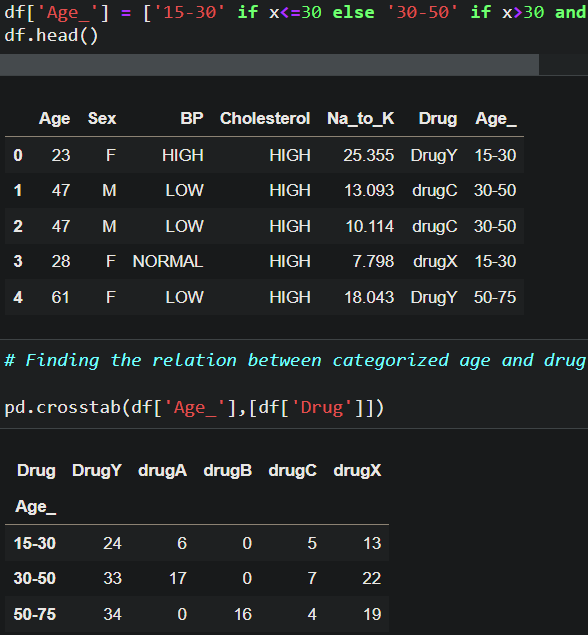
**Activity 4: Bivariate analysis**

To find 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.



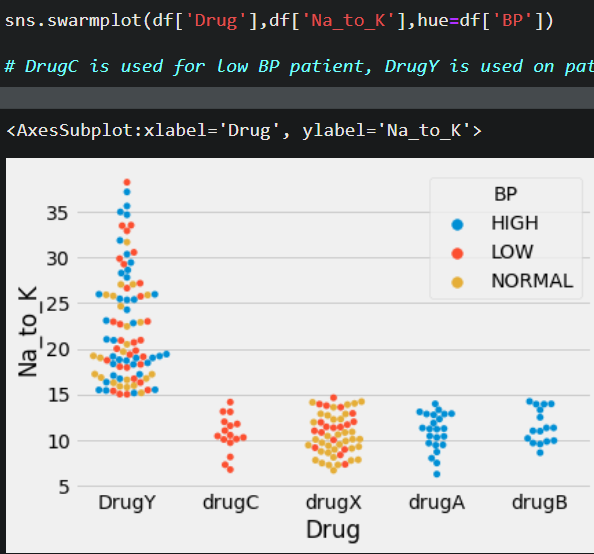
* With the help of age feature we are creating an age interval and finding the relation between drug feature and age interval feature. Function crosstab is used to find the relationship. From the below image we get a clear understanding, DrugB is preferred only for patients above age 50 years. And drugA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



**Activity 5: Multivariate analysis**

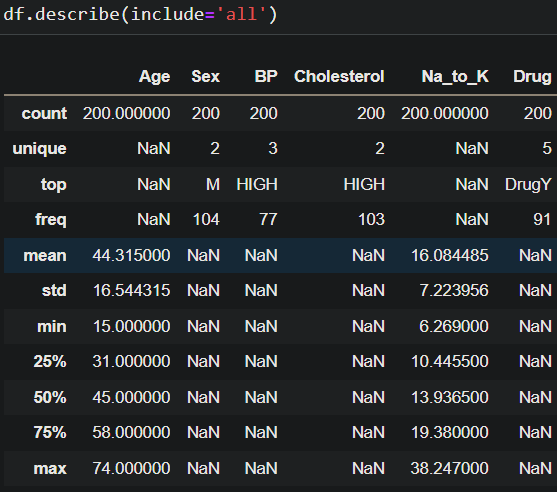
In simple words, multivariate analysis is to find the relation between multiple features. Here we have used swarmplot from seaborn package.

* 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).



**Activity 6: 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 find mean, std, min, max and percentile values of continuous features.



# Milestone 3: Data Pre-processing

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

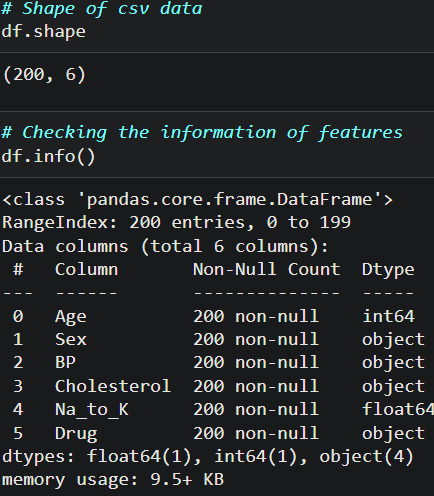
The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

* Handling missing values
* Handling categorical data
* Handling outliers
* Scaling Techniques
* Splitting dataset into training and test set

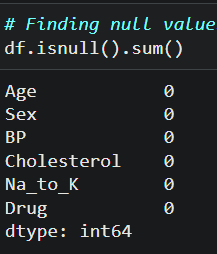
Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

**Activity 1: Checking for null values**

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

****

* 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.

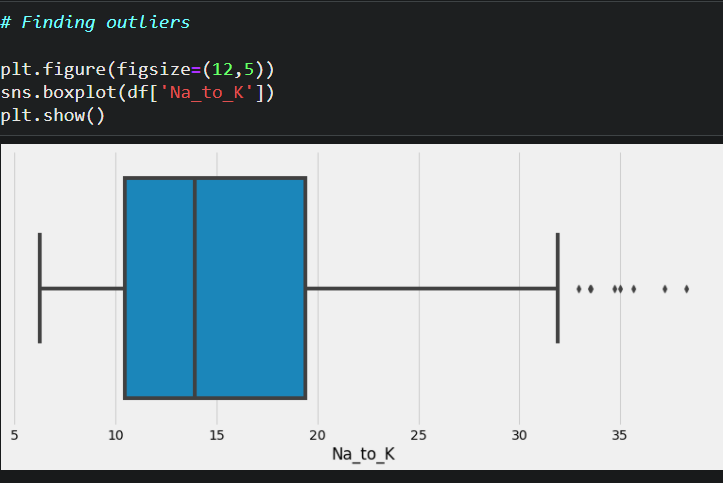
****

Let’s look for any outliers in the dataset

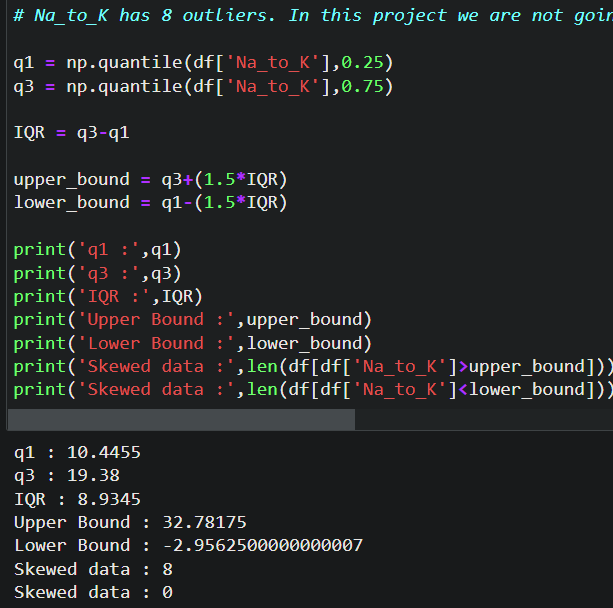
**Activity 2: Handling outliers**

With the help of boxplot, outliers are visualized. And here we are going to find upper bound and lower bound of Na\_to\_K feature with some mathematical formula.

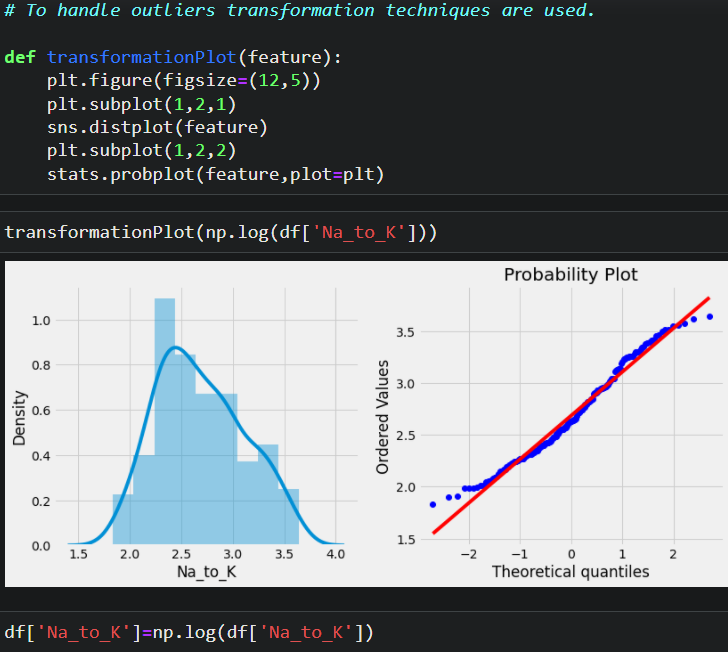
* From the below diagram, we could visualize that Na\_to\_K feature has outliers. Boxplot from seaborn library is used here.



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



* 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.

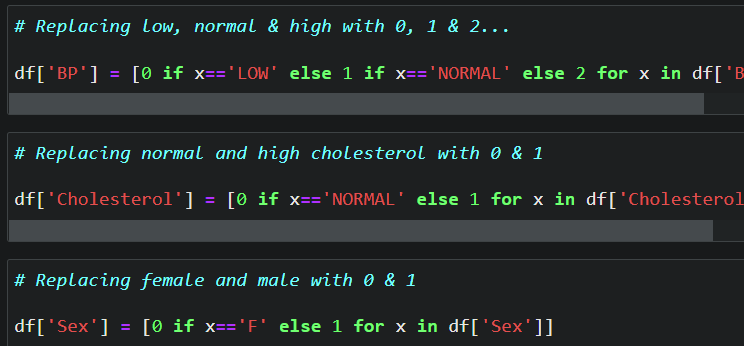


**Activity 3: Handling Categorical Values**

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.

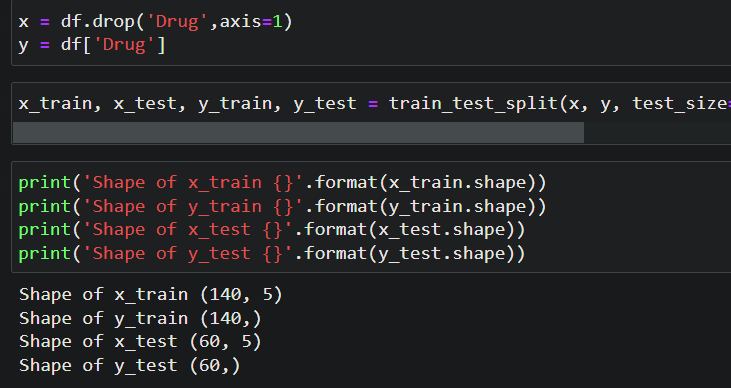


**Activity 4: Splitting data into train and test**

Now let’s split the Dataset into train and test sets

Changes: first 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.

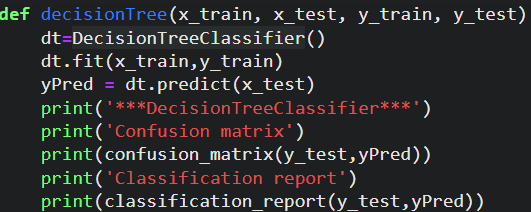


# Milestone 4: Model Building

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 classification algorithms. The best model is saved based on its performance.

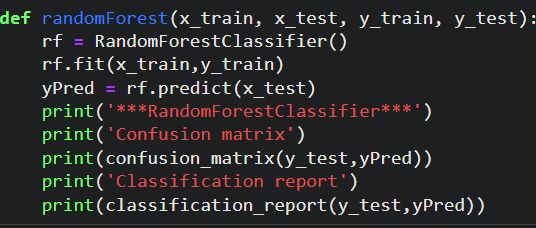
**Activity 1: Decision tree model**

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



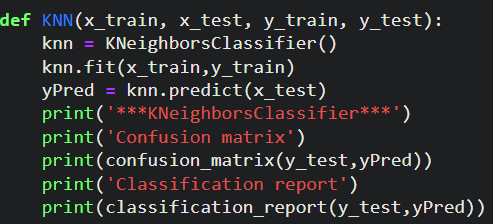
**Activity 2: Random forest model**

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



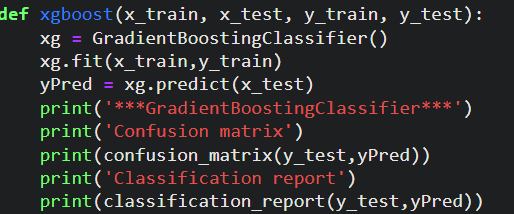
**Activity 3: KNN model**

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



**Activity 4: Xgboost model**

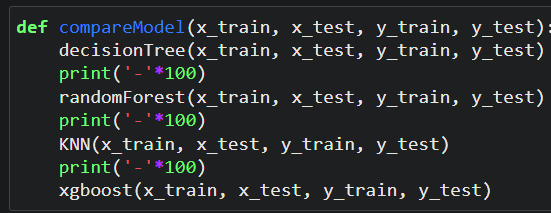
A function named xgboost is created and train and test data are passed as the parameters. Inside the function, GradientBoostingClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable.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



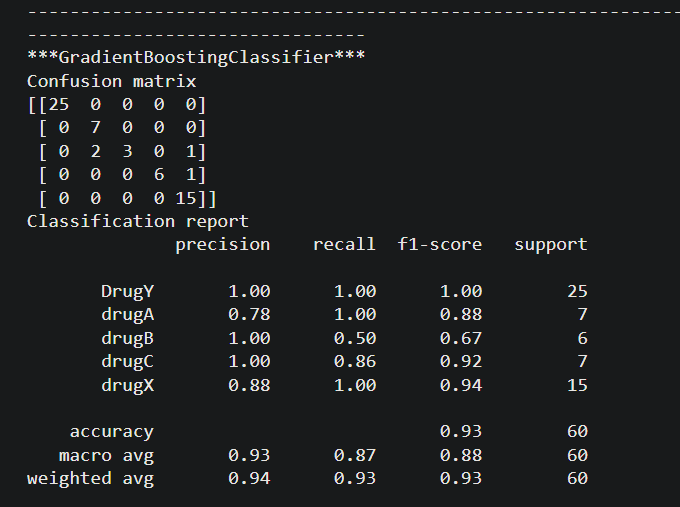
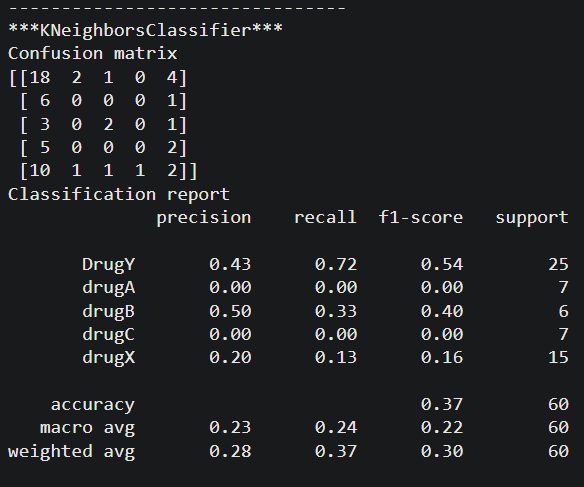
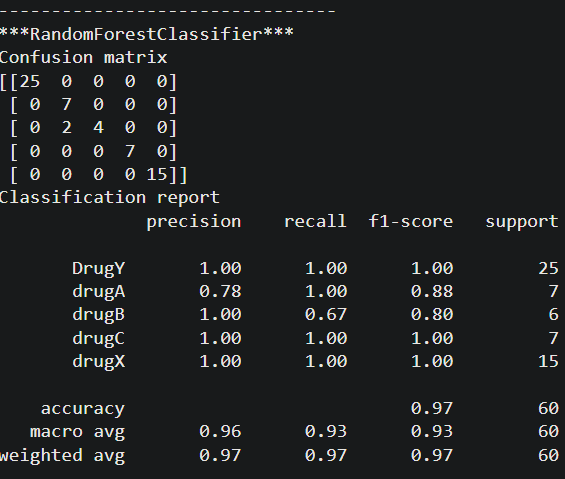
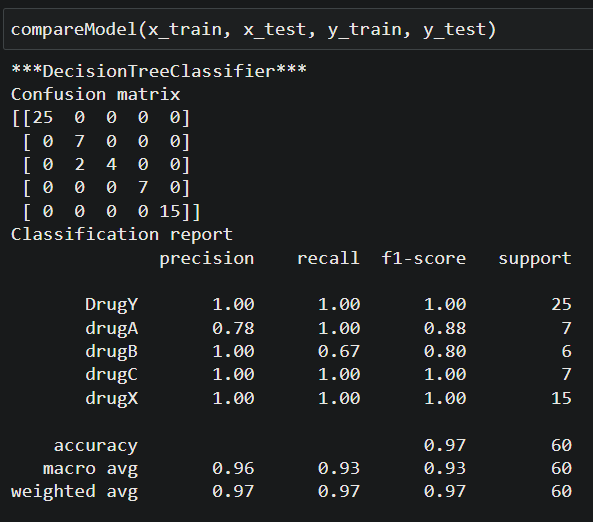
Now let’s see the performance of all the models and save the best model

**Activity 5: Compare the model**

For comparing the above four models compareModel function is defined.



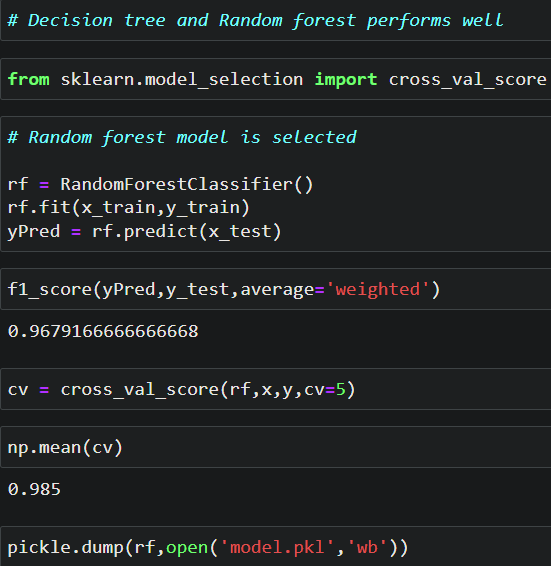
After calling the function, the results of models are displayed as output. From the four model random forest and decision tree is performing well. From the below image, We can see the accuracy of the model. Both models have 97% accuracy. Even confusion matrix also have same results. Training time of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



**Activity 6: 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, refer this link. <https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85>.



# Milestone 5: Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

* Building HTML Pages
* Building serverside script

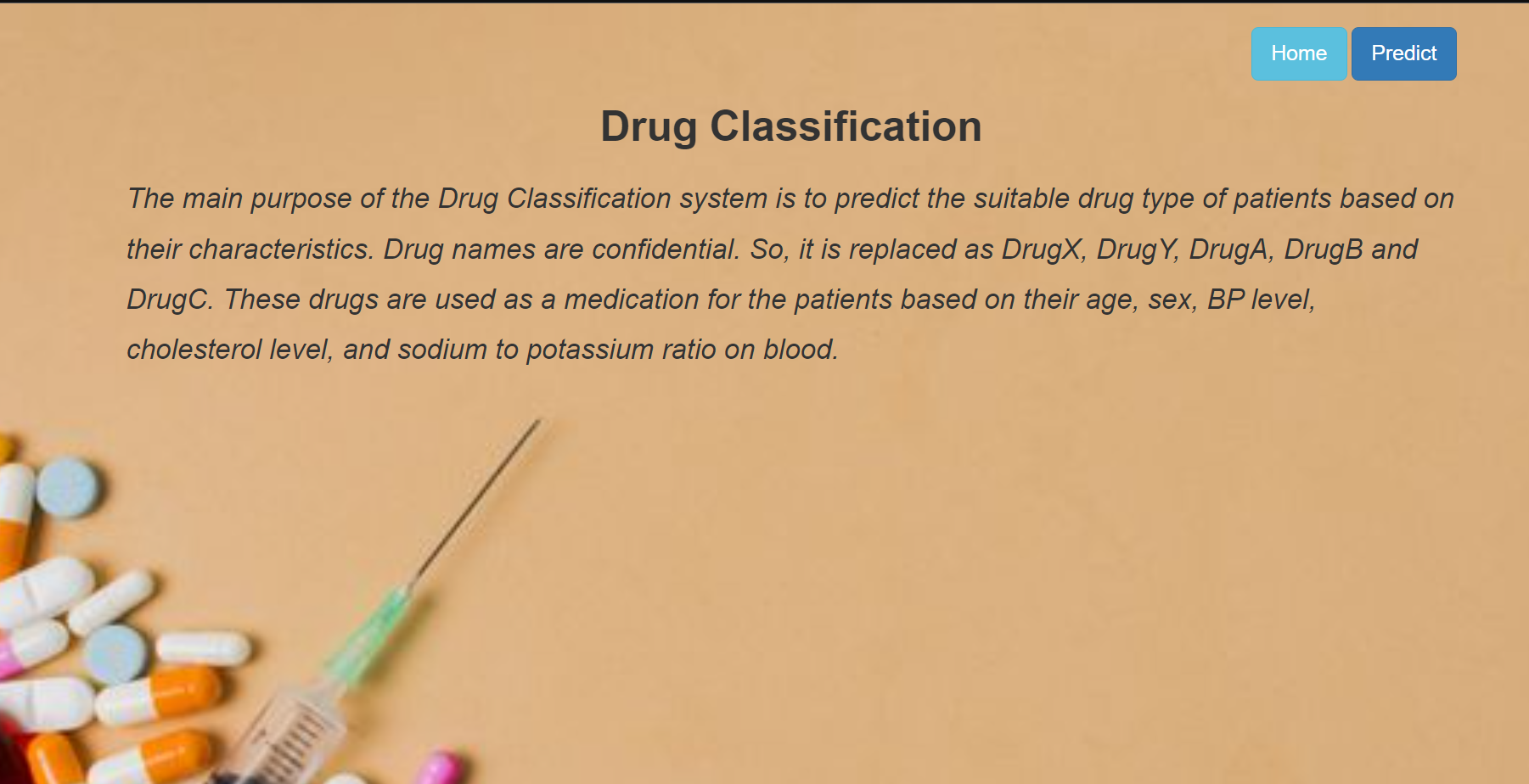
**Activity1: Building Html Pages:**

For this project create three HTML files namely

* home.html
* predict.html
* submit.html

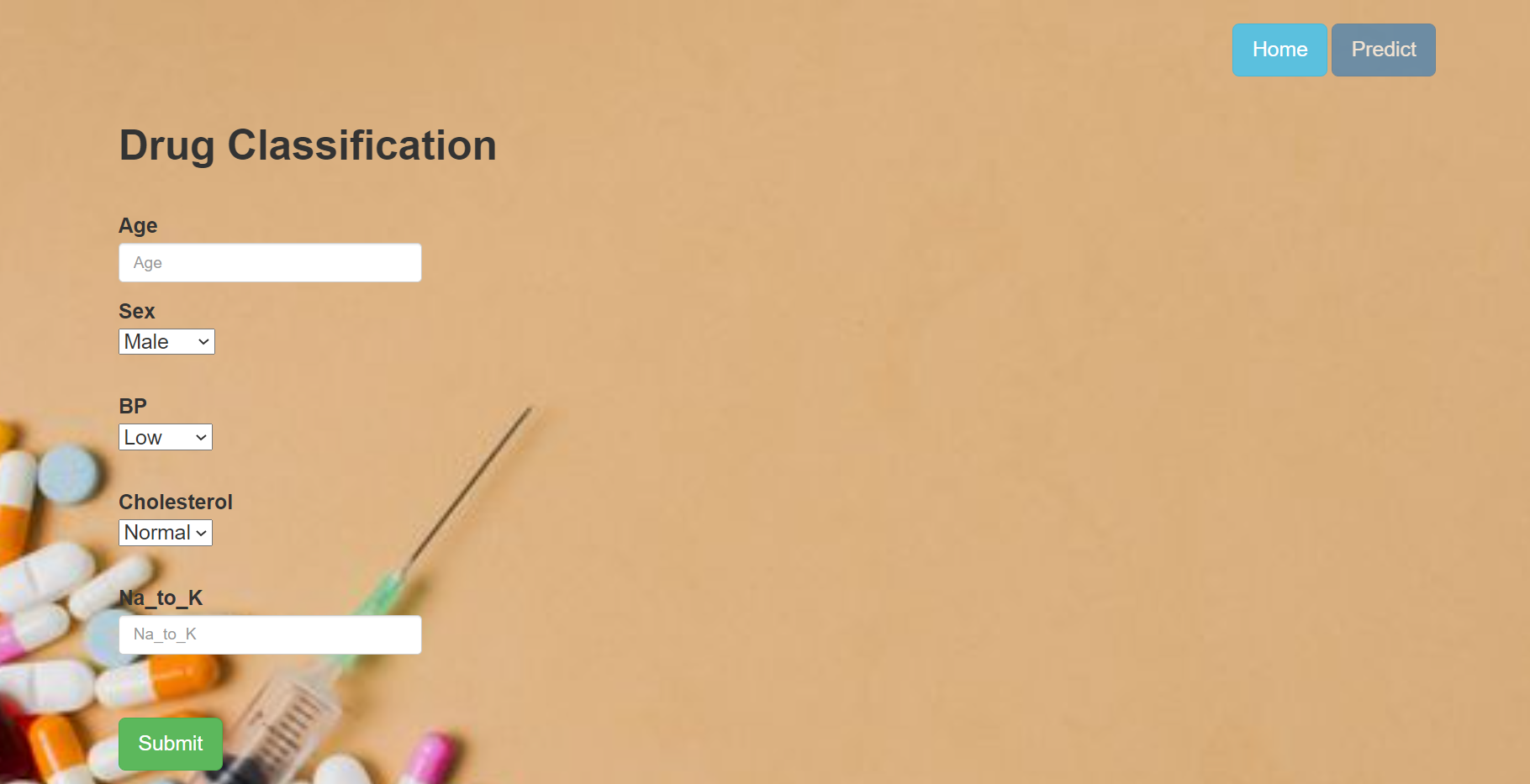
and save them in templates folder.

Let’s see how our home.html page looks like:



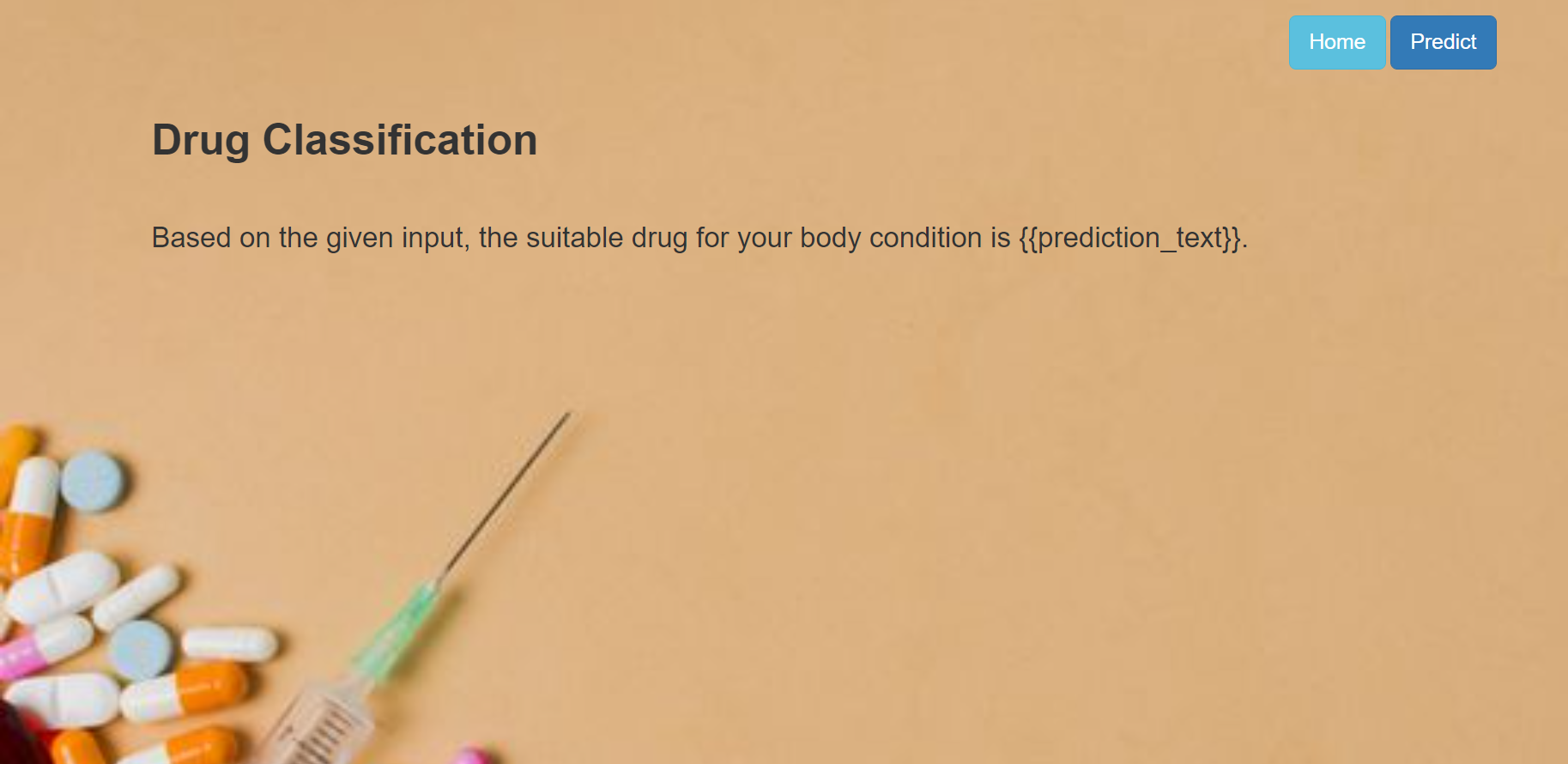
Now when you click on predict button from top right corner you will get redirected to predict.html

Lets look how our predict.html file looks like:



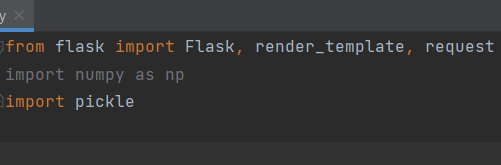
Now when you click on submit button from left bottom corner you will get redirected to submit.html

Lets look how our submit.html file looks like:

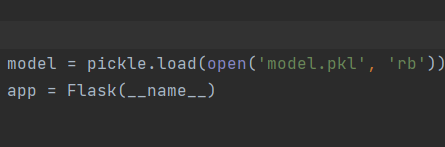


**Activity 2: Build Python code:**

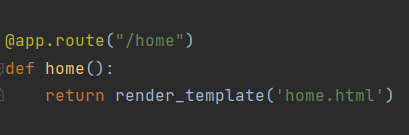
Import the libraries

****

Load the saved model. Importing flask 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.



Render HTML page:



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

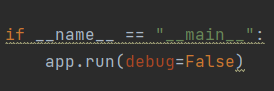
In the above example, ‘/’ URL is bound with home.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:



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:



**Activity 3: 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.

