**Thyroid Disease Classification**

Using Ibm Watson Machine Learning

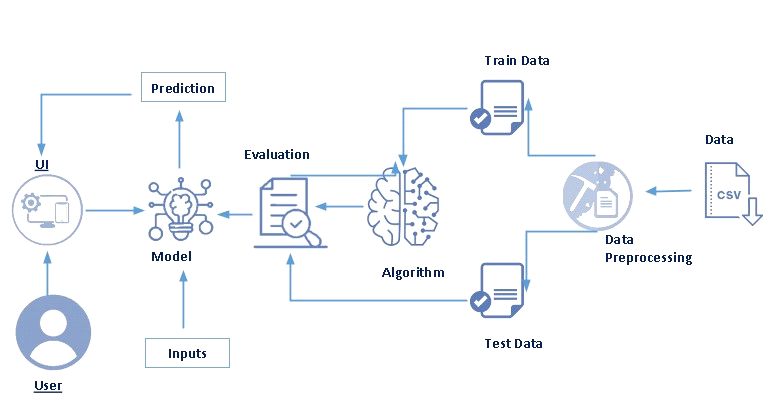
Developed By: V.Sahithi, S.Mounika, M.Manideep, Ch.Savanth, B.Surya

**Smart Bridge – Major Project Report**

**Project Description:**

The Thyroid gland is a vascular gland and one of the most important organs of a human body. This gland secretes two hormones which help in controlling the metabolism of the body. The two types of Thyroid disorders are Hyperthyroidism and Hypothyroidism. When this disorder occurs in the body, they release certain type of hormones into the body which imbalances the body’s metabolism. Thyroid related Blood test is used to detect this disease but it is often blurred and noise will be present. Data cleansing methods were used to make the data primitive enough for the analytics to show the risk of patients getting this disease. Machine Learning plays a very deciding role in the disease prediction. Machine Learning algorithms, SVM - support vector machine, logistic regression, KNN - K-nearest neighbours, Random Forest Classifier, XGB Classifier are used to predict the patient’s risk of getting thyroid disease. Web app is created to get data from users to predict the type of disease.

**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:**

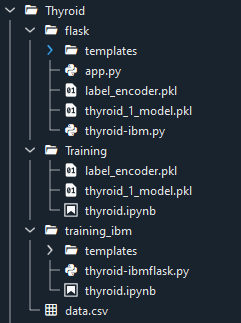
* User interacts with the UI to enter the input.
* Entered input is analyzed by the model which is integrated.
* Once model analyses the input the prediction is showcased on the UI

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

* Data collection
* Collect the dataset or create the dataset
* Visualizing and analysing the data
* Importing the libraries
* Read the Dataset
* Data pre-processing
* Handling missing values
* Descriptive analysis
* Splitting the dataset as x and y
* Handling Categorical Values
* Checking Correlation
* Converting Datatype
* Splitting dataset into training and test set
* Handled Imbalanced Data
* Applying StandardScaler
* Model building
* Import the model building libraries
* Performing Feature Importance
* Selected Output Columns
* Model Building on Selected Columns
* Random Forest Classifier
* XGBoost Classifier
* SVC
* 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.
* thyroid\_1\_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>

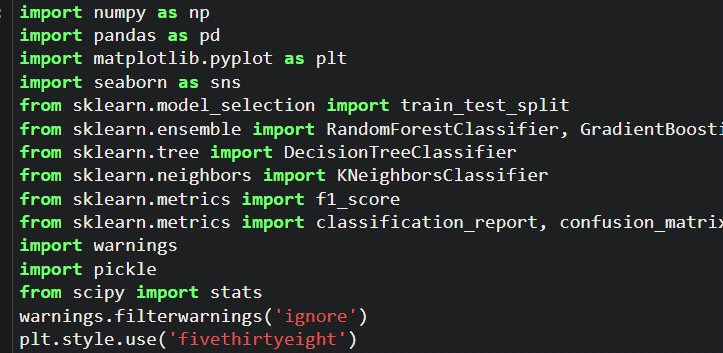
**Milestone 2: Visualizing and analysing the data**

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**

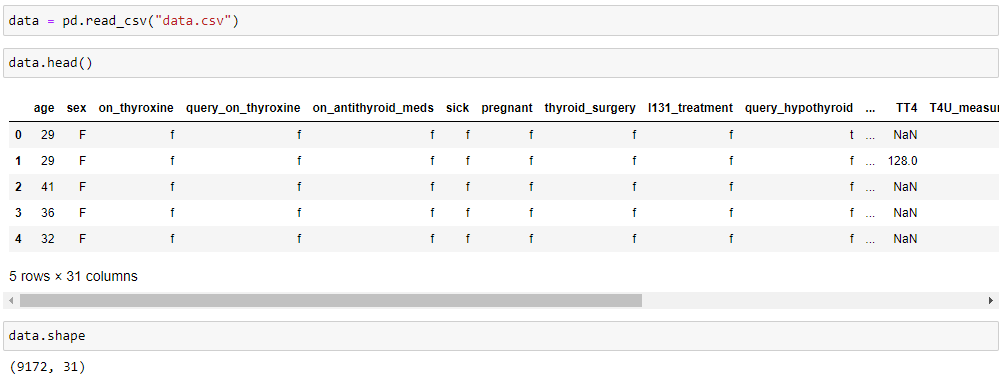
Import the necessary libraries as shown in the image. (optional) Here we have used visualization style as fivethirtyeight.



**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.



**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.

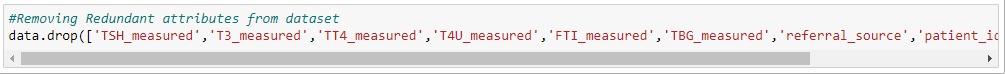
* Handling missing values
* Descriptive analysis
* Splitting the dataset as x and y
* Handling Categorical Values
* Checking Correlation
* Converting Datatype
* Splitting dataset into training and test set
* Handled Imbalanced Data
* Appying StandardScaler

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.

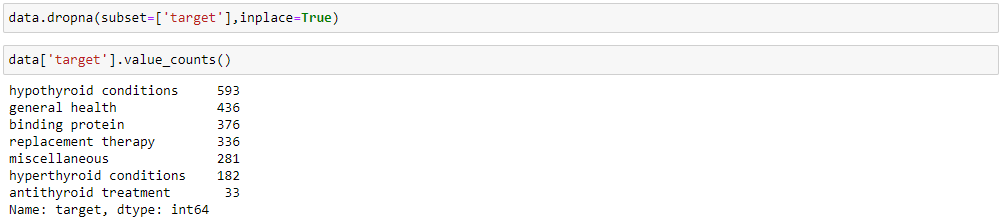
**Activity 1: Checking for null values**

* For checking the null values, data.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.

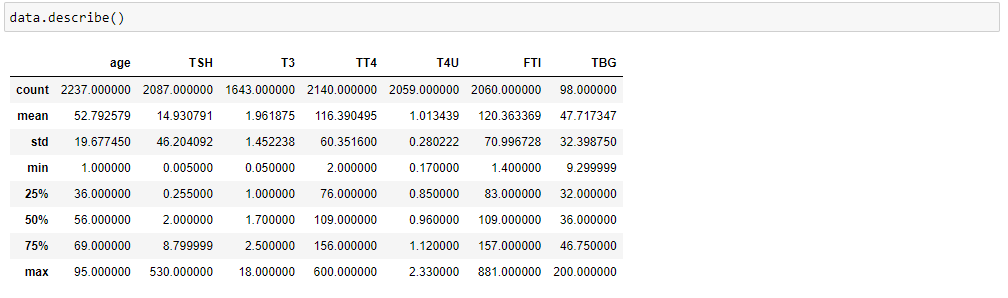
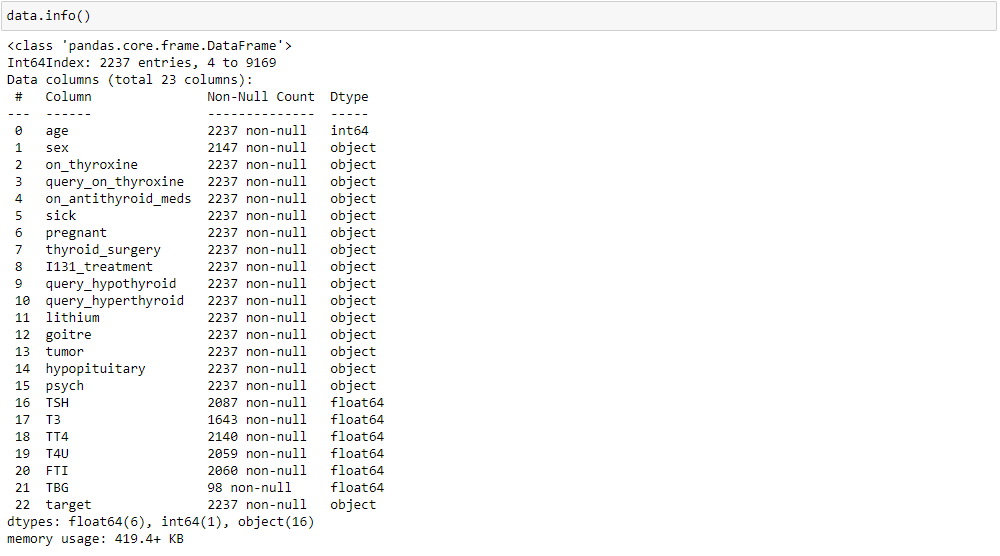
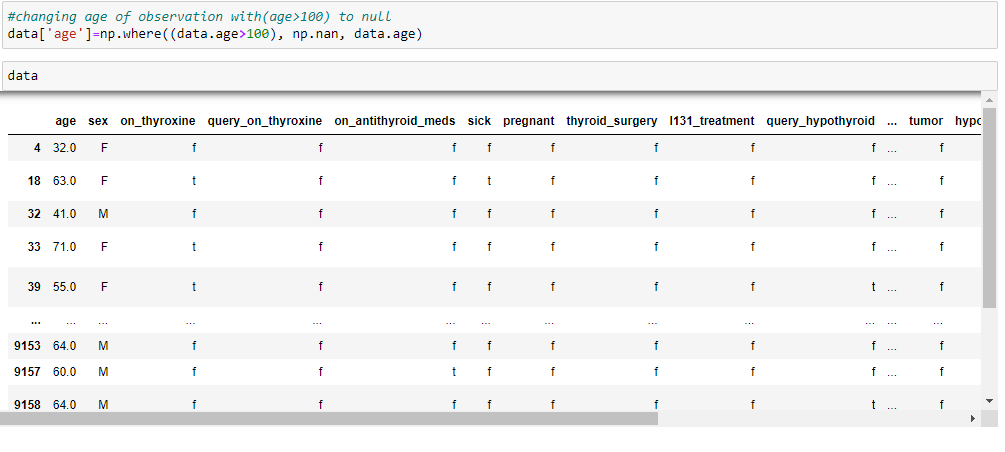
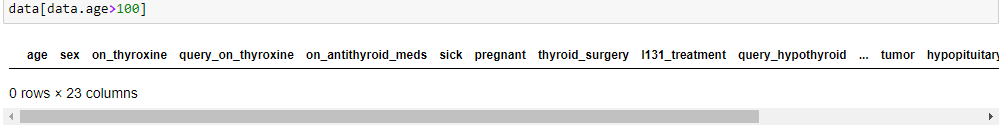


* Removing the Reduntant attributes from dataset.
* Re-mapping the 'target' values to diagnostic Group



* Dropping the 'target' of nan values.

**Activity 2: 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 find mean, std, min, max and percentile values of continuous features.
* Checking info about data by using data\_info()
* Checking the 'age' is there any above 100 and we dropping the age>100.

**Activity 3: Splitting the data x and y**

* Splitting the data x and y 
* Making 'F' on where ever we have the 'nan' values on data.



**Activity 4: Converting the Datatype**

Converting the datatype from object to float. So that we will get output properly. And Checking info about data.



* Here, we have the object values are 'TSH', 'T3', 'TT4', 'T4U', 'FTI', 'TBG' and convert to float values.



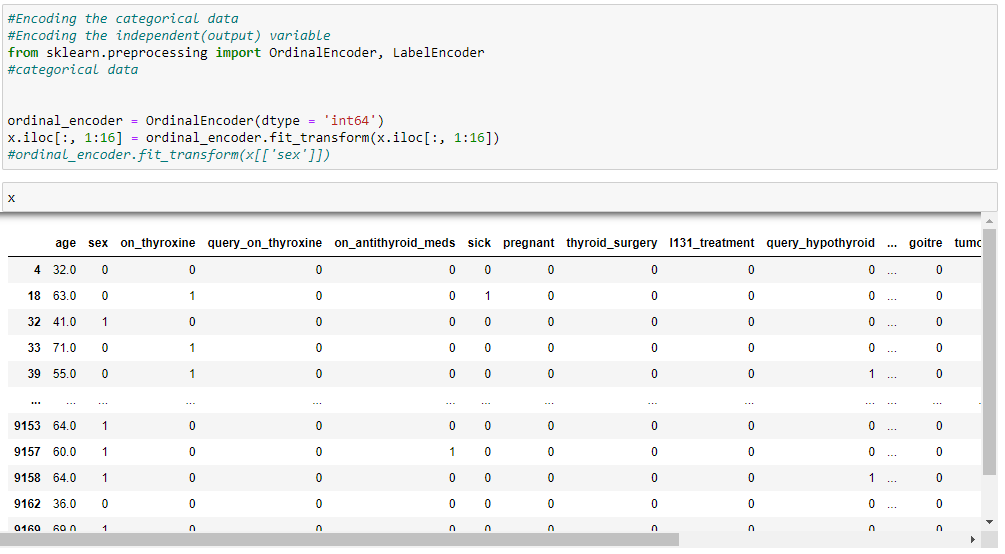
* Then we can check the datatype information about dataset by code of x.info()

**Activity 5: Handling Categorical Values**

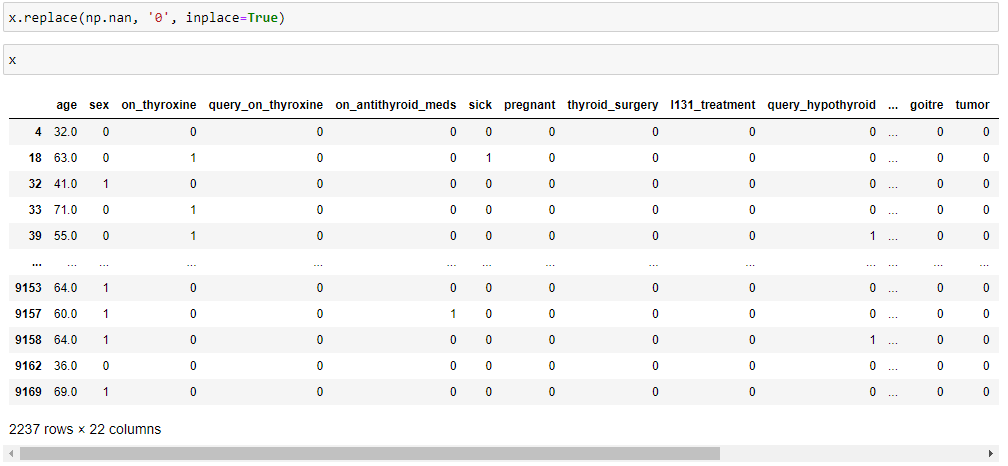
As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using Ordinal Encoding and Label Encoding.

* In our project, categorical features are x and y values.
* Here, applying Ordinal Encoding on x values.

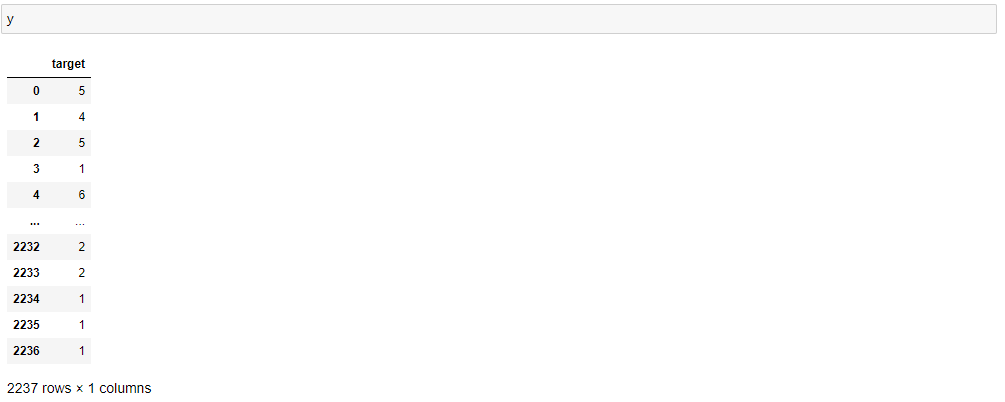


* Replacing the nan values with zero (0) values.



* Now, applying Label Encoding on y(Independent variable) value.

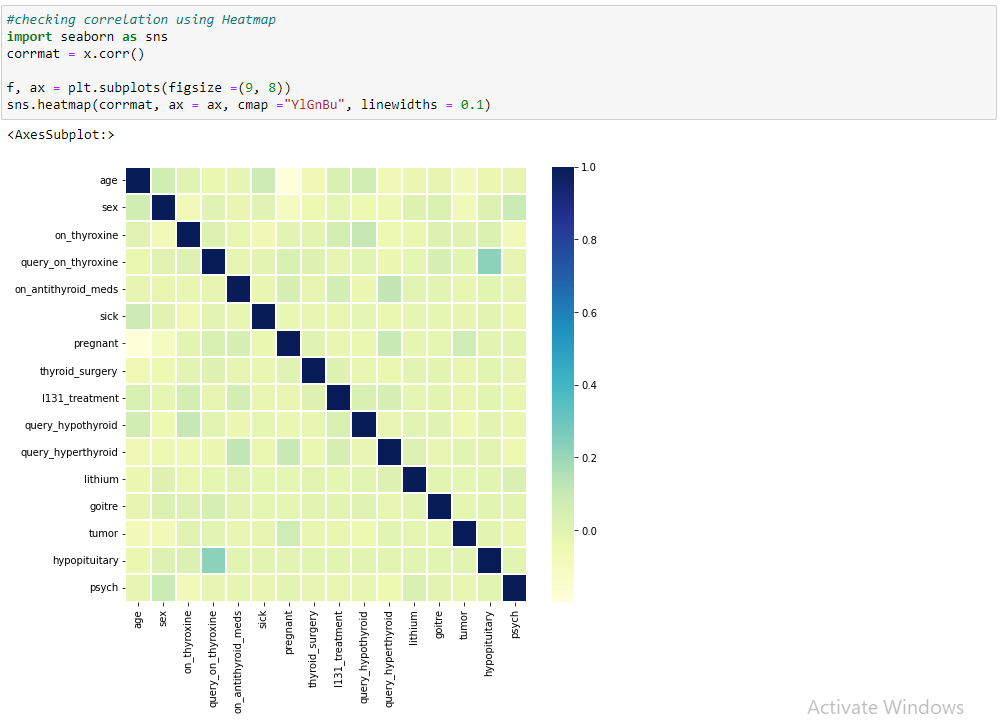




**Activity 6: Checking Correlation.**

Here, I 'm finding the correlation using HeatMap. It visualize the data in 2-D colored maps making use of color variations. It describes the relationship variables in form of colors instead of numbers it will be plotted on both axes.

Here, there is no any correlation between columns.

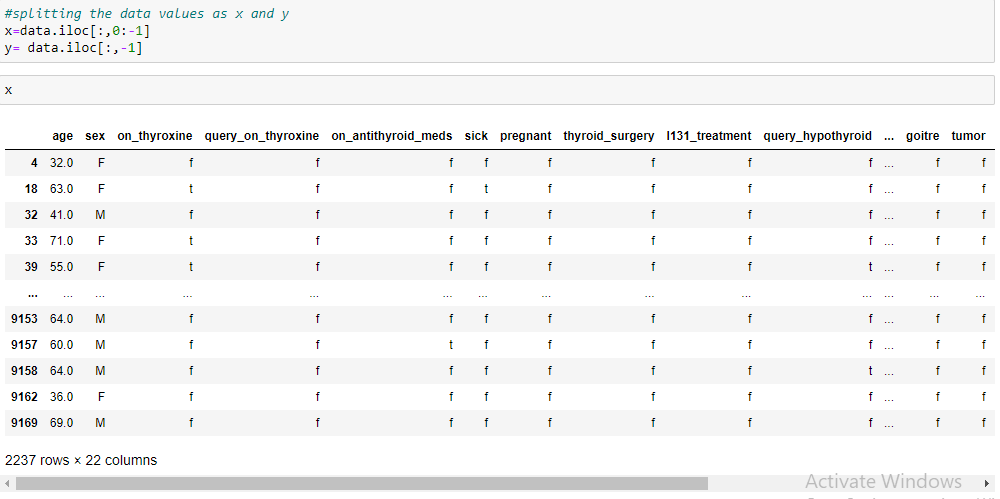


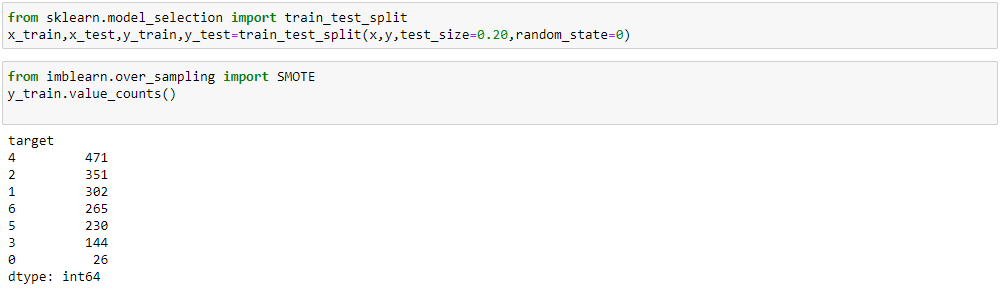
**Activity 7: Splitting data into train and test**

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

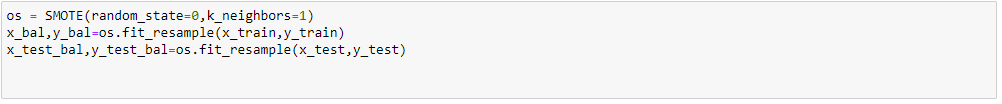
Changes: first split the dataset into x and y and then split the data set

Here x and y variables are created. On x variable, data 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.



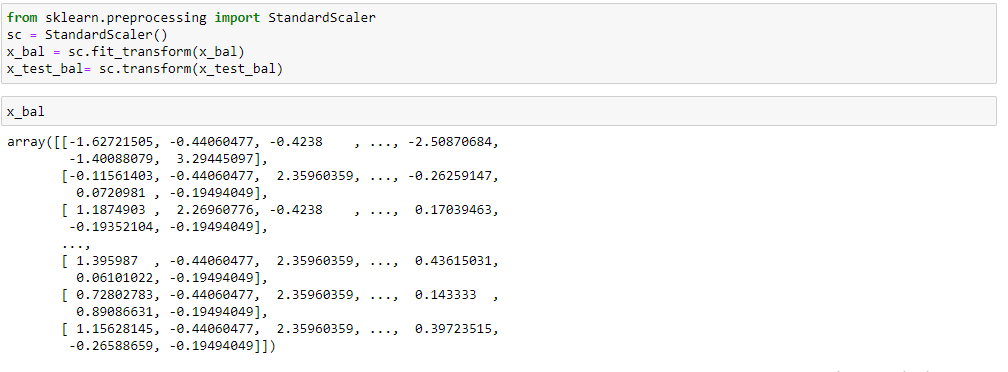


**Activity 8: Handling Imbalanced Data**

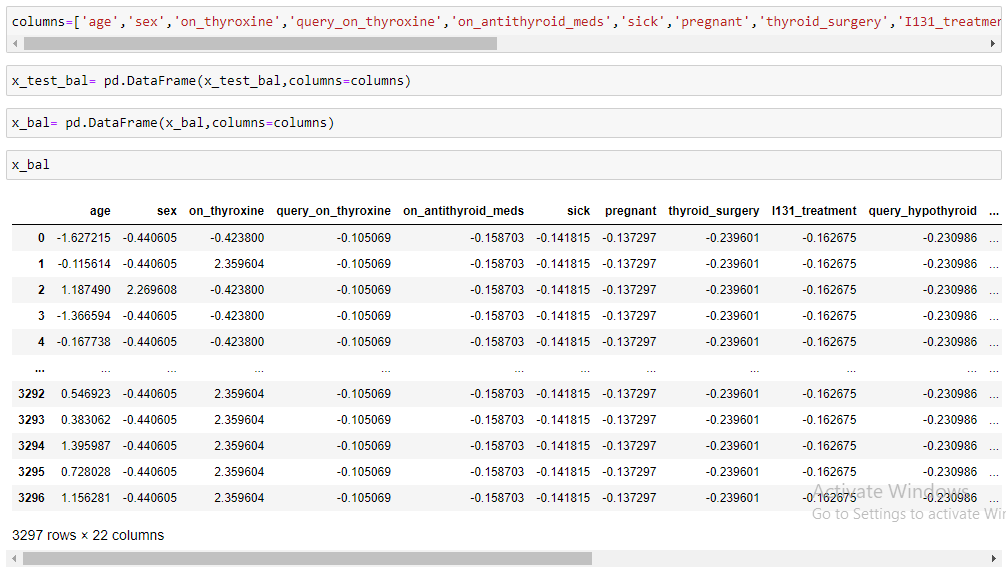


**Activity 9: Applying StandardScaler**

* Scaling the features makes the flow of gradient descent smooth and helps algorithms quickly reach the minima of the cost function.
* Without scaling features, the algorithm maybe biased toward the feature which has values higher in magnitude. it bring every feature in the same range and the model uses every feature wisely.



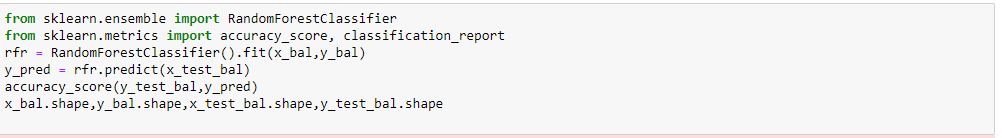
* Here, we have the data in array format and we are making it dataframe(table format).



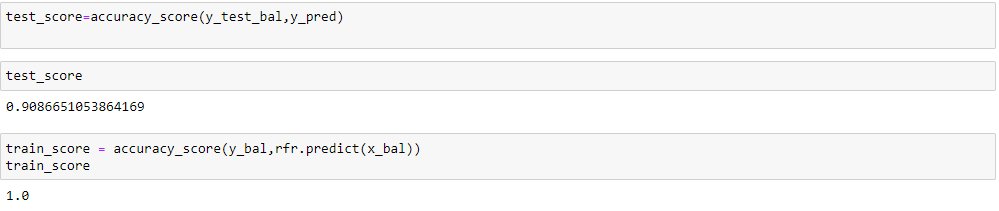
**Milestone 4: Model Building**

**Activity 1: Random forest model**

* A function named Random forest model is created and train and test data are passed as the parameters. Inside the function, RandomForest 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, accuracy\_score, classification\_report(precision, recall, f1 score, support). It gives the best results.

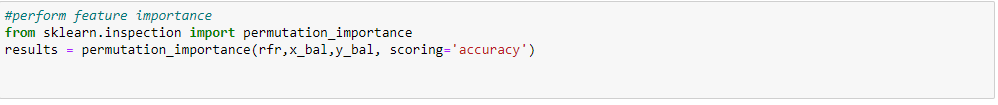


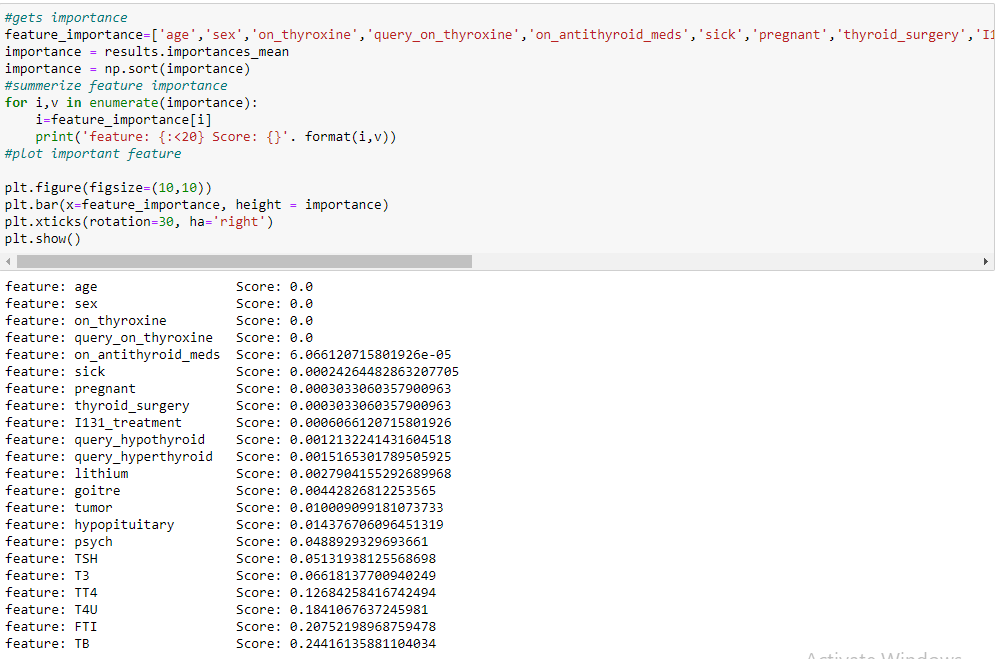




**Milestone 5: Performing Feature Importance**

* The idea behind permutation feature importance is simple. The feature importance is calculated by noticing the increase or decrease in error when we permute the values of a feature.
* If permuting the values causes a huge change in the error, it means the feature is important for our model.

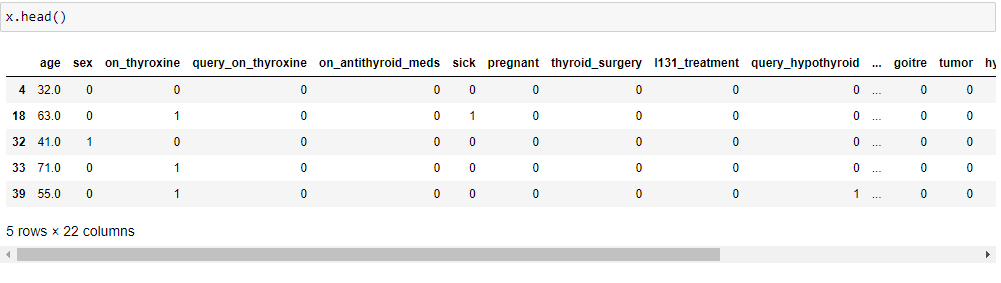




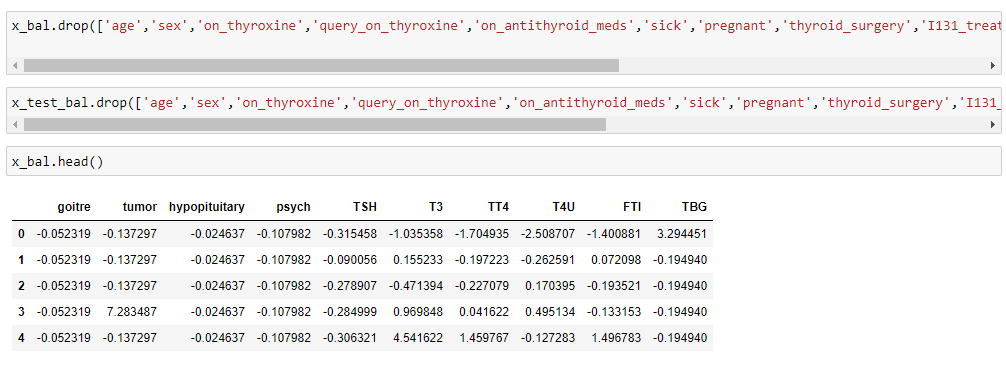


**Milestone 6: Selecting Output Columns**

* Before we have this many columns



* After Performing Feature Importance by using 'Permutation Importance' we are dropping some columns which are not important for 'target'.

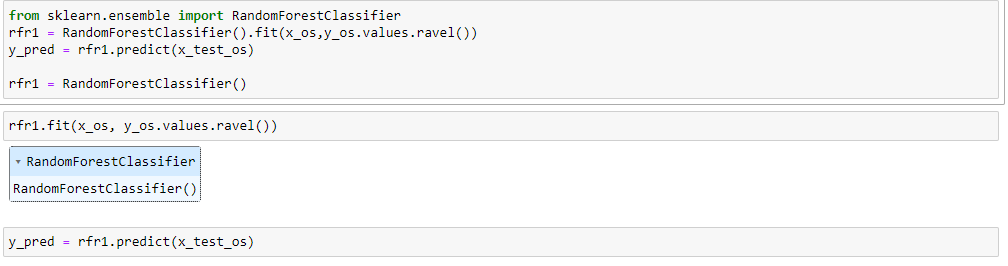


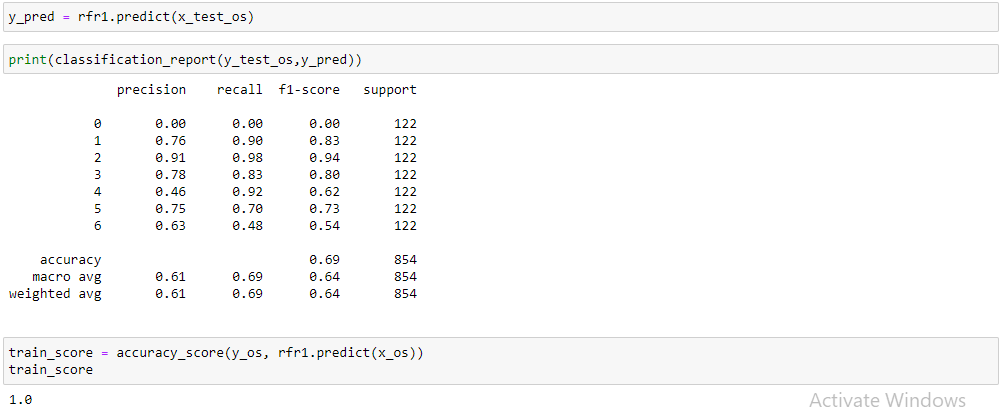
**Milestone 7: Model Building On Selected Columns**

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: Random Forest Classifier Model**

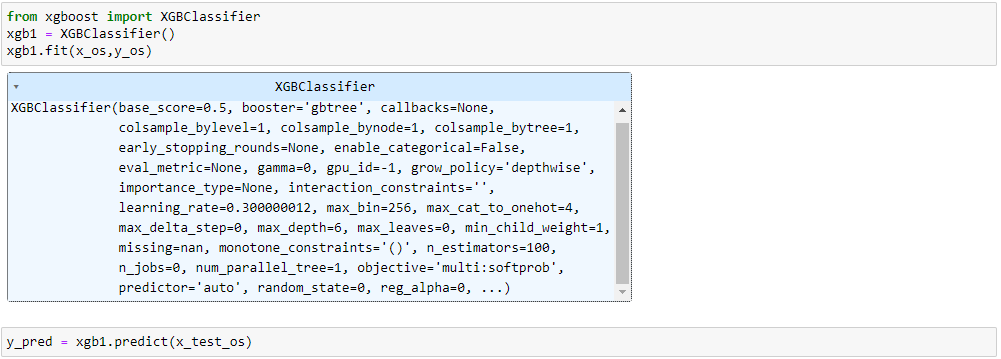
A function named Random Forest Classifier Model is created and train and test data are passed as the parameters. Inside the function, Random Forest Classifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, accuracy\_score and classification report is done.

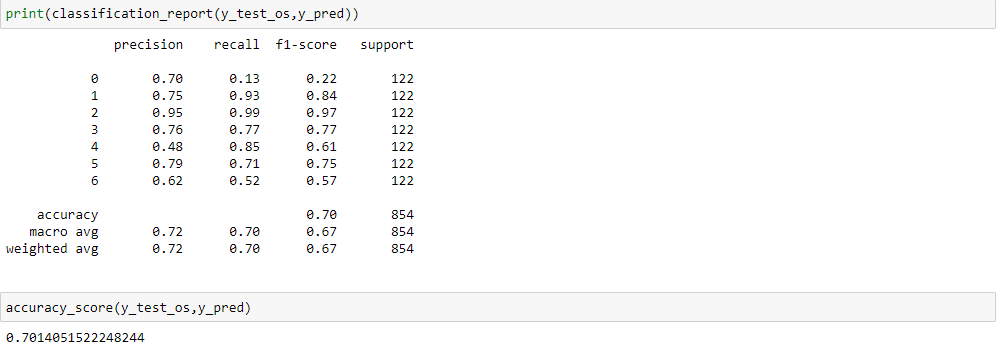




**Activity 2: XGBClassifier model**

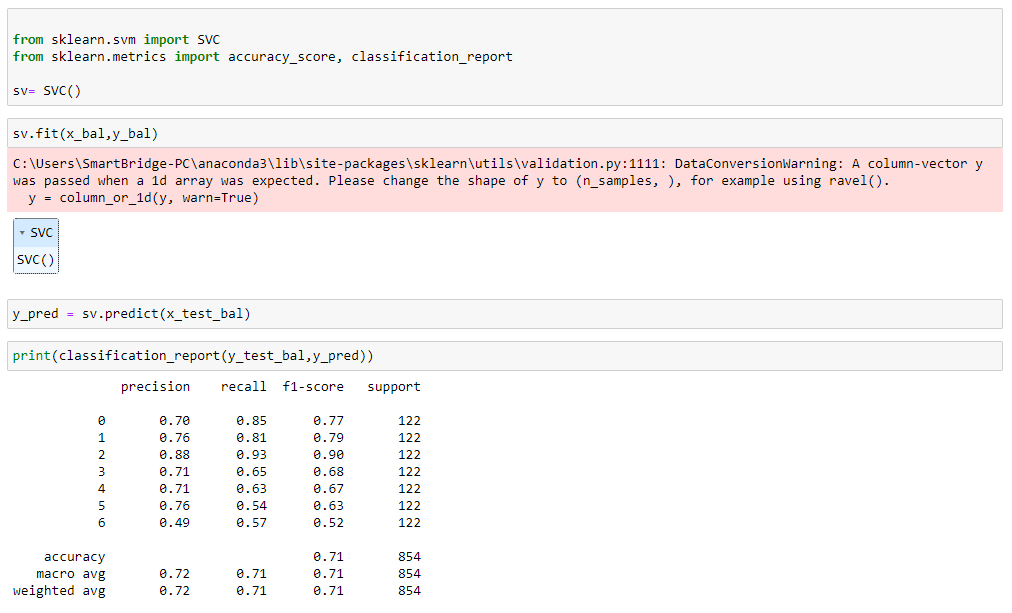
A function named XGBClassifier model is created and train and test data are passed as the parameters. Inside the function, XGBClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For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





**Activity 3: SVC model**

A function named SVC model is created and train and test data are passed as the parameters. Inside the function, SVC 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, accuracy score and classification report is done.

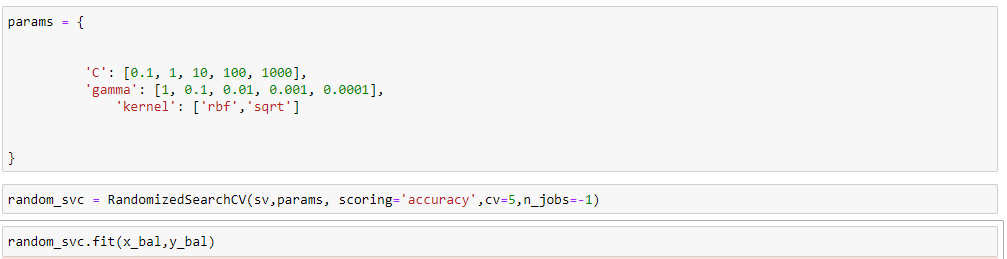


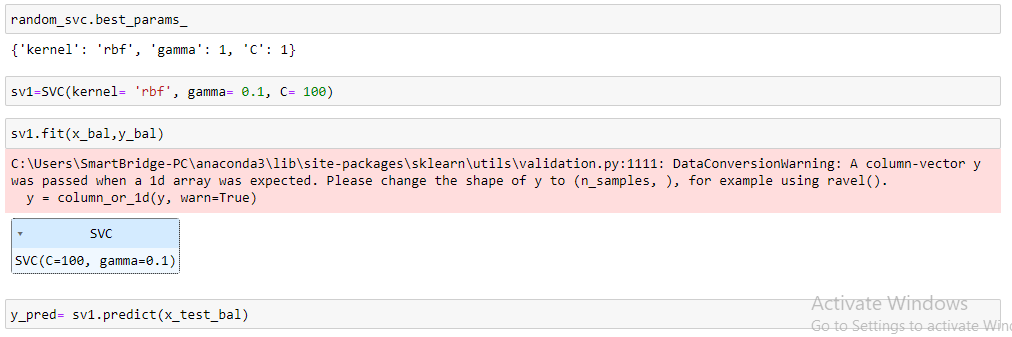


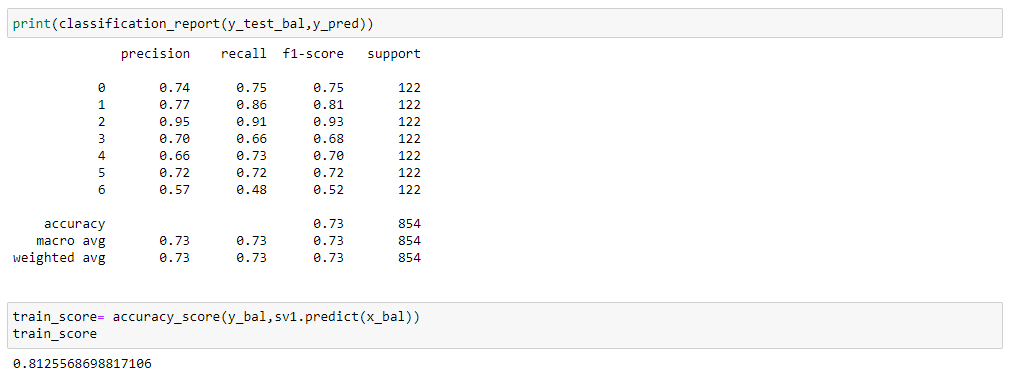
**Activity 4: Evaluating performance of the model Using RandomSearch CV and saving the model**

From sklearn, accuracy is used to evaluate the score of the model. On the parameters, we have given sv1 (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model by pickle.dump().

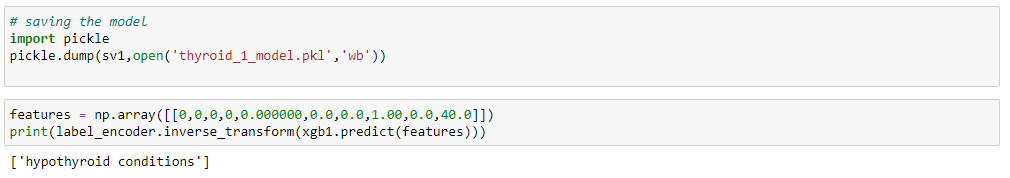
Note: To understand cross validation, refer this link. <https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85>.



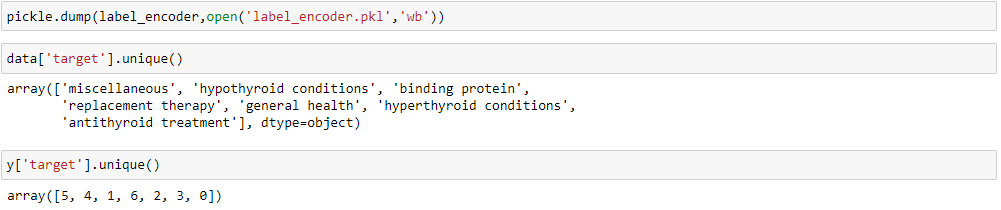




Saving the model as thyroid1\_model.pkl



Here, we are saving label\_encoding also as label\_encoder.pkl



**Milestone 8: 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

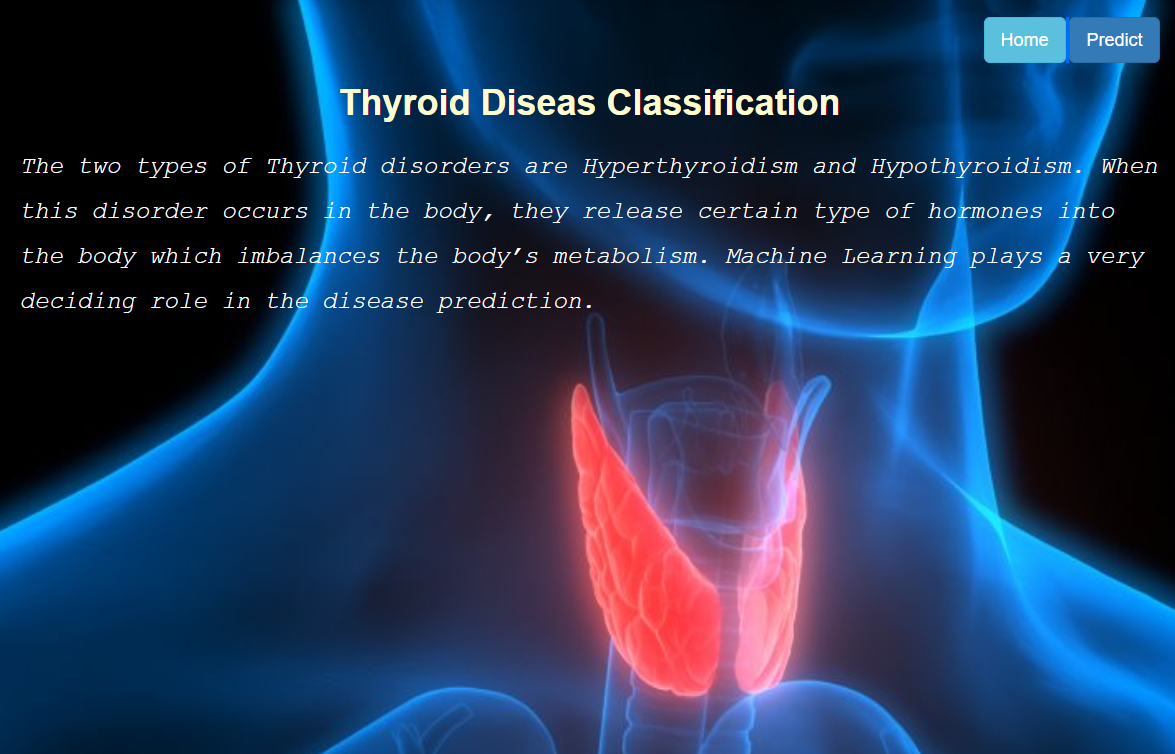
**Activity1: Building Html Pages:**

For this project create three HTML files namely

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

and save them in templates folder.

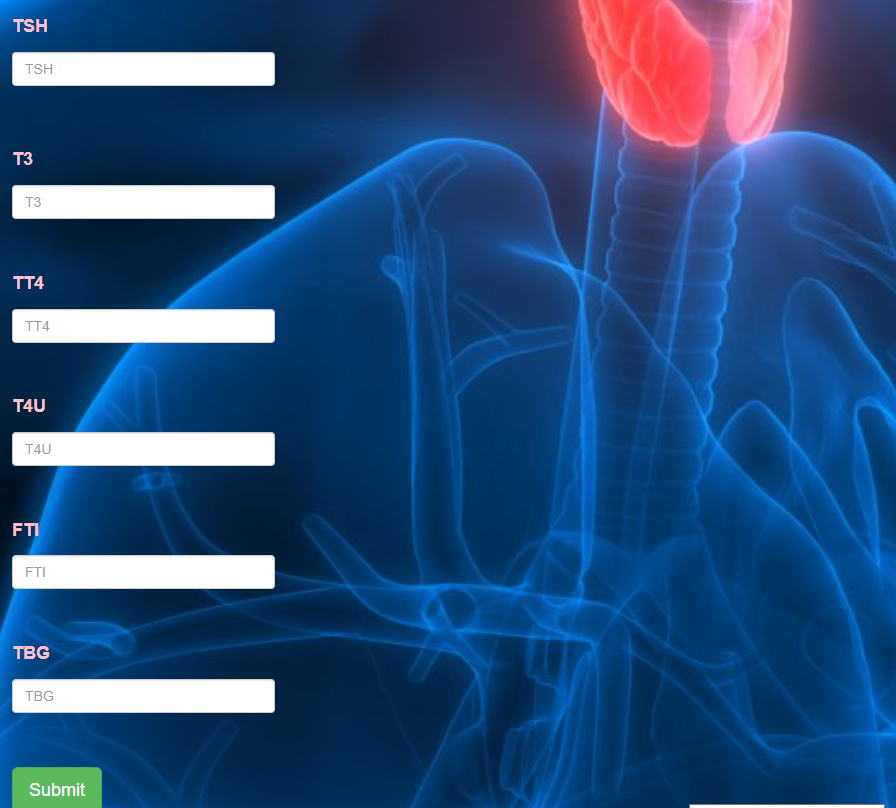
Let’s see how our home.html page looks like:



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

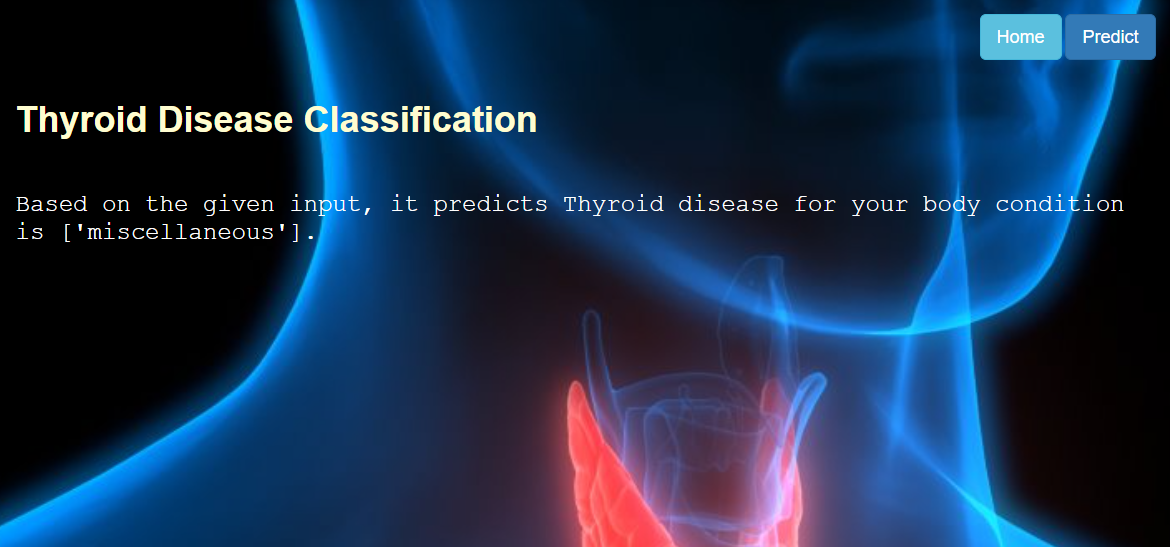
Lets look how our predict.html file looks like:





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

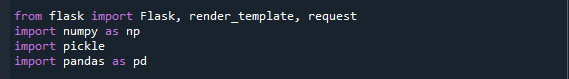
Lets look how our submit.html file looks like:



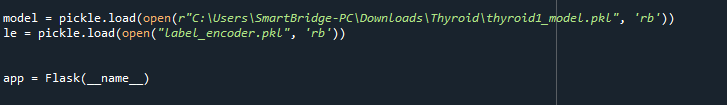
it's is ['miscellaneous'].

**Activity 2: Build Python code:**

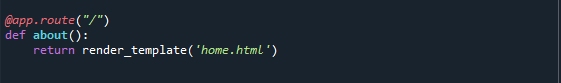
Import the libraries



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.



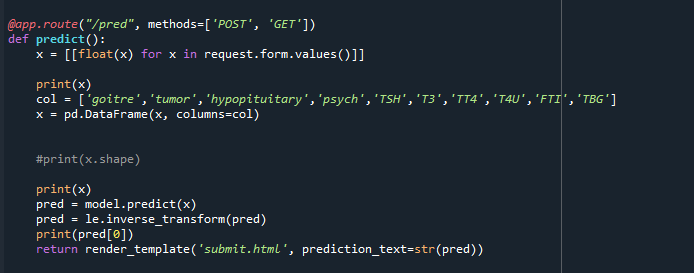
Render HTML page:



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.

