**PREDICTION OF ELECTRICAL OUTPUT POWER OF**

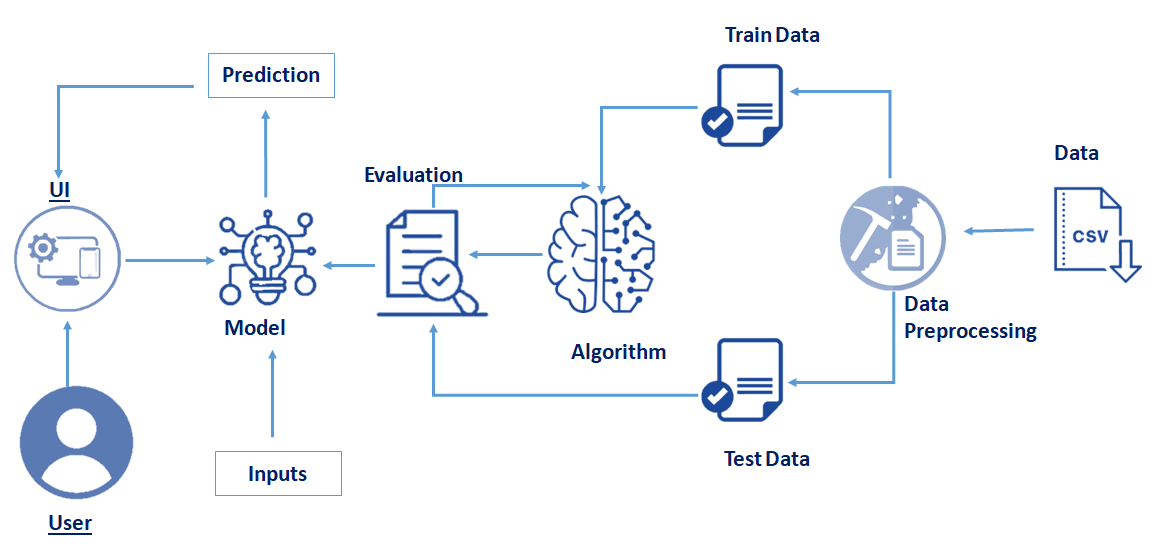
**COMBINED CYCLE POWER PLANT**

**Introduction to project:**

The Combined Cycle Power Plant or combined cycle gas turbine, a gas turbine generator generates electricity and waste heat is used to make steam to generate additional electricity via a steam turbine. The gas turbine is one of the most efficient one for the conversion of gas fuels to mechanical power or electricity. The use of distillate liquid fuels, usually diesel, is also common as alternate fuels. More recently, as simple cycle efficiencies have improved and as natural gas prices have fallen, gas turbines have been more widely adopted for base load power generation, especially in combined cycle mode, where waste heat is recovered in waste heat boilers, and the steam used to produce additional electricity.

The basic principle of the Combined Cycle is simple: burning gas in a gas turbine (GT) produces not only power, which can be converted to electric power by a coupled generator, but also fairly hot exhaust gases. Routing these gases through a water-cooled heat exchanger produces steam, which can be turned into electric power with a coupled steam turbine and generator. This type of power plant is being installed in increasing numbers round the world where there is access to substantial quantities of natural gas.

**Technical Architecture:**

****

**Pre requisites:**

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

* **Anaconda navigator:**
  + Refer to the link below to download anaconda navigator
  + **Link : https://www.youtube.com/watch?v=5mDYijMfSzs**
* **Python packages:**
  + open anaconda prompt as administrator
  + Type “pip install tensorflow” (make sure you are working on python 64 bit)
  + Type “pip install flask”.

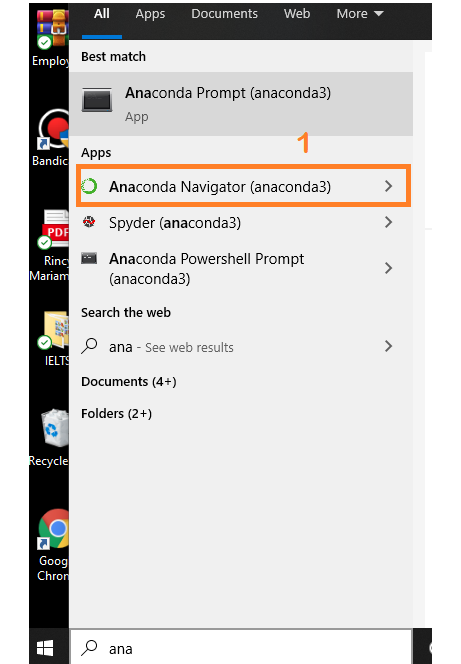
or

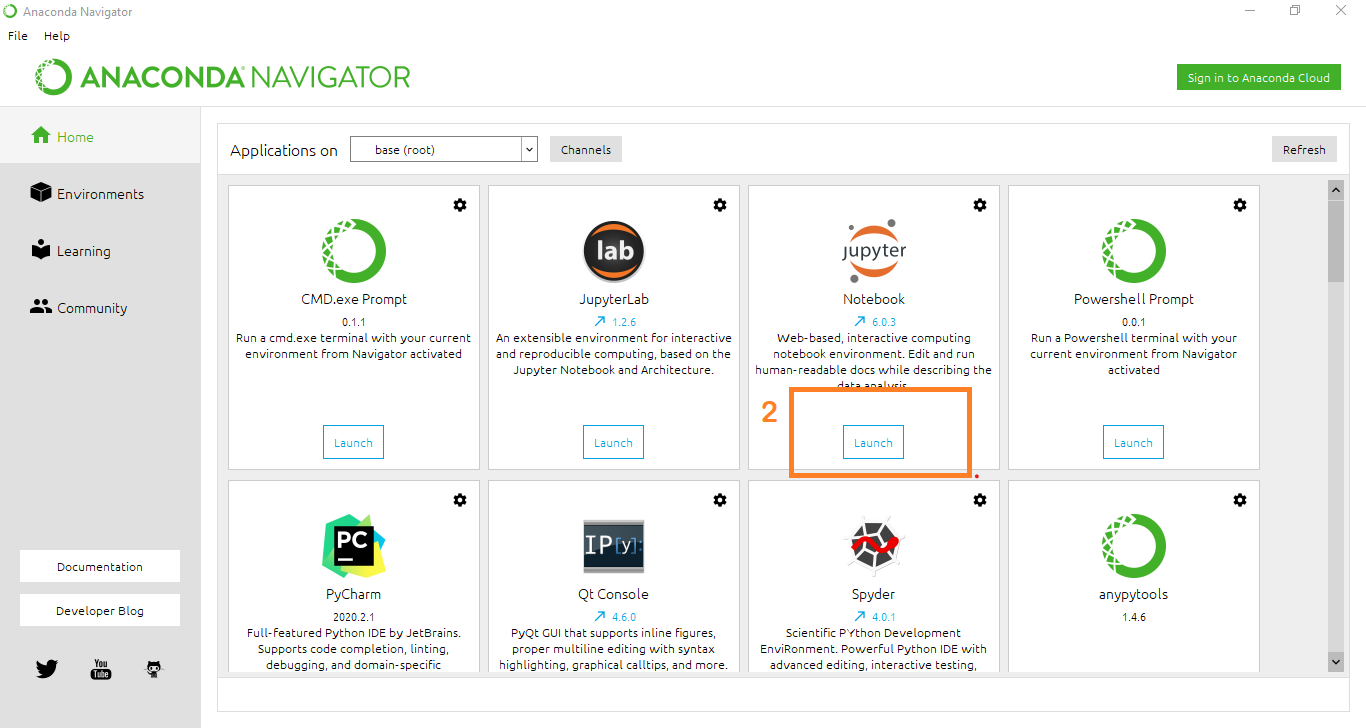
* + Open anaconda prompt as administrator
  + Type “pip install -r requirements.txt” (make sure you are working on python 64 bit)

The above requirements.txt files allow you to install all the required files in the anaconda environment for your project.

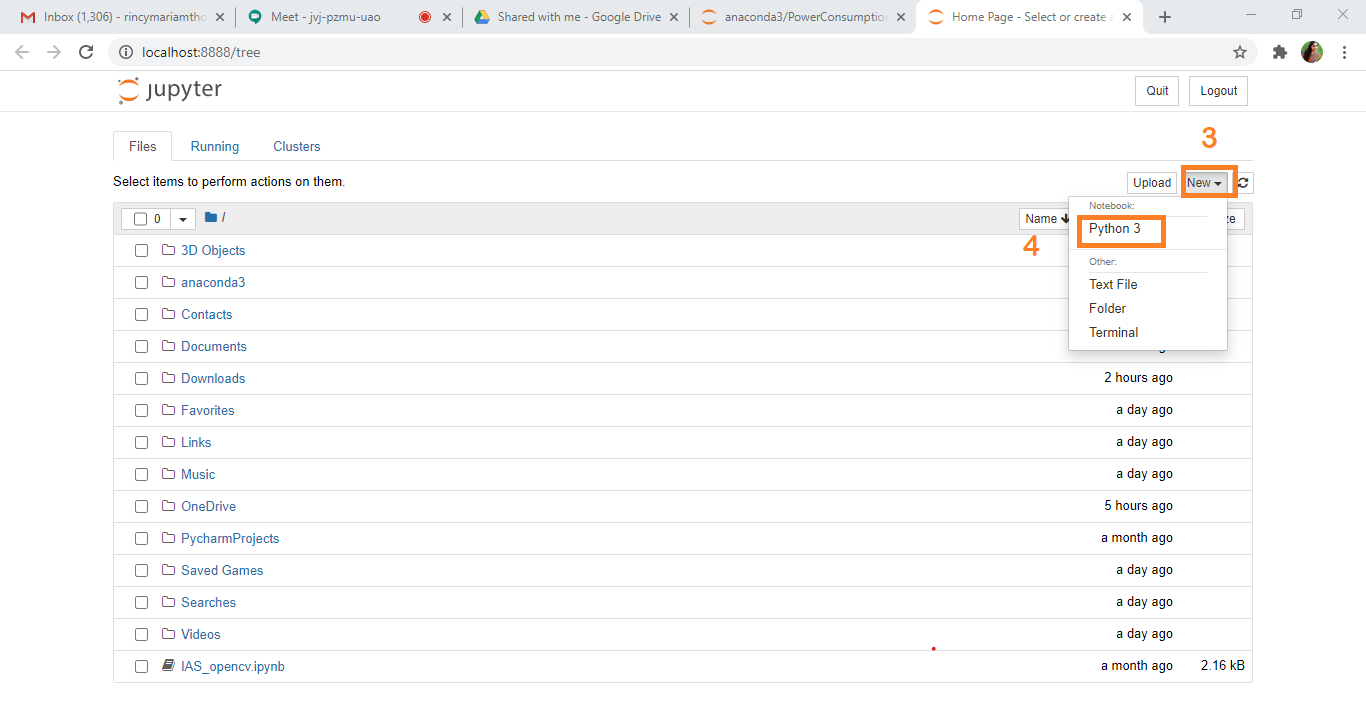
the above steps allow you to install keras and tensorflow in the anaconda environment

* **Machine Learning Concepts** 
  + **Linear Regression:** <https://towardsdatascience.com/introduction-to-linear-regression-and-polynomial-regression-f8adc96f31cb>
  + **Decision Tree Regression:** <https://towardsdatascience.com/decision-tree-in-machine-learning-e380942a4c96>
  + **Random Forest Regression:** <https://medium.com/swlh/random-forest-and-its-implementation-71824ced454f>
  + **Flask Basics** : <https://www.youtube.com/watch?v=lj4I_CvBnt0>
* **Launch Jupyter**
  + Search for Anaconda Navigator and open Launch Jupyter notebook.





* Then you will be able to see that the jupyter notebook runs on local host:8888.
* To Create a new file Go to New 🡪Python3. The file in jupyter notebook is saved with .ipynb extension.



**Project Objectives:**

By the end of this project you will:

* Know fundamental concepts and techniques of time Machine learning algorithms.
* Know fundamental concepts of python
* Know how to install necessary packages and setting up the environment.
* Know how to build a web application using Flask framework.

**Project Flow:**

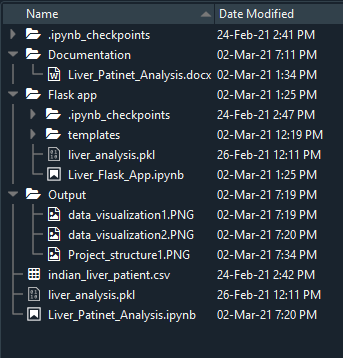
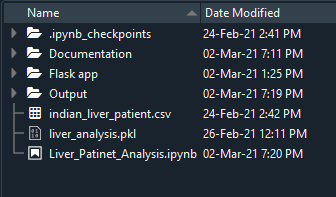
* User interacts with the UI (User Interface) to predict the values based on previous data
* The list of predicted output is showcased on the UI.
* Entered data is analyzed by the model which is integrated

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

* Data Collection.
  + Collect the dataset or Create the dataset
* Data Preprocessing.
  + Import the Libraries.
  + Importing the dataset.
  + Analyse the data
  + Taking care of Missing Data
  + Data Visualization
  + Splitting Data into Train and Test.
* Model Building
  + Import the model building Libraries
  + Initializing the model
  + Adding Machine Learning Model
  + Training the model
  + Model Evaluation
  + Save the Model
* Application Building
  + Create an HTML file
  + Build Python Code

**Project Structure:**

Create a 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 CCPP\_Flask\_App.ipynb for serverside scripting
* CCPP\_Flask\_App.ipynb - contains the actual python code that will import the app and start the development server.
* CCPP.ipynb - This is where you define models for your application.
* CCPP.pkl - This is our model weights file
* templates - This is where you store your html templates i.e. index.html, home.html, predict.html

**Milestone 1: Data Collection**

ML depends heavily on data, without data, a machine can't learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training data set. It is the actual data set used to train the model for performing various actions.

**Activity 1: Download dataset /create dataset**:

You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository etc.

The dataset used for this project was obtained from Kaggle. Please refer to the link given below to download the data set and to know about the dataset

* + **Link**:<https://drive.google.com/file/d/1pVJGYpR1lYtRw1WBVLzuTzh_WUkDYsTk/view>
  + Combined cycle power plant contains several columns.

**Milestone 2: Data Preprocessing**

Data Pre-processing includes the following main tasks

* + Import the Libraries.
  + Importing the dataset.
  + Analyse the data
  + Taking care of Missing Data
  + Data Visualization
  + Splitting Data into Train and Test.

**Activity 1: Import the Libraries**

The first step is usually importing the libraries that will be needed in the program.

The required libraries to be imported to Python script are:

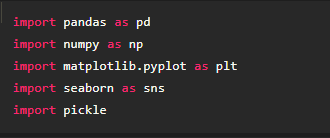
**Numpy:** It is an open-source numerical Python library. It contains a multi-dimensional array and matrix data structures. It can be used to perform mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.

**Pandas**: It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

**Matplotlib**: Visualisation with python. It is a comprehensive library for creating static,animated, and interactive visualizations in Python.

**Seaborn:** Seaborn is a library for making statistical graphics in Python. Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

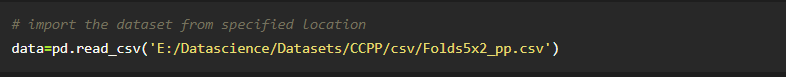
**Pickle:** The pickle module implements serialization protocol, which provides an ability to save and later load Python objects using special binary format**.**



Note: It’s conventional to refer to alias. When you add the alias name at the end of your import statement, your Jupyter Notebook understands that from this point on every time you type alias name, you are actually referring to the particular library.

**Activity 2**: **Importing the dataset**

* You might have your data in .csv files, .excel files
* Let’s load the excel data file into pandas using the **read\_excel() function.** We will need to locate the directory of the excel file at first (it’s more efficient to keep the dataset in the same directory as your program).



* If your dataset is in some other location, Then

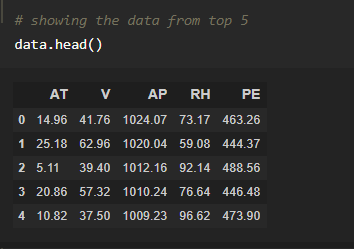
Data=pd.read\_excel(r”File\_location”)

**Note:**r stands for "raw" and will cause backslashes in the string to be interpreted as actual backslashes rather than special characters.

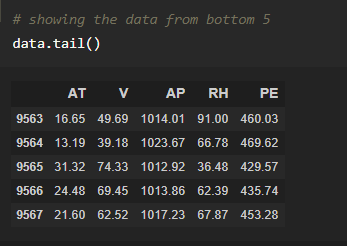
* If the dataset in same directory of your program, you can directly read it, without giving raw as r.

**Activity 3** : **Analyse the data**

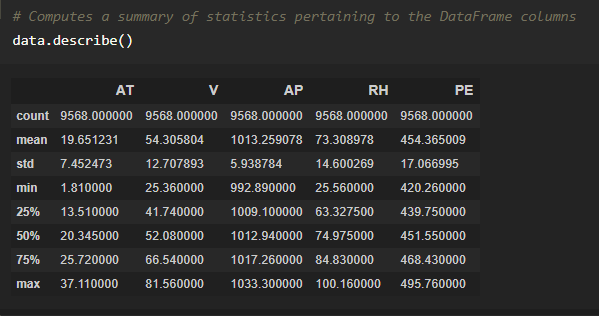
* head() method is used to return top n (5 by default) rows of a DataFrame or series.



* tail() method is used to return bottom n (5 by default) rows of a DataFrame or series.

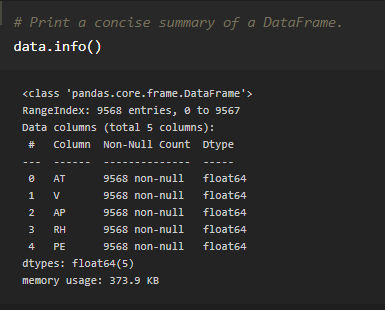


* describe() method computes a summary of statistics like count, mean, standard deviation, min, max, and quartile values.



From the data we info that there are 9568 records

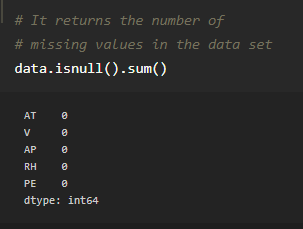
* info() gives information about the data



**Activity 4**: **Taking care of Missing Data**

1. After loading the dataset, it is important to check the complete information of such as null values in a column or a row

2.Check whether any null values are there or not. if it is present then the following can be done,



As we can see he there is no null values.

**Activity 5** : **Data Visualization**

**Link**: <https://towardsdatascience.com/data-visualization-for-machine-learning-and-data-science-a45178970be7>

* Data visualization is where a given data set is presented in a graphical format. It helps the detection of patterns, trends and correlations that might go undetected in text-based data.
* Understanding your data and the relationship present within it is just as important as any algorithm used to train your machine learning model. In fact, even the most sophisticated machine learning models will perform poorly on data that wasn’t visualized and understood properly.
* To visualize the dataset we need libraries called Matplotlib and Seaborn.
* The Matplotlib library is a Python 2D plotting library which allows you to generate plots, scatter plots, histograms, bar charts etc.

Let’s visualize our data using Matplotlib and searborn library.

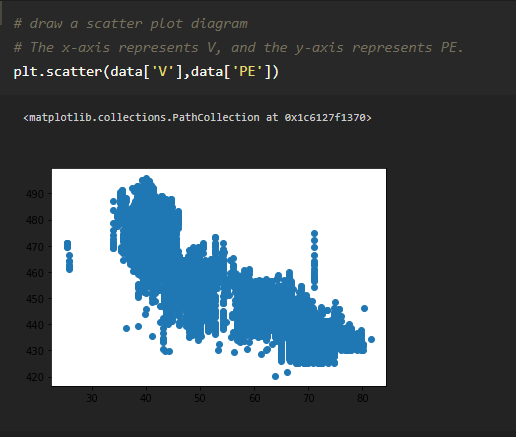
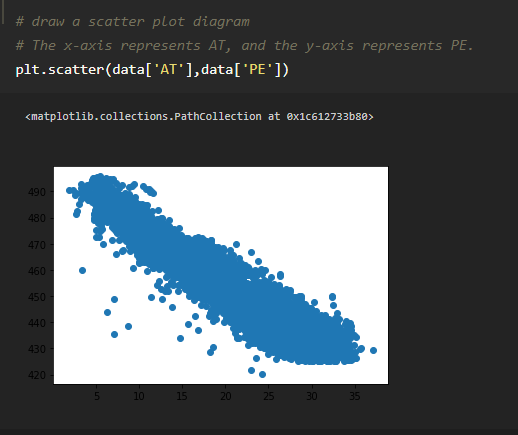
Before diving into the code, let's look at some of the basic properties we will be using when plotting.

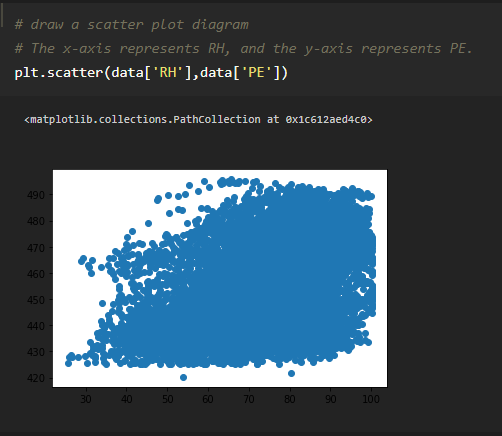
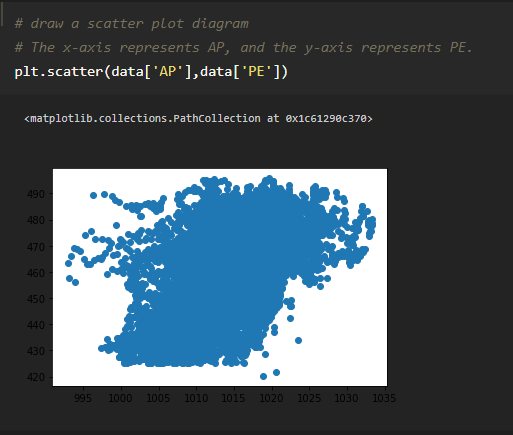
xlabel: Set the label for the x-axis.

ylabel: Set the label for the y-axis.

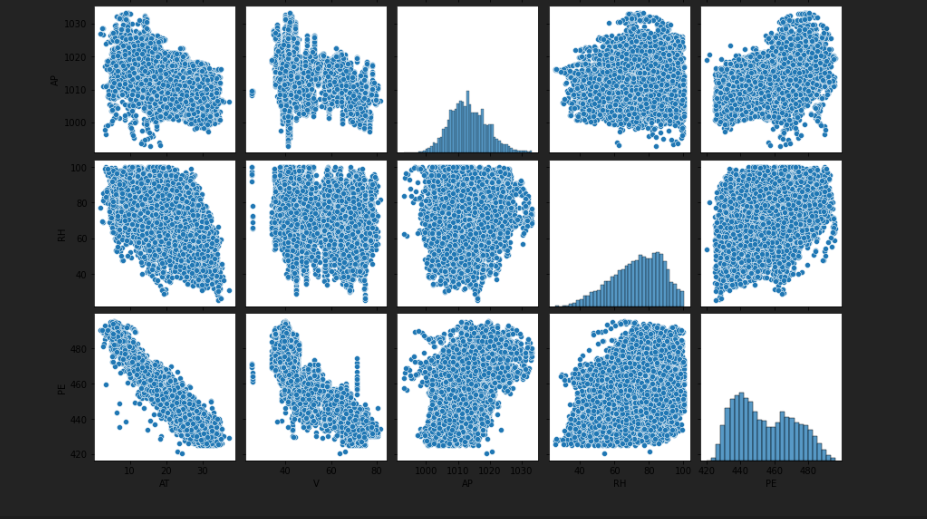
title: Set a title for the axes.

Legend: Place a legend on the axes.

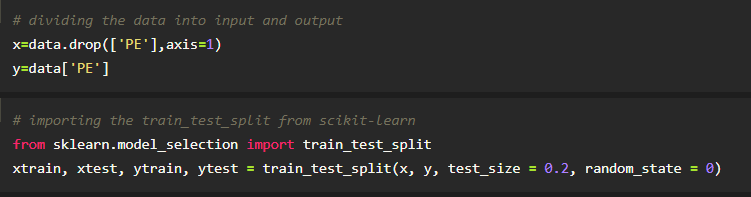




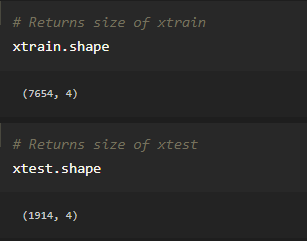




**Activity 6** : **Splitting Data into Train and Test**

* When you are working on a model and you want to train it, you have a dataset. But after training, we have to test the model on some test dataset. For this, you will a dataset which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.
* 
* **Scikit-learn:** Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction. We can also called sklearn as short.
* **Sklearn.model\_selection:** train\_test\_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually. By default, Sklearn train\_test\_split will make random partitions for the two subsets.
* **Test\_size():** This parameter decides the size of the data that has to be split as the test dataset. This is given as a fraction. For example, if you pass 0.5 as the value, the dataset will be split 50% as the test dataset. If you're specifying this parameter, you can ignore the next parameter.

The size of train and test data after splitting



**Milestone 3: Model Building**

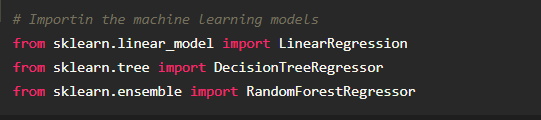
The model building process involves setting up ways of collecting data, understanding and paying attention to what is important in the data to answer the questions you are asking, finding a statistical, mathematical or a simulation model to gain understanding and make predictions.

Model Building Includes:

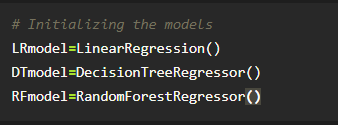
* + Import the model building Libraries
  + Initializing the model
  + Training the model
  + Model Evaluation
  + Save the Model

**Activity 1 : Importing the Model Building Libraries**

Importing the necessary libraries



**Activity 2 : Initializing the model**

****

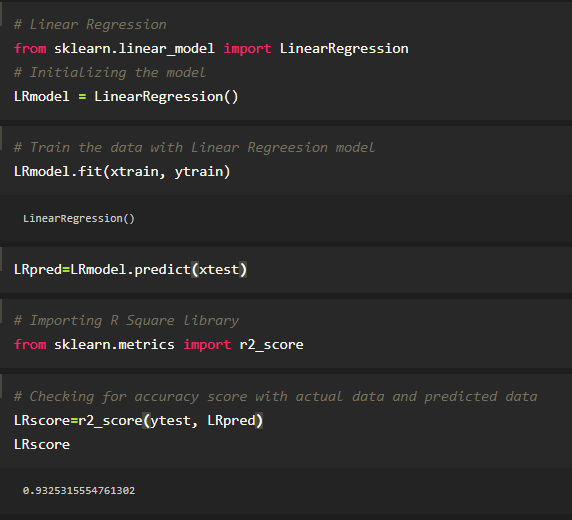
**Activity 6: Train and Testing the model**

In a dataset, a training set is implemented to build up a model, while a test (or validation) set is to validate the model built. Data points in the training set are excluded from the test (validation) set. The models generated are to predict the results unknown which is named as the test set.

Here we are used three different types machine learning models.

**Model 1: Linear Regression**

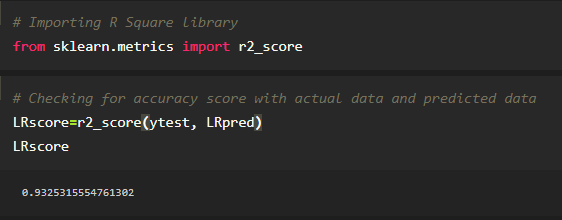
Regression analysis consists of a set of machine learning methods that allow us to predict a continuous outcome variable (y) based on the value of one or multiple predictor variables (x). Linear regression is the most simple and popular technique for predicting a continuous variable. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.



**LRmodel.fit():** Model fitting is a measure of how well a machine learning model generalizes to similar data to that on which it was trained. A model that is well-fitted produces more accurate outcomes.

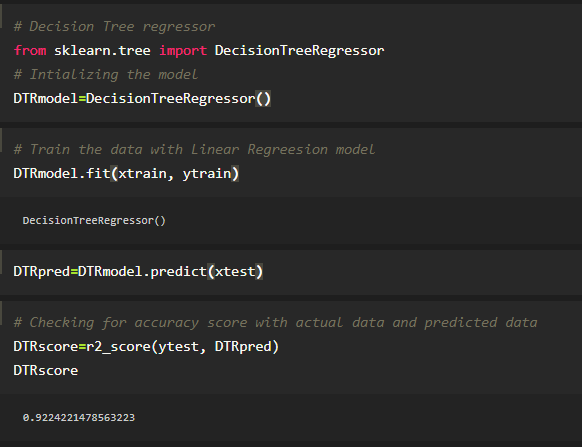
**LRmodel.predict():** The predict() function accepts only a single argument which is usually the data to be tested. It returns the labels of the data passed as argument based upon the learned or trained data obtained from the model. Thus, the predict() function works on top of the trained model and makes use of the learned label to map and predict the labels for the data to be tested.

**r2\_score():** R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.



**Model 2: Decision Tree Regression**

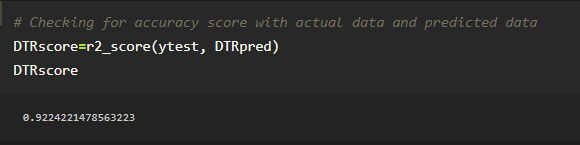
Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.



**DTmodel.fit():** Model fitting is a measure of how well a machine learning model generalizes to similar data to that on which it was trained. A model that is well-fitted produces more accurate outcomes.

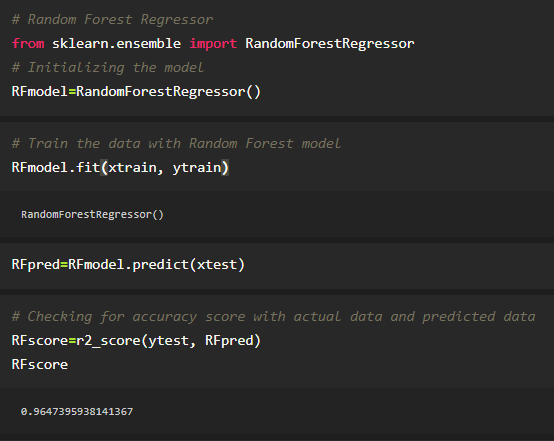
**DTmodel.predict():** The predict() function accepts only a single argument which is usually the data to be tested. It returns the labels of the data passed as argument based upon the learned or trained data obtained from the model. Thus, the predict() function works on top of the trained model and makes use of the learned label to map and predict the labels for the data to be tested.

**r2\_score():** R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.

****

**Model 3: Random Forest Regression**

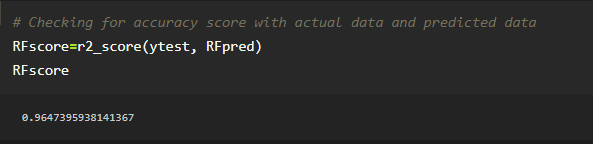
Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

****

**RFmodel.fit():** Model fitting is a measure of how well a machine learning model generalizes to similar data to that on which it was trained. A model that is well-fitted produces more accurate outcomes.

**RFmodel.predict():** The predict() function accepts only a single argument which is usually the data to be tested. It returns the labels of the data passed as argument based upon the learned or trained data obtained from the model. Thus, the predict() function works on top of the trained model and makes use of the learned label to map and predict the labels for the data to be tested.

**r2\_score():** R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.

****

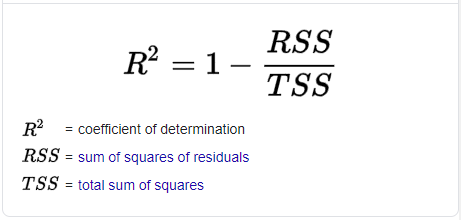
**Activity 7**: **Model Evaluation**

Finally, we need to check to see how well our model is performing on the test data.

**Classification Evaluation Metrics:**

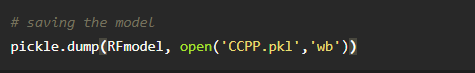
### R Square Score:

R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression. 100% indicates that the model explains all the variability of the response data around its mean.



**Activity 8: Save the Model**

The model is saved with .pkl extension as follows



**Milestone 4: Application Building**

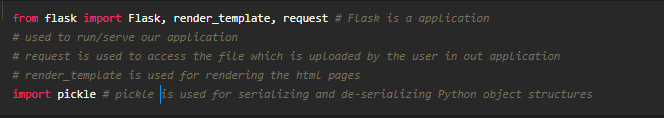
* + **Link**: <https://thesmartbridge.com/documents/spsaimldocs/FlaskML.pdf>

**Activity 1 : Create an HTML File**

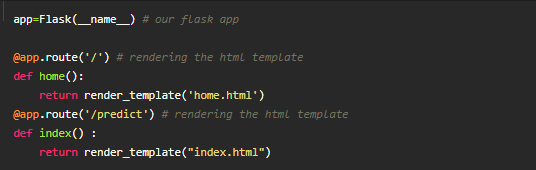
* We use HTML to create the front end part of the web page.
* Here, we created 4 html pages- index.html, home.html, chance.html, nochance.html.
* Home page displays the introduction of project
* index.html accepts the values from the input and displays the prediction.
* For more information regarding HTML refer the link below.
  + **Link:** [**https://www.w3schools.com/bootstrap/bootstrap\_forms\_inputs.asp**](https://www.w3schools.com/bootstrap/bootstrap_forms_inputs.asp)
  + **Link:**<https://www.w3schools.com/css/>

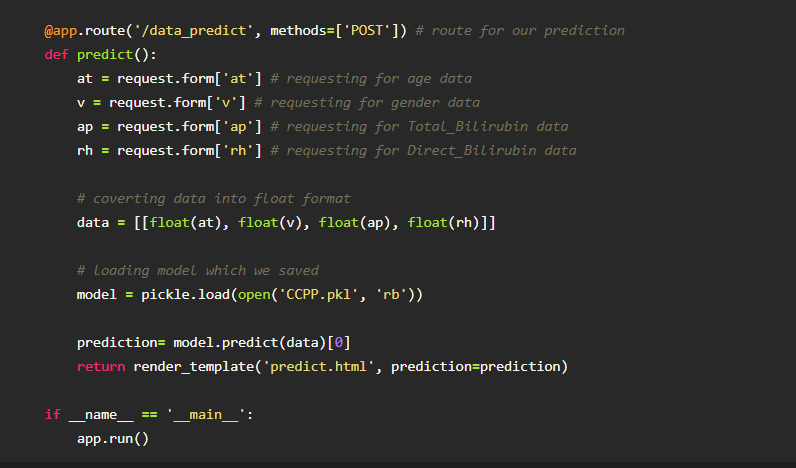
**Activity 2 : Build python code**

* Let us build flask file ‘Liver\_Flask\_App.ipynb’ which is a web framework written in python for server-side scripting. Let’s see step by step procedure for building the backend application.
* App starts running when “\_\_name\_\_” constructor is called in main.
* render\_template is used to return html file.
* “GET” method is used to take input from the user.
* “POST” method is used to display the output to the user.
* Importing Libraries

****

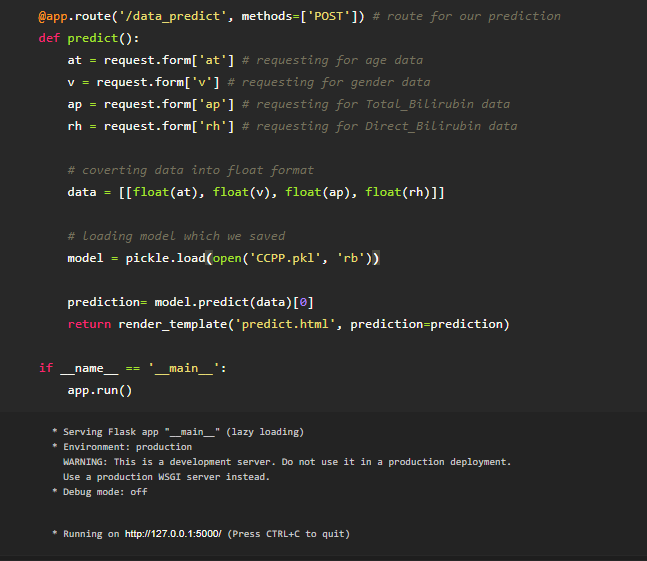
* + Routing to the html Page



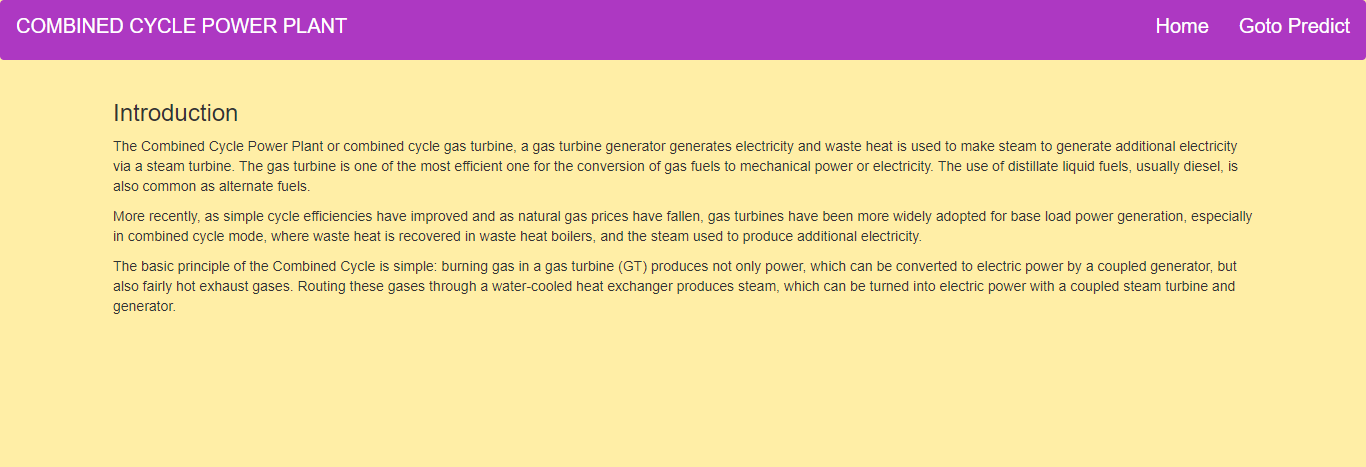


**Activity 3: Run The app in local browser**

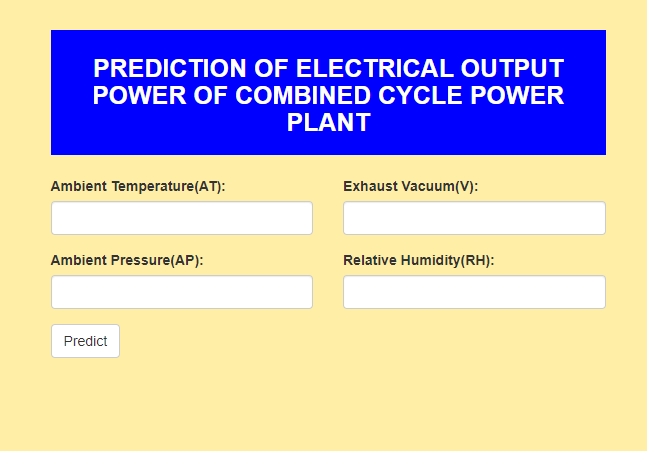
* + Open jupyter notebook from the start menu
  + Navigate to the folder where your python script is.
  + Now press Ctrl+Enter
  + Navigate to the localhost where you can view your web page

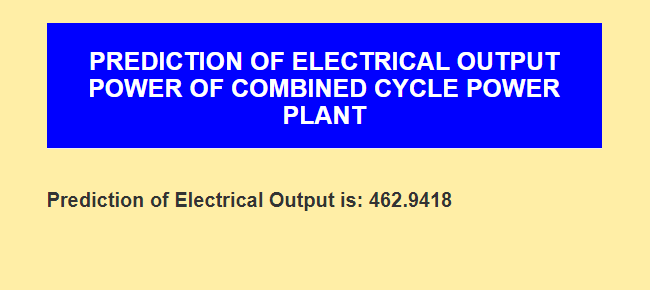


**Activity 4: Showcasing prediction on UI**

****

This is our home page where we get to know the summary of the project.

****

****

As we see the predicted output is displayed on the User Interface