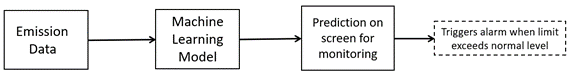
**H1B Visa Approval Prediction**

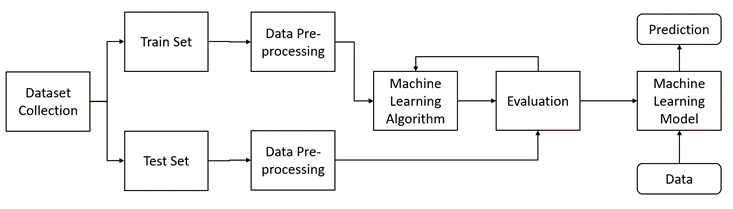
The H-1B Dataset selected for this project contains data from employer’s Labor Condition Application and the case certification determinations processed by the Office of Foreign Labor Certification (OFLC) where the date of the determination was issues on or after October 1, 2016 and on or before June 30, 2017.

The Labor Condition Application (LCA) is a document that a perspective H-1B employer files with U.S. Department of Labor Employment and Training Administration (DOLETA) when it seeks to employ non-immigrant workers at a specific job occupation in an area of intended employment for not more than three years.

**Block Diagram:**



**Machine Learning Workflow:**



**Project Flow:**

* User interacts with the UI (User Interface) to enter the current mine working conditions.
* Entered readings are analyzed and predictions are made based on interpretation that whether mine will explode or situation will lead to explosion.
* Predictions are popped onto the UI.

**Prerequisites:**

1. **To develop this project, we need to install following software/packages:**

**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 so very nice tools like JupyterLab, Jupyter Notebook,

QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupyter notebook and Visual studio code.

To install Anaconda navigator and to know how to use Jupyter Notebook & Spyder using Anaconda watch the video

Link:  [Click here to](https://www.youtube.com/watch?v=5mDYijMfSzs&feature=emb_logo) Watch video

1. **To build a Mine explosion prediction**

* **Install sci-kit learn**

**Scikit**-**learn** (**Sklearn**) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction via a consistence interface in Python.

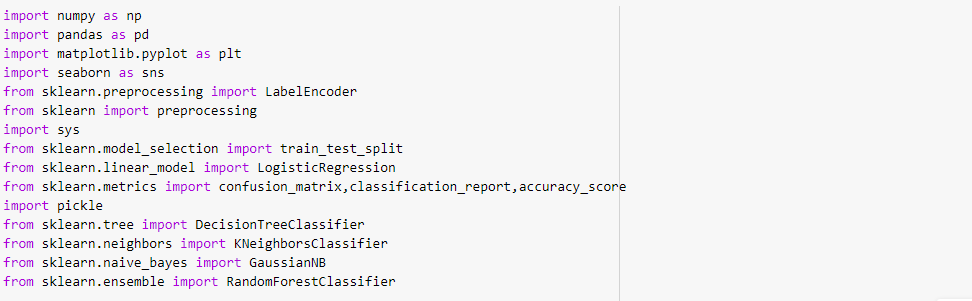
* **Install Flask**
* Flask is a Web application framework written in python
* For installation of Flask
  + Open your anaconda prompt and Type “pip install Flask”

**1. Data Collection**

Download dataset /create dataset: The given dataset is related to H1B Visa Prediction. Dataset was downloaded from kaggle.com website.

**2. Data Pre-processing**

Importing required Libraries:



**Pandas:** It is a python library mainly used for data manipulation.

**NumPy:** This python library is used for numerical analysis.

**Matplotlib and Seaborn:** Both are the data visualization library used for plotting graph which will help us for understanding the data.

**Accuracy score:** used in classification type problem and for finding accuracy it is used.

**Train\_test\_split:** used for splitting data arrays into training data and for testing data.

**Scikit-learn (Sklearn)** is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction via a consistence interface in Python.

**Pickle:** It is used for dumping of values.

Importing the dataset:



* You might have your data in .csv files, .excel files or .tsv files or something else. But the goal is the same in all cases. If you want to analyse that data using pandas, the first step will be to read it into a data structure that’s compatible with pandas.

* Let’s load a .csv data file into pandas. There is a function for it, called **read\_csv().**We will need to locate the directory of the CSV file at first (it’s more efficient to keep the dataset in the same directory as your program).

* Path names on Windows tend to have backslashes in them. But we want them to mean actual backslashes, not special characters.

Data Visualization:

Exploratory data analysis is an approach to analyzing data sets to summarize their main characteristics, often with visual methods and used for determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

* To check first five rows of dataset, we have a function call **head( ).**

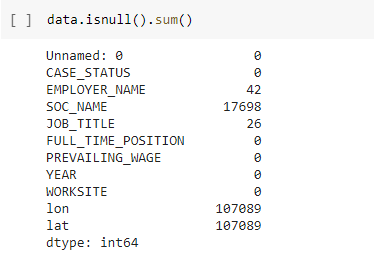


* This head () function returns the first 5 rows for the object based on position. It

is useful for quickly testing if your object has the right type of data in it.

* To check last five rows of dataset, we have a function call **tail().**
* For finding the names of the columns present in the dataset we make use of **columns**
* **data.columns** will return you all the column names which are present in your data.
* Taking care of Missing Data:

Sometimes you may find some data are missing in the dataset. We need to be equipped to handle the problem when we come across them. Obviously, you could remove the entire line of data but what if you are unknowingly removing crucial information? Of course we would not want to do that. One of the most common ideas to handle the problem is to take a mean of all the values for continuous and for categorical we make use of mode values and replace the missing data.

* We will be using **isnull().sum()** method to see which column has missing values.
* 
* Since there are missing values in the dataset, we need to execute these following step.
* 
* Label encoding

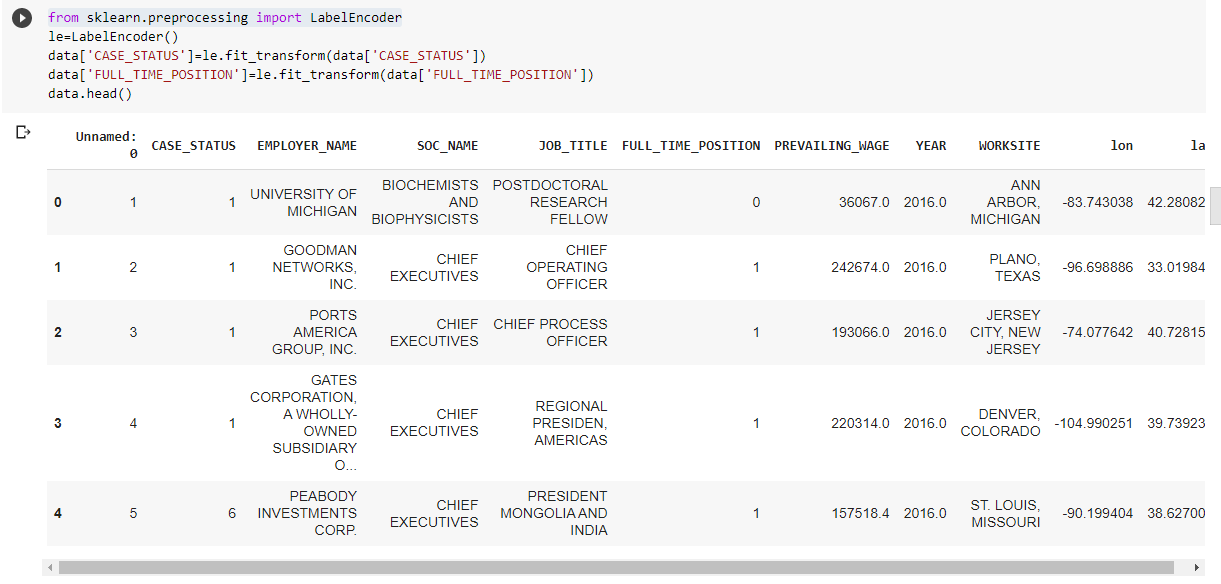
Typically, any structured dataset includes multiple columns with combination of numerical as well as categorical variables. A machine can only understand the numbers. It cannot understand the text. That’s essentially the case with [Machine Learning algorithms](https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/?utm_source=blog&utm_medium=one-hot-encoding-vs-label-encoding-using-scikit-learn) too. We need to convert each text category to numbers in order for the machine to process those using mathematical equations.

How should we handle categorical variables? There are Multiple way to handle, but will see one of it is LabelEncoding.

* **Label Encoding** is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering.

Let’s see how to implement label encoding in Python using the [scikit-learn library](https://www.analyticsvidhya.com/blog/2020/02/everything-you-should-know-scikit-learn/?utm_source=blog&utm_medium=one-hot-encoding-vs-label-encoding-using-scikit-learn).

As we have to convert only the text class category columns, we first select it then we will implement Label Encoding to it.



* Feature Scaling
* Splitting Data into Train and Test:

When you are working on a model and you want to train it, you obviously 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.

But the question is, how do you split the data? You can’t possibly manually split the dataset into two sets. And you also have to make sure you split the data in a random manner. To help us with this task, the Scikit library provides a tool, called the Model Selection library. There is a class in the library which is, **‘[train\_test\_split](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html).’** Using this we can easily split the dataset into the training and the testing datasets in various proportions.

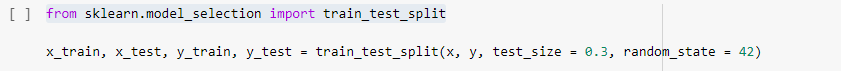
The train-test split is a technique for evaluating the performance of a machine learning algorithm.

* **Train Dataset**: Used to fit the machine learning model.
* **Test Dataset**: Used to evaluate the fit machine learning model.

In general you can allocate 80% of the dataset to training set and the remaining 20% to test set.

We will create 4 sets— X\_train (training part of the matrix of features), X\_test (test part of the matrix of features), Y\_train (training part of the dependent variables associated with the X train sets, and therefore also the same indices), Y\_test (test part of the dependent variables associated with the X test sets, and therefore also the same indices.

There are a few other parameters that we need to understand before we use the class:

* **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
* **train\_size** — you have to specify this parameter only if you’re not specifying the test\_size. This is the same as test\_size, but instead you tell the class what percent of the dataset you want to split as the training set.
* **random\_state** — here you pass an integer, which will act as the seed for the random number generator during the split. Or, you can also pass an instance of the Random\_state class, which will become the number generator. If you don’t pass anything, the Random\_state instance used by np.random will be used instead.
* Now split our dataset into train set and test using train\_test\_split class from scikit learn library.
* 
* **3. Model Building**

Training and testing the model:

There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms that you can choose according to the objective that you might have it may be Classification algorithms are Regression algorithms.

Example: 1. Linear Regression.

2. Logistic Regression.

3. Random Forest Regression / Classification.

4. Decision Tree Regression / Classification.

You will need to train the datasets to run smoothly and see an incremental improvement in the prediction rate.

**Now we apply Logistic regression algorithm on our dataset.**

**Logistic Regression** is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In **logistic regression**, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

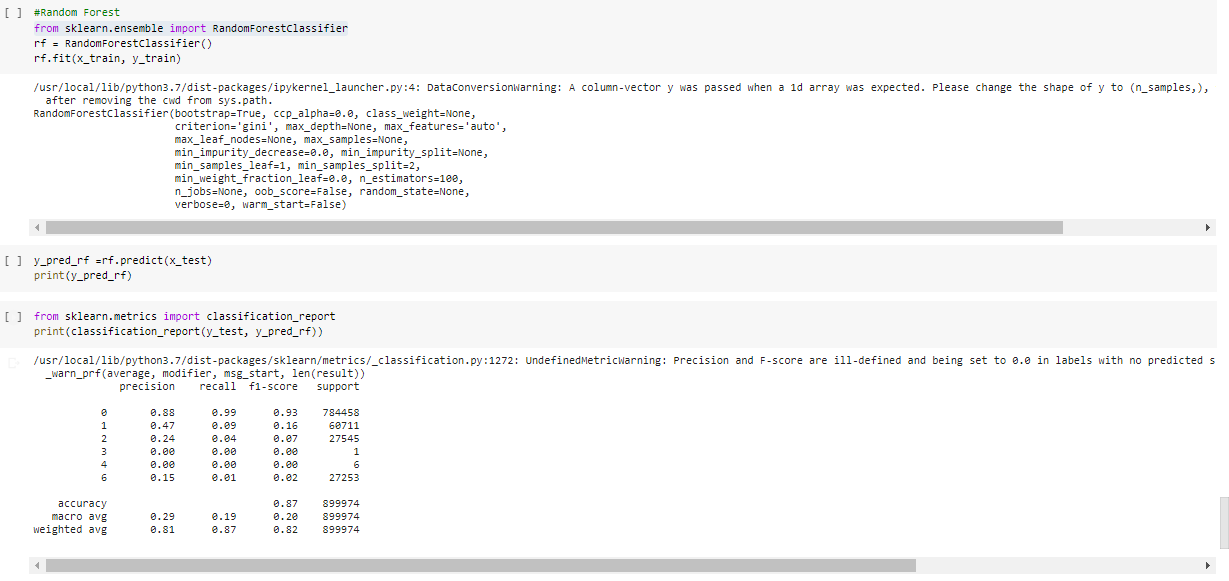
**Predict the values**

Once the model is trained, it’s ready to make predictions. We can use the **predict** method on the model and pass **x\_test** as a parameter to get the output as **pred.**

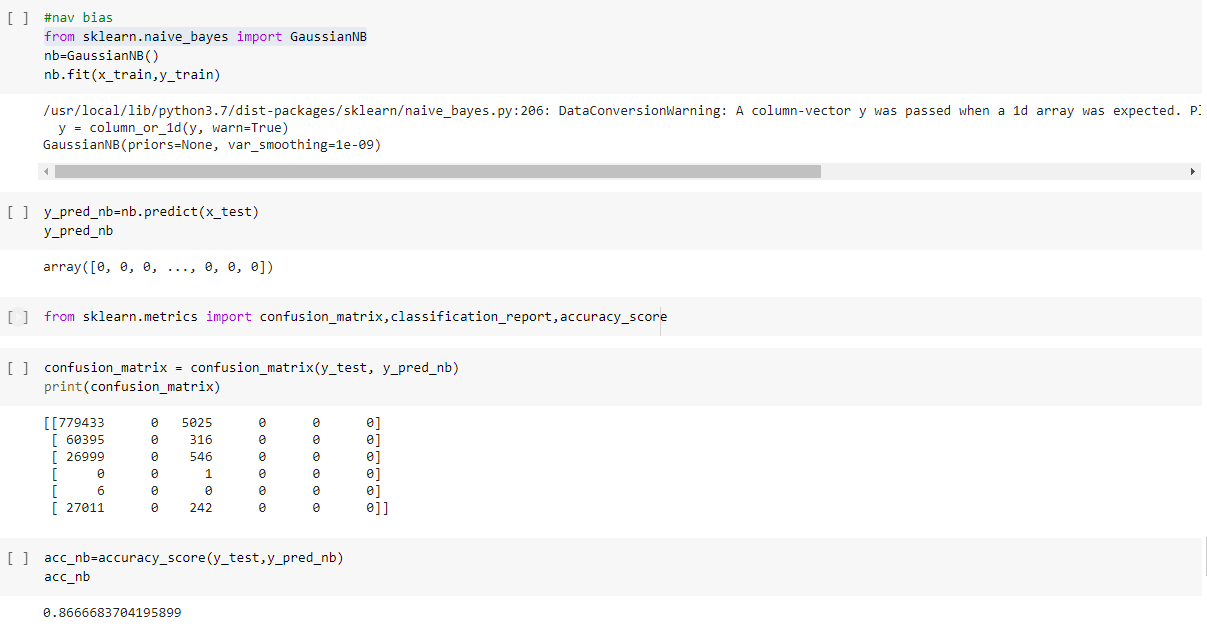
Notice that the prediction output is an array of real numbers corresponding to the input array.



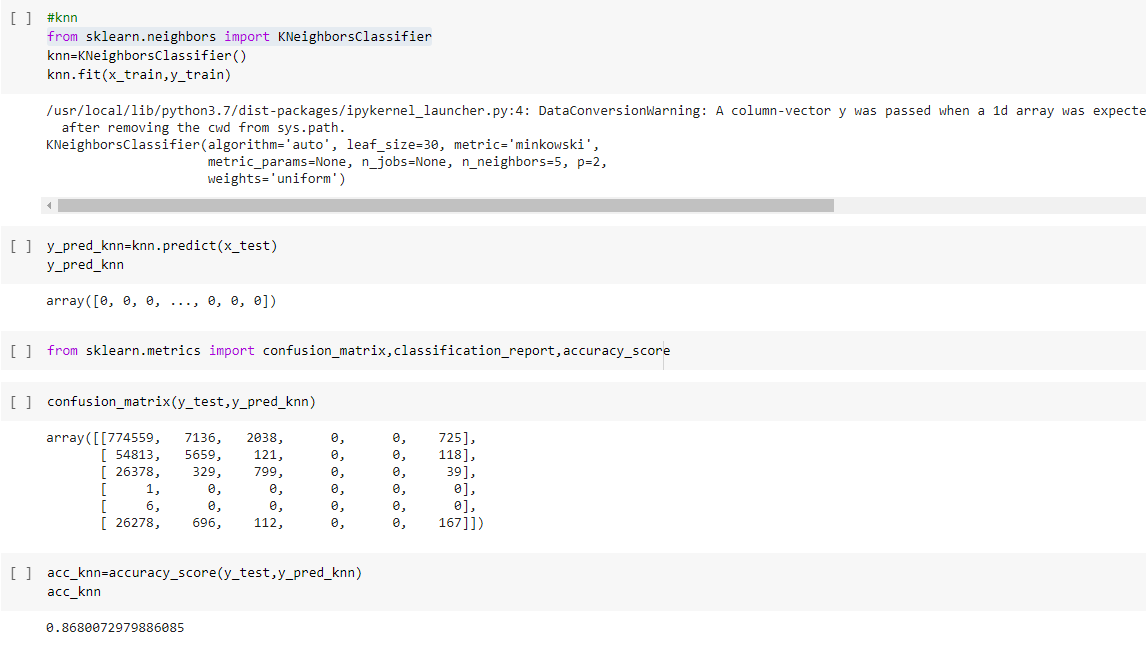
**Random Forest Classification**



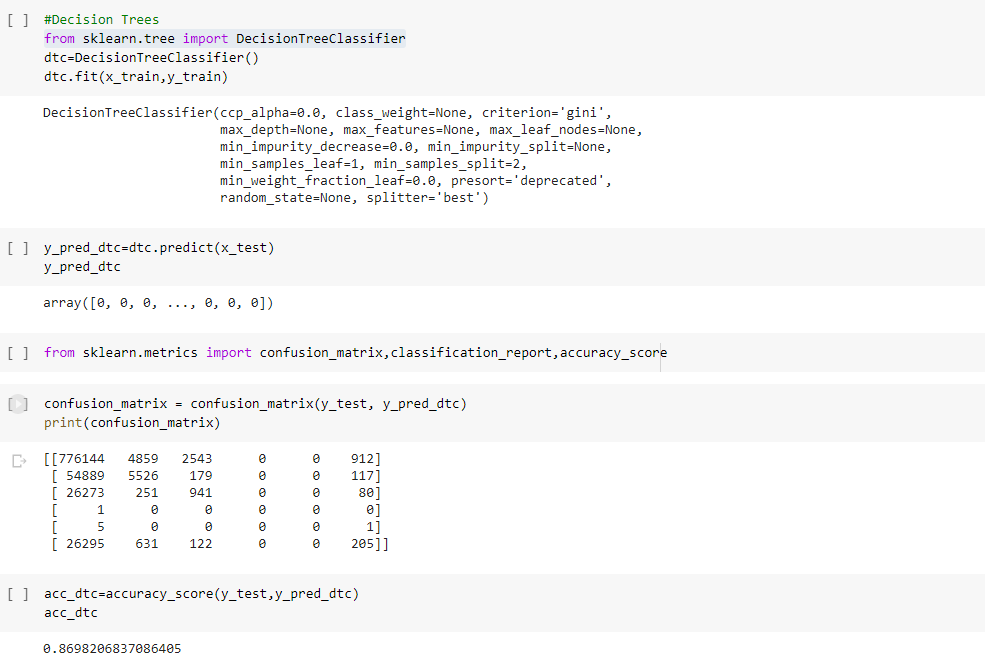
**Naïve Bayes**



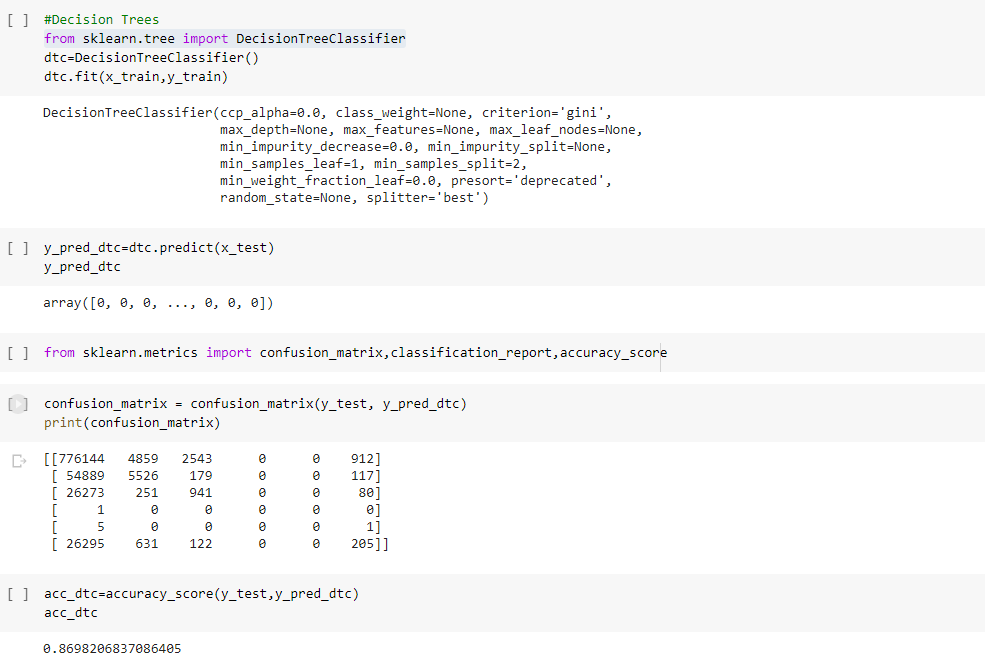
**K Nearest Neighbour**



**Decision Tree Classifier**



A **confusion matrix** is a tabular summary of the number of correct and incorrect predictions made by a classifier. It can be used to evaluate the performance of a classification model through the calculation of performance metrics like accuracy, precision, recall, and F1-score.



**Saving a model:**

Model is saved so it can be used in future and no need to train it again.



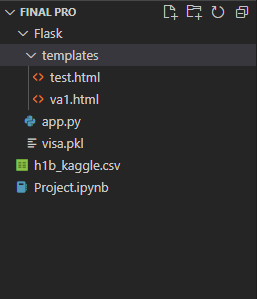
**4. Application Building**

Creating a HTML File, flask application.

* Build python code
  + Importing Libraries
  + Routing to the html Page
  + Showcasing prediction on UI
  + Run The app in local browser

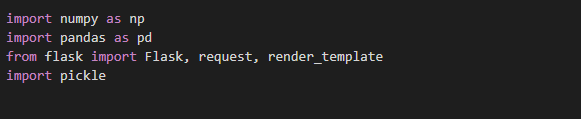
**Project Structure:**

Create a Project folder that contains files as shown below



* We are building a Flask Application that needs HTML pages stored in the templates folder
* Templates folder contains index.html
* Static folder contains CSS and image files.

**Task 1: Importing Libraries**

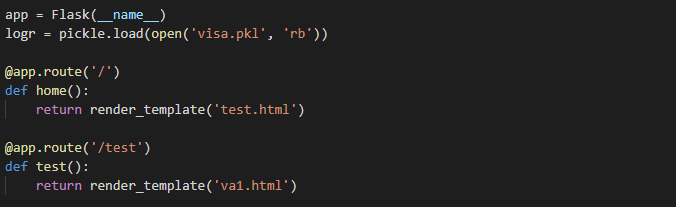
Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (\_\_name\_\_) as argument Pickle library to load the model file.

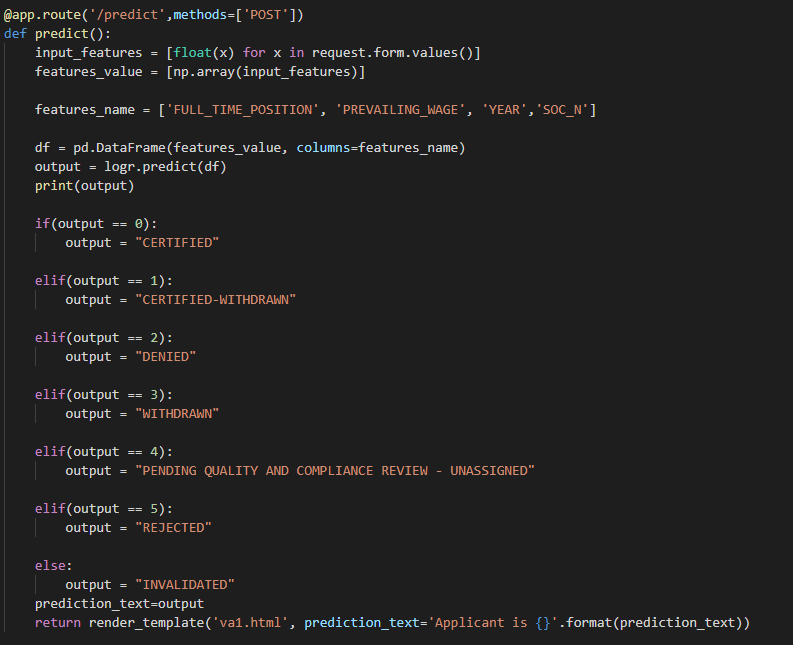
**Task 2: Routing to the html Page**

Here, declared constructor is used to route to the HTML page 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 is rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

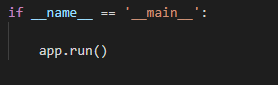
Here, “index.html” is rendered when home button is clicked on the UI





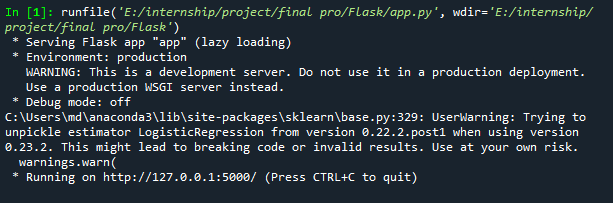
**Task 3: Main Function**

This is used to run the application in a local host.



**Activity 3: Run the application**

* Open the anaconda prompt from the start menu.
* Navigate to the folder where your app.py resides.
* Now type “python app.py” command.
* It will show the local host where your app is running on **http://127.0.0.1:5000/**
* Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.
* Enter the values, click on the predict button and see the result/prediction on the web page.



**Output Screen:**



