Wine Quality Prediction using ML

Project Idea:

Wine is an alcoholic drink made from fermented grapes. Yeast consumes the sugar in the

grapes and converts it to ethanol, carbon dioxide, and heat. Different varieties of grapes and

strains of yeasts produce different styles of wine. Wine is categorised based on quality and

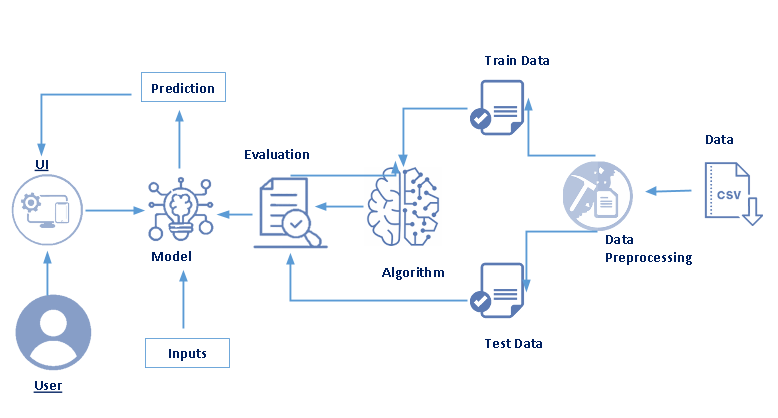
price is set accordingly. These days all types of industries are improving by adopting new

technologies and applying them in all areas to enhance the production. But still some areas

need human expertise such as quality assurance. This process becomes expensive over the

demand of product. To automate this process in area of wine there is need of an application.

Architecture:



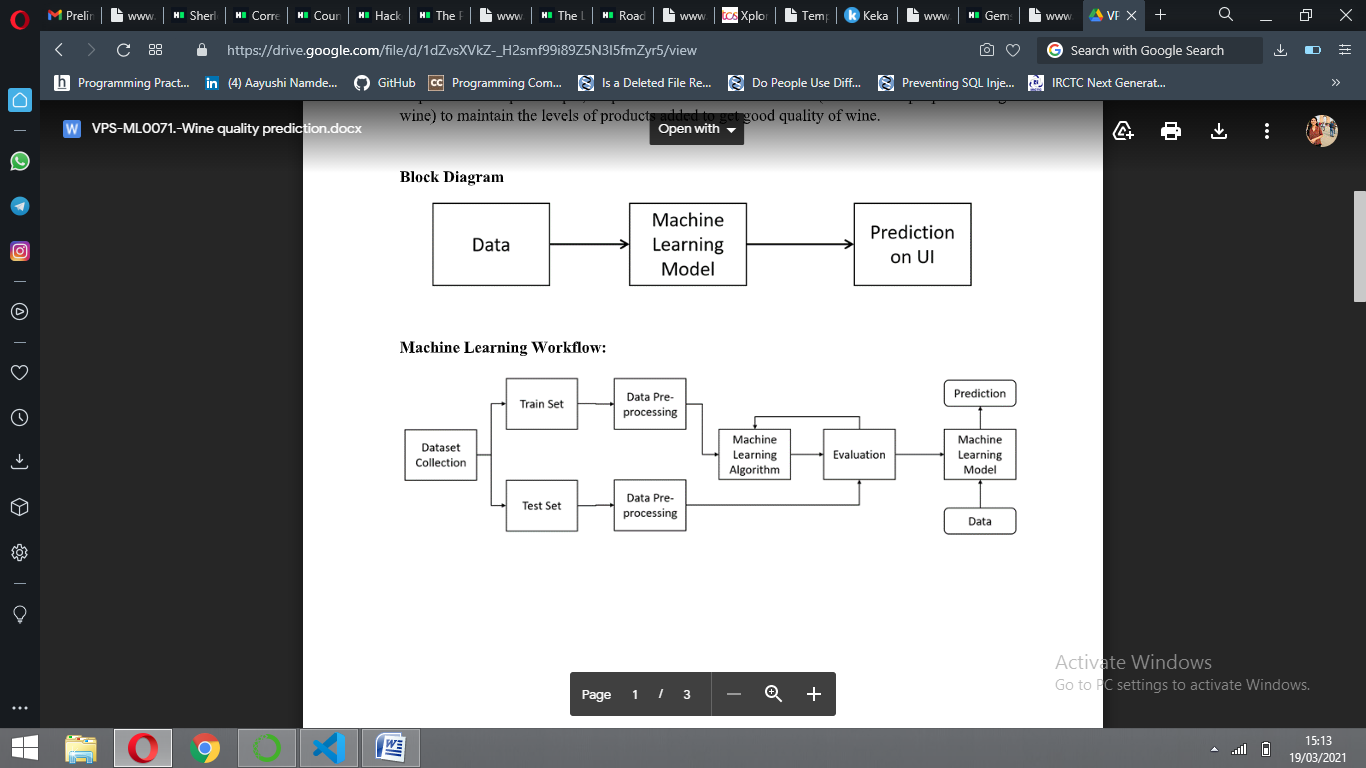
Solution:

An application is built that rates the quality of wine in a range of 1 – 5. The quality of wine

can be predicted considering the factors like volatile and fixed acidity, citric acid added, total

sulphur dioxide present pH, sulphates etc. This enables users (farmers and people making

wine) to maintain the levels of products added to get good quality of wine.



Learning Outcomes:

By the end of this project:

* You’ll be able to understand the problem to classify if it is a regression or a classification kind of problem.
* You will be able to know how to pre-process / clean the data using different data pre-processing techniques.
* You will able to analyze or get insights of data through visualization.
* Applying different algorithms according to dataset and based on visualization.
* You will able to know how to find accuracy of the model.
* You will be able to know how to build a web application using Flask framework.

**Pre requisites:**

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

1. **In order to develop this project we need to install following software’s/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 Spyder.

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 Machine learning models you must require the following packages**

* **Numpy**:
* It is an open-source numerical Python library. It contains a multidimensional array and matrix data structures and can be used to perform mathematical operations
* **Scikit-learn:**
  + It is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy
* **Scikit-image**
  + Scikit-image, or skimage, is an open source Python package designed for image preprocessing.
* **Matplotlib and Seaborn:**
  + Matplotlib is mainly deployed for basic plotting. Visualization using Matplotlib generally consists of bars, pies, lines, scatter plots and so on. Seaborn: Seaborn, on the other hand, provides a variety of visualization patterns. It uses fewer syntax and has easily interesting default themes.
* **Wordcloud:**
  + A word cloud is an image made of words that together resemble a cloudy shape.The size of a word shows how important it is e.g. how often it appears in a text.
* **Flask:** 
  + Web framework used for building Web applications

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

* Open anaconda prompt.
* Type “pip install jupyter notebook” and click enter.
* Type “pip install spyder” and click enter.
* Type “pip install numpy” and click enter.
* Type “pip install pandas” and click enter.
* Type “pip install matplotlib” and click enter.
* Type “pip install seaborn” and click enter.
* Type “pip install sklearn” and click enter.
* Type “pip install WordCloud” and click enter.
* Type “pip install Flask” and click enter.

If you are using Pycharm IDE, you can install the packages through the command prompt and follow the same syntax as above.

**Prior Knowledge:**

One should have knowledge on the following Concepts:

**Link:**[Supervised and Unsupervised Learning](https://www.youtube.com/watch?v=kE5QZ8G_78c&feature=emb_logo)

Watch the below video to know about the types of machine learning

**Link:**[Regression, Classification and Clustering](https://www.youtube.com/watch?v=6za9_mh3uTE&feature=emb_logo)

**Link:**[Linear Regression:](https://www.youtube.com/watch?v=1-OGRohmH2s)

**Link:** [Flask:](https://www.youtube.com/watch?v=lj4I_CvBnt0)

It is recommended to watch above video’s to understand the concepts before you start your project.

Project Work Flow:

* User interacts with the UI (User Interface) to upload the input features.
* Uploaded features/input is analyzed by the model which is integrated.
* Once model analyses the uploaded inputs, the prediction is showcased on the UI.

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

**Tasks:**

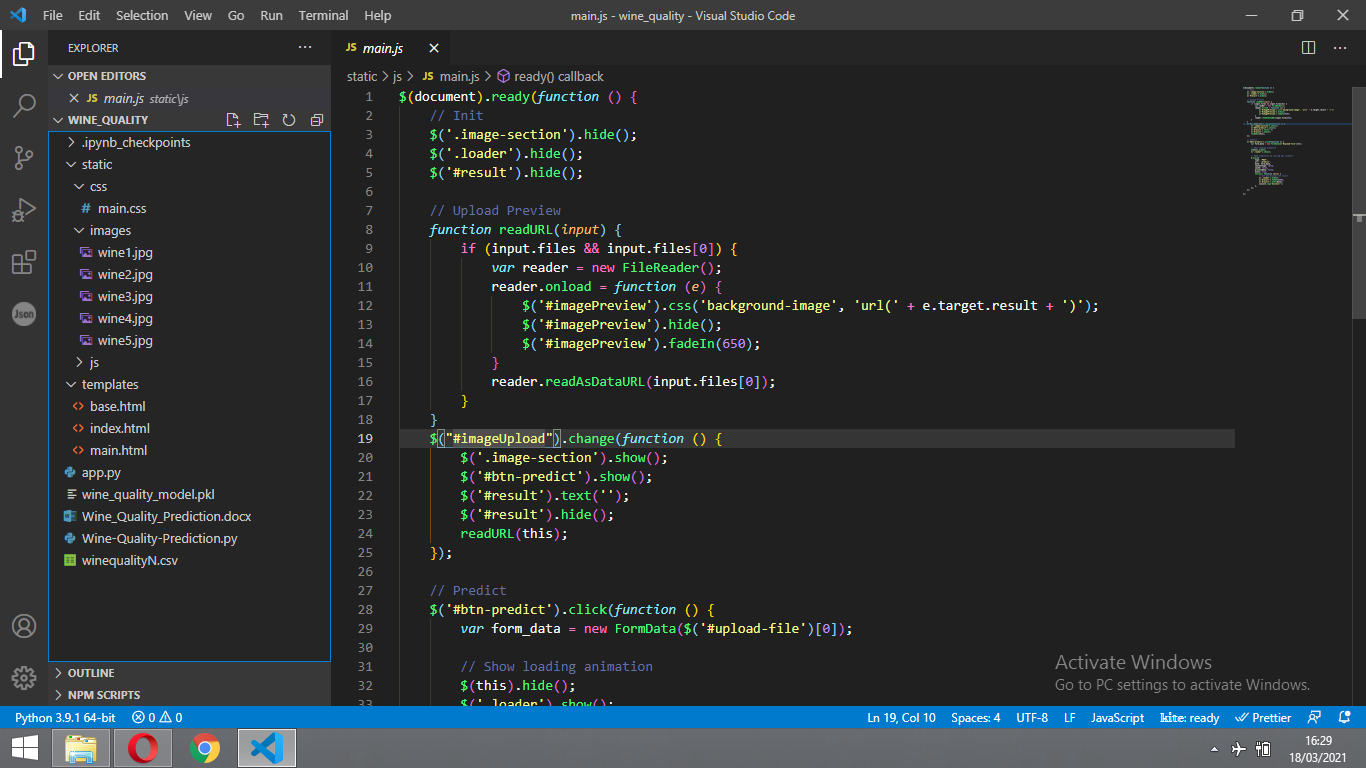
1. Data Collection.

* Collect the dataset or Create the dataset

1. Data Pre- processing.
   1. Import the Libraries.
   2. Reading the dataset.
   3. Exploratory Data Analysis
   4. Converting json objects to strings
   5. Checking for Null Values.
   6. Data Visualization.
   7. Dropping the columns
   8. Label Encoding
   9. Splitting the Dataset into Dependent and Independent variable.
   10. Feature scaling
   11. Splitting Data into Train and Test.
2. Model Building
   1. Training and testing the model
   2. Evaluation of Model
   3. Save the model
   4. Predicting the output using the model
3. Application Building
   1. Create an HTML file
   2. Build a Python Code
   3. Run the app

**Project Structure:**

Create a Project folder which contains files as shown below

****

* We have folders static and templates.
* A python file called app.py for server side scipting.
* We need the model which is saved and the saved model in this content is (**wine\_quality\_model.pkl**).
* Templates folder which contains index.html and main.html files.
* Static folder which contains css folder which contains main.css.

The above project can be downloaded from :

**Milestone 1: Dataset Collection**

ML depends heavily on data, without data, it is impossible for an “AI” to 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: Collect the dataset or create the dataset**

Our base data consists of a CSV file containing information about wine categories and the list of factors on which its quality depends.

* winequalityN.csv : Contains information like the wine type, fixed acidity, volatile acidity, citric acid, residual sugar etc.

You can collect dataset 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://www.kaggle.com/rajyellow46/wine-quality>

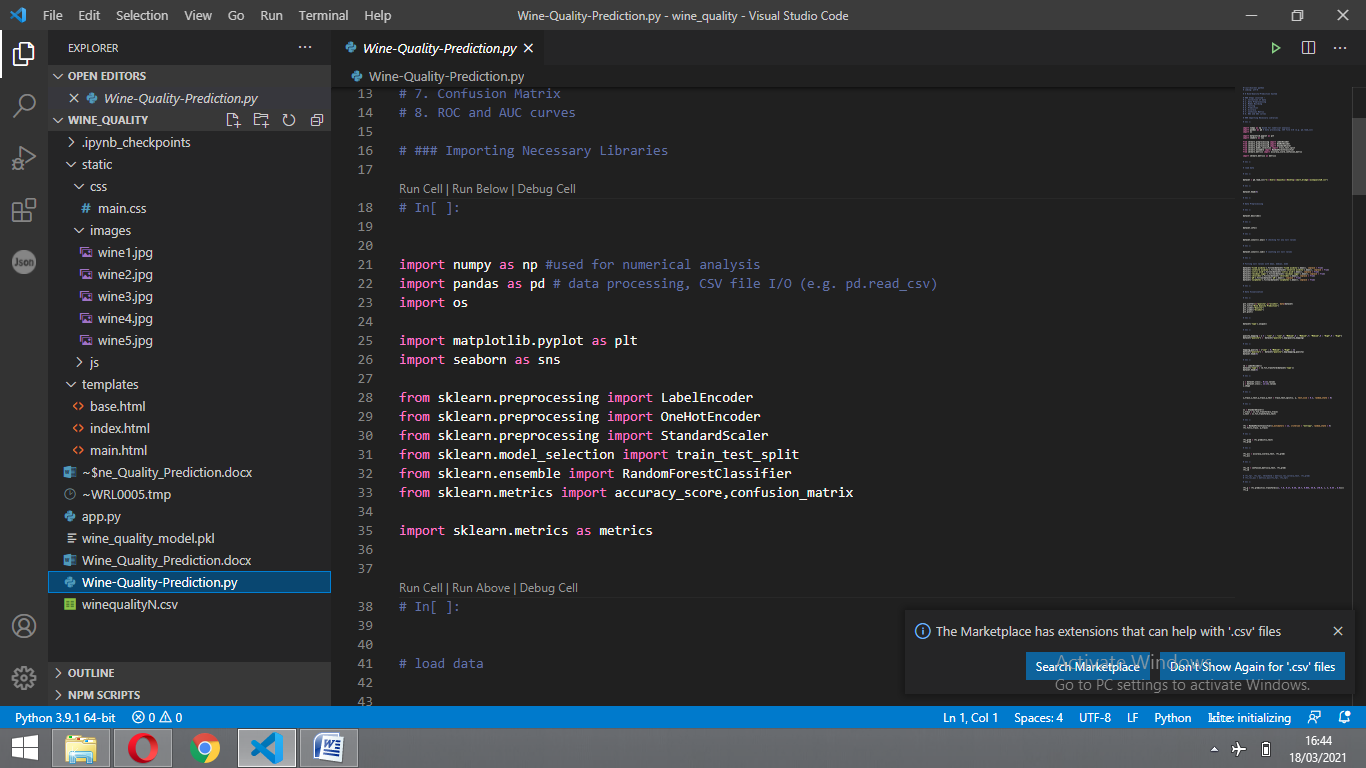
**Milestone 2: Data Pre-processing**

Data Pre-processing includes the following main tasks

1. Import the Libraries.
2. Reading the dataset.
3. Exploratory Data Analysis
4. Converting json objects to strings
5. Checking for Null Values.
6. Data Visualization.
7. Dropping the columns
8. Label Encoding
9. Splitting the Dataset into Dependent and Independent variable.
10. Feature scaling
11. Splitting Data into Train and Test

**Activity 1: Importing the Libraries**

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

****

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

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

**OS:** OS module in Python provides functions for interacting with the operating system

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

**LabelEncoder** :  It can also be **used** to transform non-numerical labels (as long as they are hashable and comparable) to numerical labels.

**OneHotEncoder**  : It refers to splitting the column which contains numerical categorical data to many columns depending on the number of categories present in that column.

**StandardScaler** :  it **will** transform your data such that its distribution **will** have a mean value 0 and **standard** deviation of 1.

**train\_test\_split:** splitting data arrays into two subsets: for training data and for testing data.

**Random Forest**: ensemble model made of many decision **trees** using bootstrapping, random subsets of features, and average voting to make predictions.

**Confusion matrix:** N x N **matrix** used for evaluating the performance of a classification model

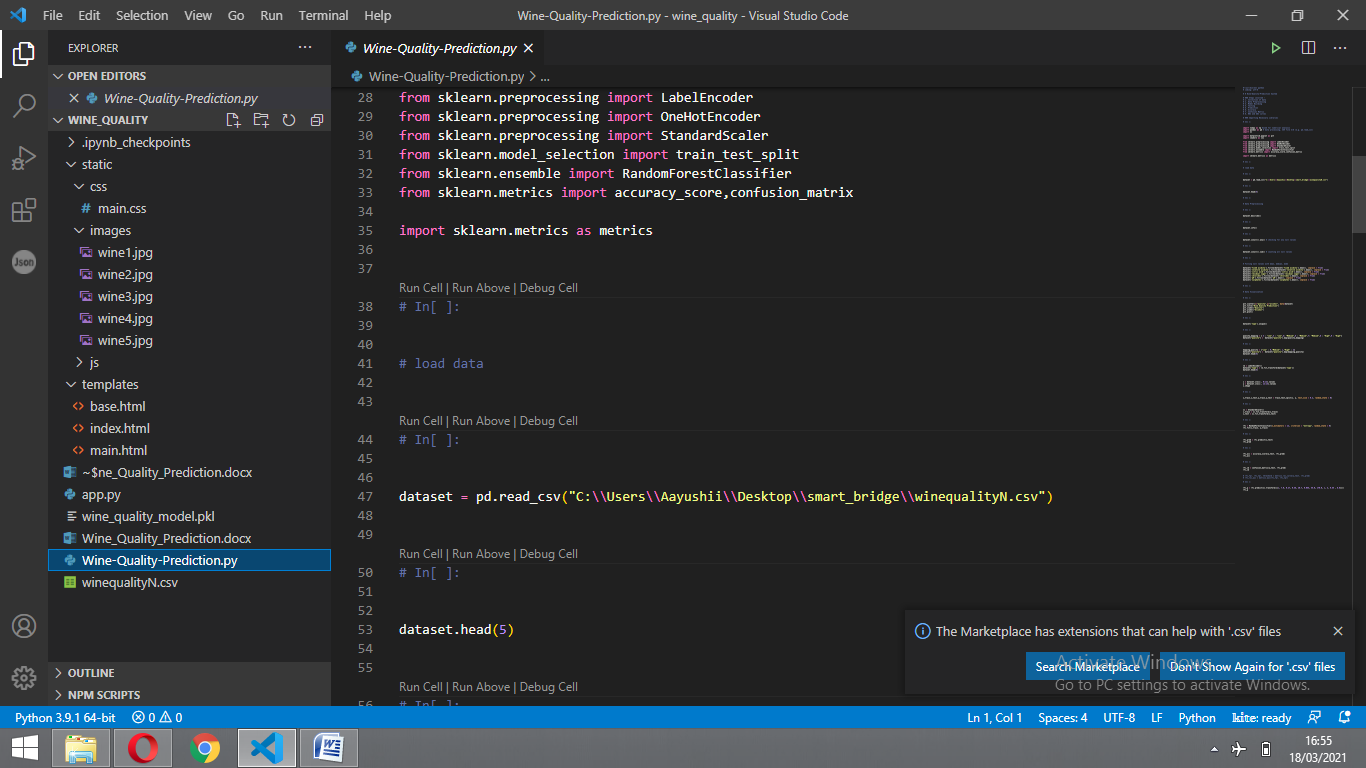
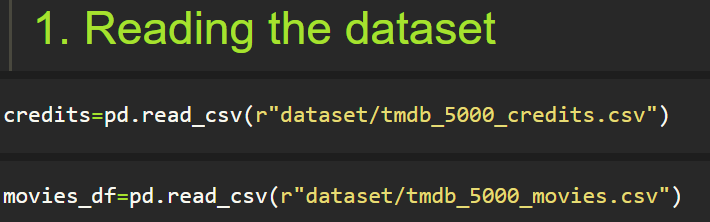
**Accuracy score:** used in classification type problem and for finding accuracy it is used.

**R2 Score:**Coefficient of Determination or R² is another metric used for evaluating the performance of a regression model. The metric helps us to compare our current model with a constant baseline and tells us how much our model is better.

**Pickle:**to serialize your machine learning algorithms and save the serialized format to a file.

Activity 2: Reading 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).
* names on Windows tend to have backslashes in them. But we want them to mean actual backslashes, not special characters.



**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: Exploratory Data Analysis**

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.

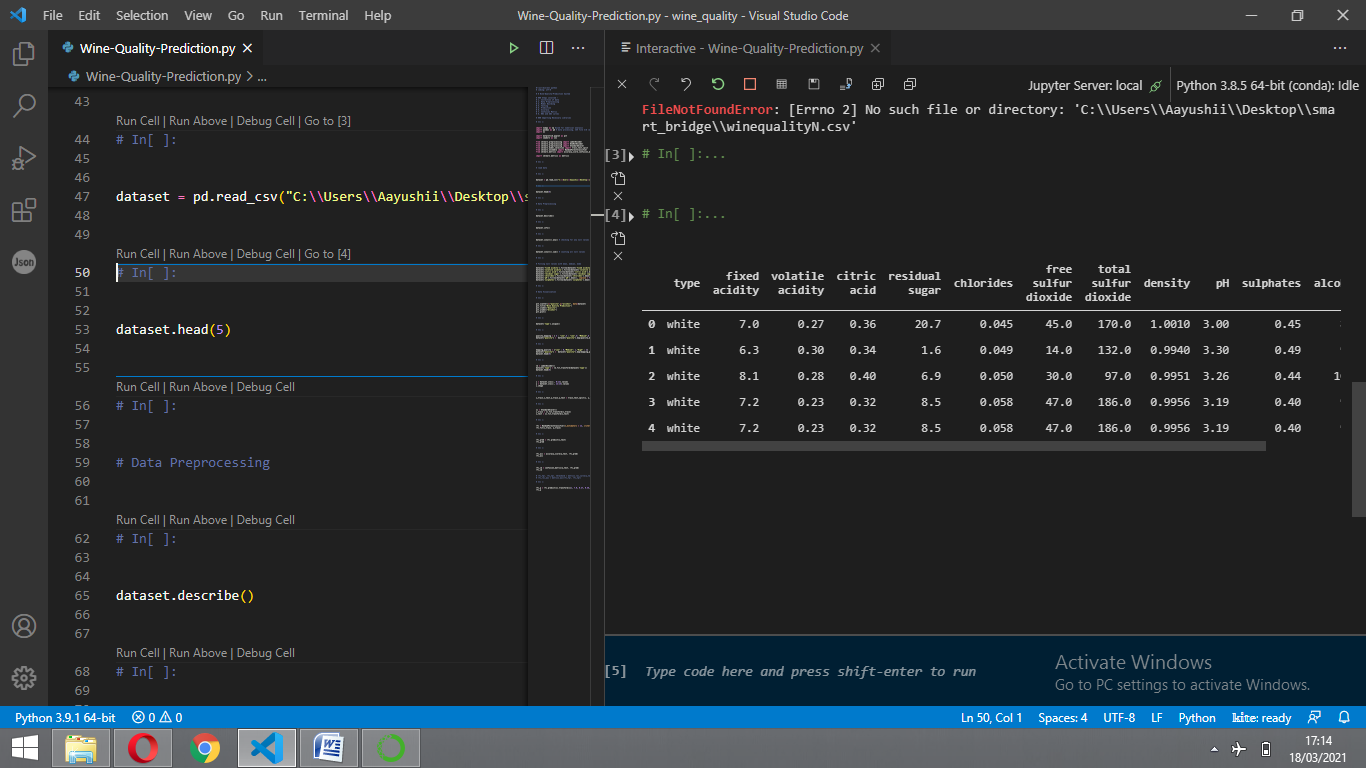
**head() :**To check first five rows of dataset, we have a function call **head( ).**

dataset.head(5)

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

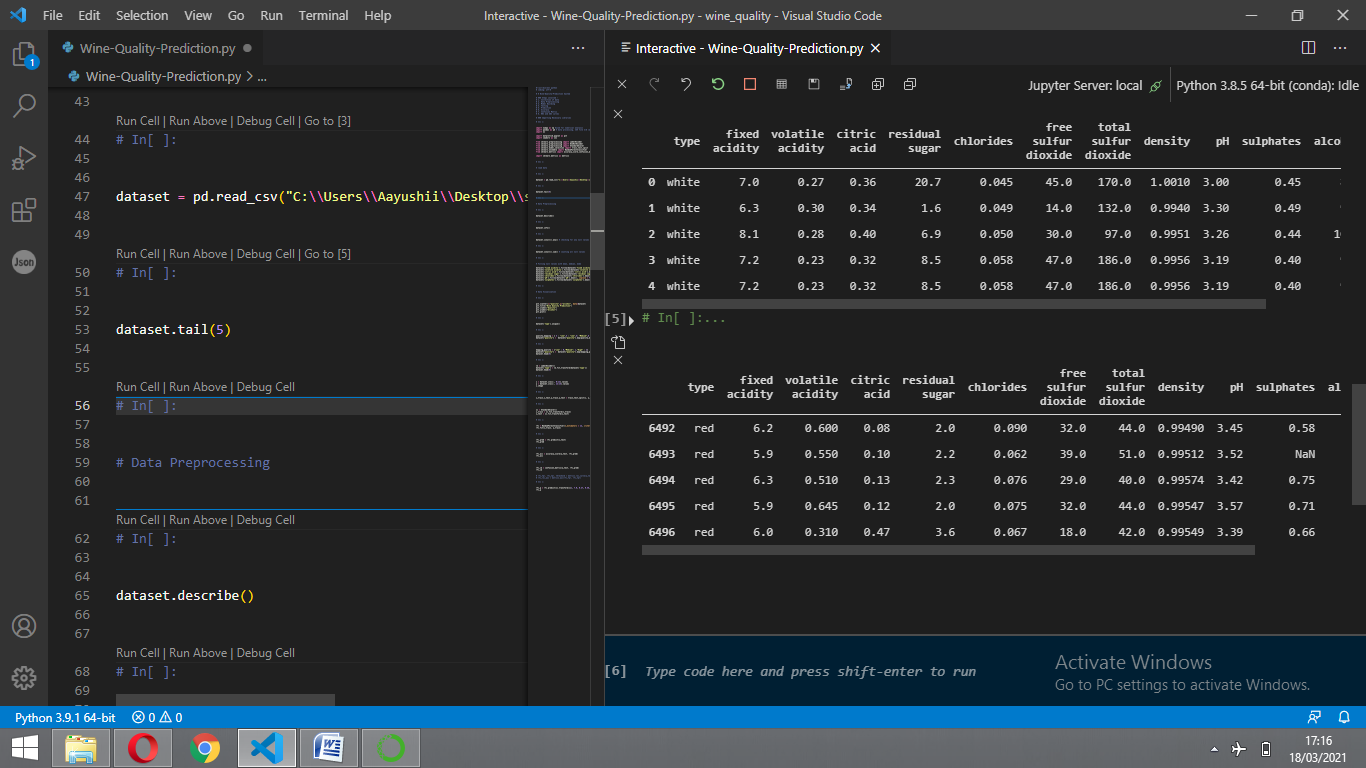
is useful for quickly testing if your object has the right type of data in it.

Dataset :dataset

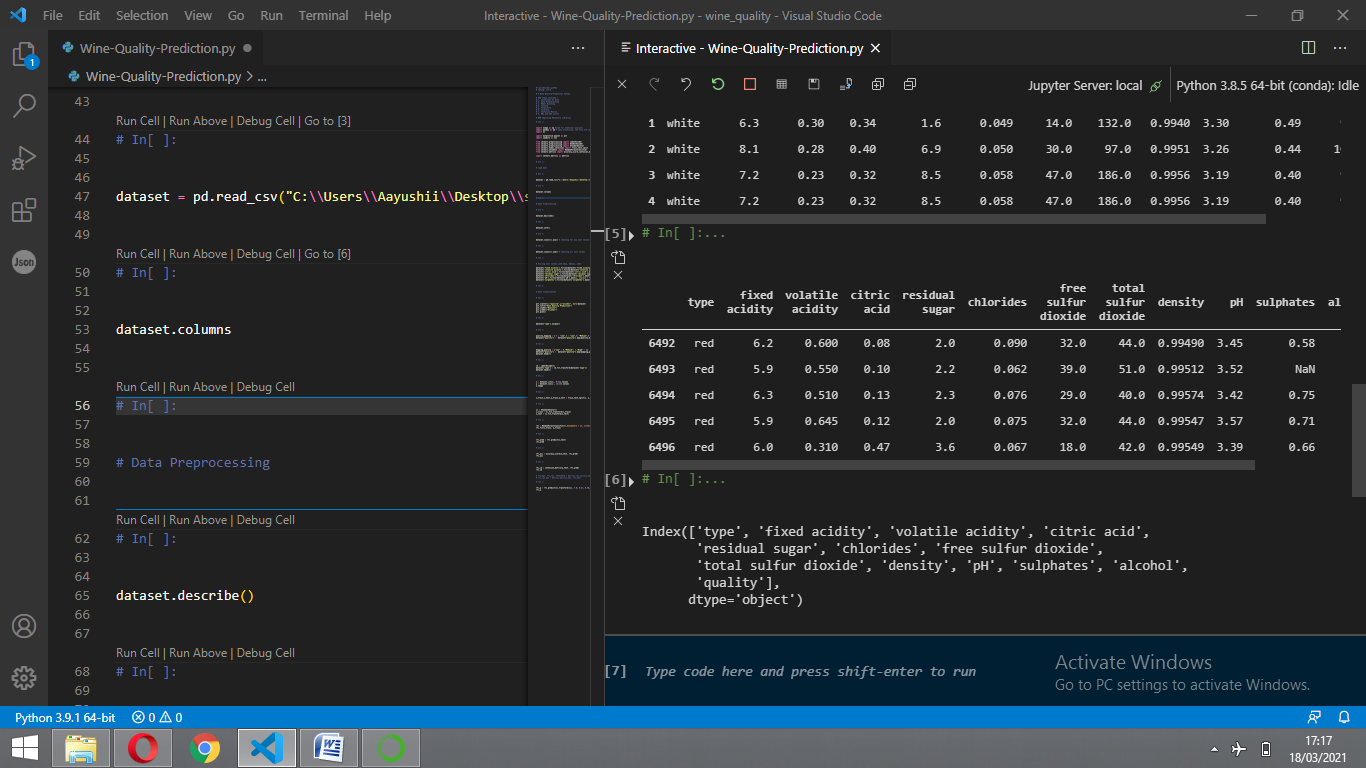


**Tail():** To check last five rows of dataset, we have a function call **tail().**

dataset.tail()

****

**columns:**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 .

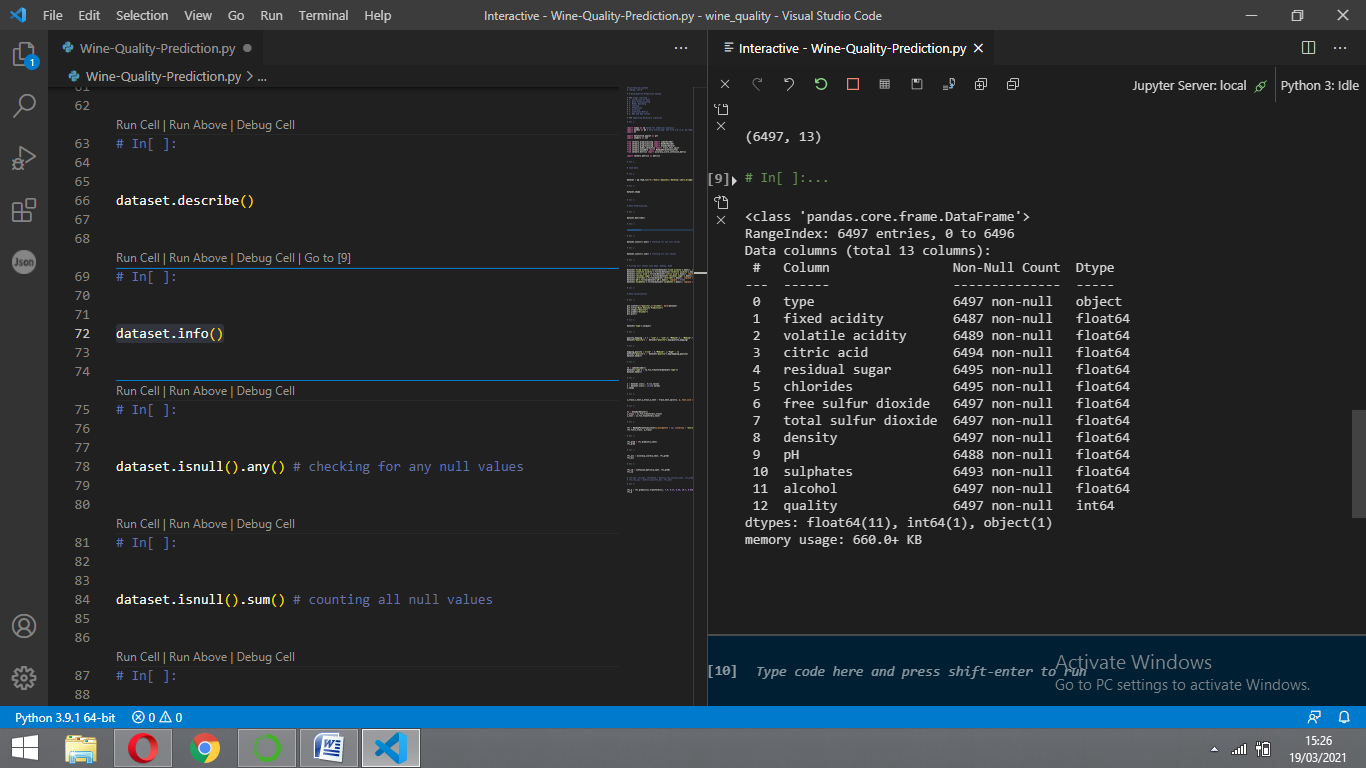
dataset.columns

**shape:** shape function is used to know number of columns and rows in the dataset

dataset.shape

(6497, 13)

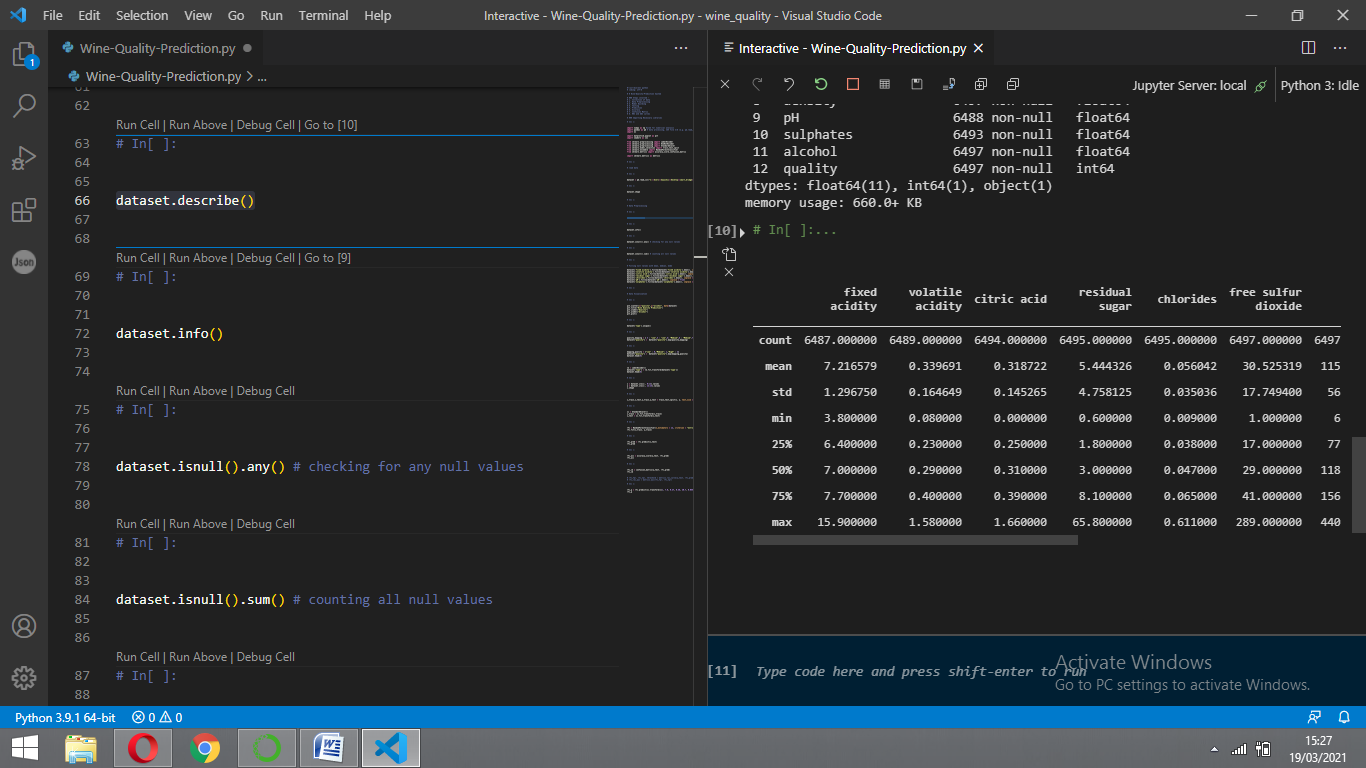
* **Understanding Data Type and Summary of features**
* How the information is stored in a DataFrame or Python object affects what we can do with it and the outputs of calculations as well. There are two main types of data those are numeric and text data types.
* Numeric data types include integers and floats.
* Text data type is known as Strings in Python, or Objects in Pandas. Strings can contain numbers and / or characters.
* For example, a string might be a word, a sentence, or several sentences.
* Will see how our dataset is, by using **info()** method.
* dataset.info()



* We notice that there are both numerical and object data present in the dataset winequality, but it is not necessary that all the continuous data which we are seeing has to be continuous in nature. There may be a case that some categorical data is in the form of numbers but when we perform info() operation we will get numerical output. So, we need to take care of those type of data also.

**describe():** functions are used to compute values like count, mean, standard deviation and IQR(Inter Quantile Ranges) and give a summary of numeric type data.

dataset.describe()



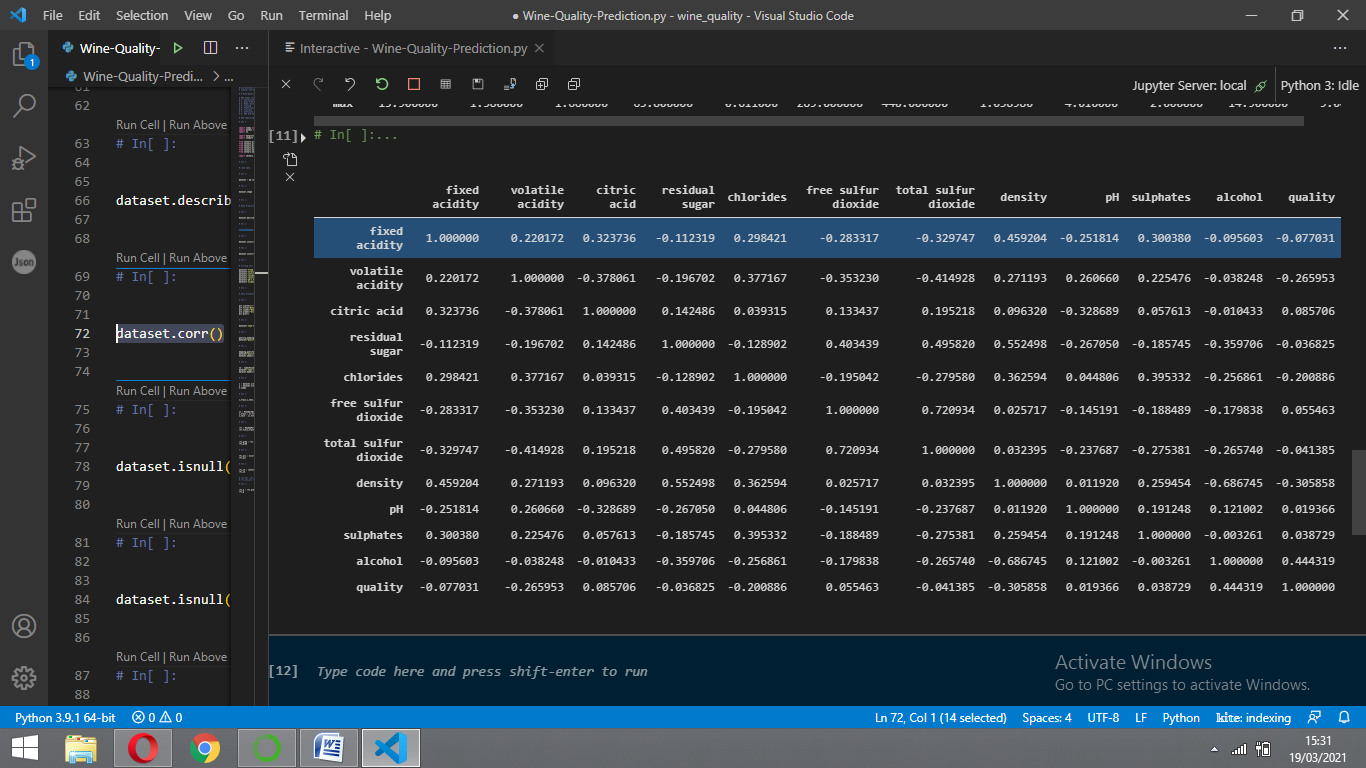
**Activity 4:Converting json objects to strings**

Now let’s convert columns into a format that can be easily read and interpreted. We will convert them into strings and later convert them into lists for easier interpretation.

**Finding the Correlation**

**Correlation** is a statistic that measures the degree to which two variables move in relation to each other.

dataset.corr()

****

Activity 5: Checking for null values

1. After loading it is important to check the complete information of data as it can indication many of the hidden infomation 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 following can be done,

a.Imputing data using Imputation method in sklearn

b.Filling NaN values with mean, median and mode using fillna() method.

We will be using **isnull().any()** method to see which column has missing values.

dataset.isnull().any()

type False

fixed acidity True

volatile acidity True

citric acid True

residual sugar True

chlorides True

free sulfur dioxide False

total sulfur dioxide False

density False

pH True

sulphates True

alcohol False

quality False

dtype: bool

“True” indicates that the particular column has missing values, we can also see the count of null values in each column by using **isnull().sum().**

dataset.isnull().sum()

type 0

fixed acidity 10

volatile acidity 8

citric acid 3

residual sugar 2

chlorides 2

free sulfur dioxide 0

total sulfur dioxide 0

density 0

pH 9

sulphates 4

alcohol 0

quality 0

dtype: int64

As our data is having null values so we have to handle the null values .

# Filling null values with mean, median, mode

dataset['fixed acidity'].fillna(dataset['fixed acidity'].mean(), *inplace* = True)

dataset['volatile acidity'].fillna(dataset['volatile acidity'].mean(), *inplace* = True)

dataset['citric acid'].fillna(dataset['citric acid'].mean(), *inplace* = True)

dataset['residual sugar'].fillna(dataset['residual sugar'].mean(), *inplace* = True)

dataset['chlorides'].fillna(dataset['chlorides'].mean(), *inplace* = True)

dataset['pH'].fillna(dataset['pH'].mean(), *inplace* = True)

dataset['sulphates'].fillna(dataset['sulphates'].mean(), *inplace* = True)

Activity 6: Data Visualization

* 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.

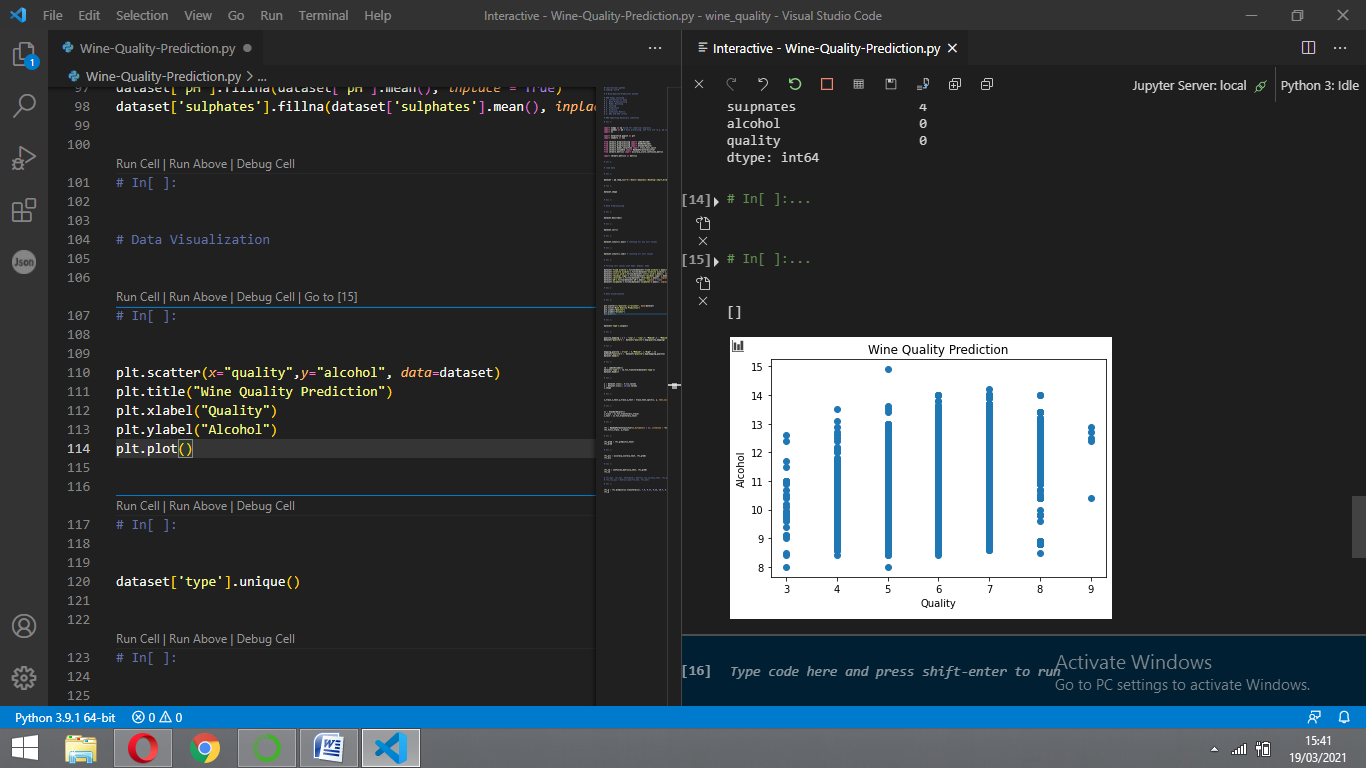
plt.scatter(*x*="quality",*y*="alcohol", *data*=dataset)

plt.title("Wine Quality Prediction")

plt.xlabel("Quality")

plt.ylabel("Alcohol")

plt.plot()



dataset.isnull().sum()

type 0

fixed acidity 0

volatile acidity 0

citric acid 0

residual sugar 0

chlorides 0

free sulfur dioxide 0

total sulfur dioxide 0

density 0

pH 0

sulphates 0

alcohol 0

quality 0

dtype: int64

We can see that now our data is free from null values.

Activity 8: 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.

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

dataset['type'].unique()

array(['white', 'red'], dtype=object)

dataset['quality'].unique()

array([6, 5, 7, 8, 4, 3, 9], dtype=int64)

quality\_mapping = { 3 : "Low",4 : "Low",5: "Medium",6 : "Medium",7: "Medium",8 : "High",9 : "High"}

dataset["quality"] =  dataset["quality"].map(quality\_mapping)

mapping\_quality = {"Low" : 0,"Medium": 1,"High" : 2}

dataset["quality"] =  dataset["quality"].map(mapping\_quality)

dataset.head(5)

le = LabelEncoder()

dataset['type'] = le.fit\_transform(dataset['type'])

**Activity 9: Splitting the Dataset into Dependent and Independent variable.**

* In machine learning, the concept of dependent variable (y) and independent variables(x) is important to understand. Here, Dependent variable is nothing but output in dataset and independent variable is all inputs in the dataset.
* With this in mind, we need to split our dataset into the matrix of independent variables and the vector or dependent variable. Mathematically, Vector is defined as a matrix that has just one column.

To read the columns, we will use **iloc** of pandas (used to fix the indexes for selection) which takes two parameters — [row selection, column selection].

* Let’s split our dataset into independent and dependent variables.

1. The independent variable in the dataset would be considered as 'x' and the ‘type, 'fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol' columns would be considered as independent variables.

2. The dependent variable in the dataset would be considered as 'y' and the quality column is considered as dependent variable.

Now we will split the data of independent variables,

**Independent\_variables:**

x = dataset.iloc[:, 0:12]

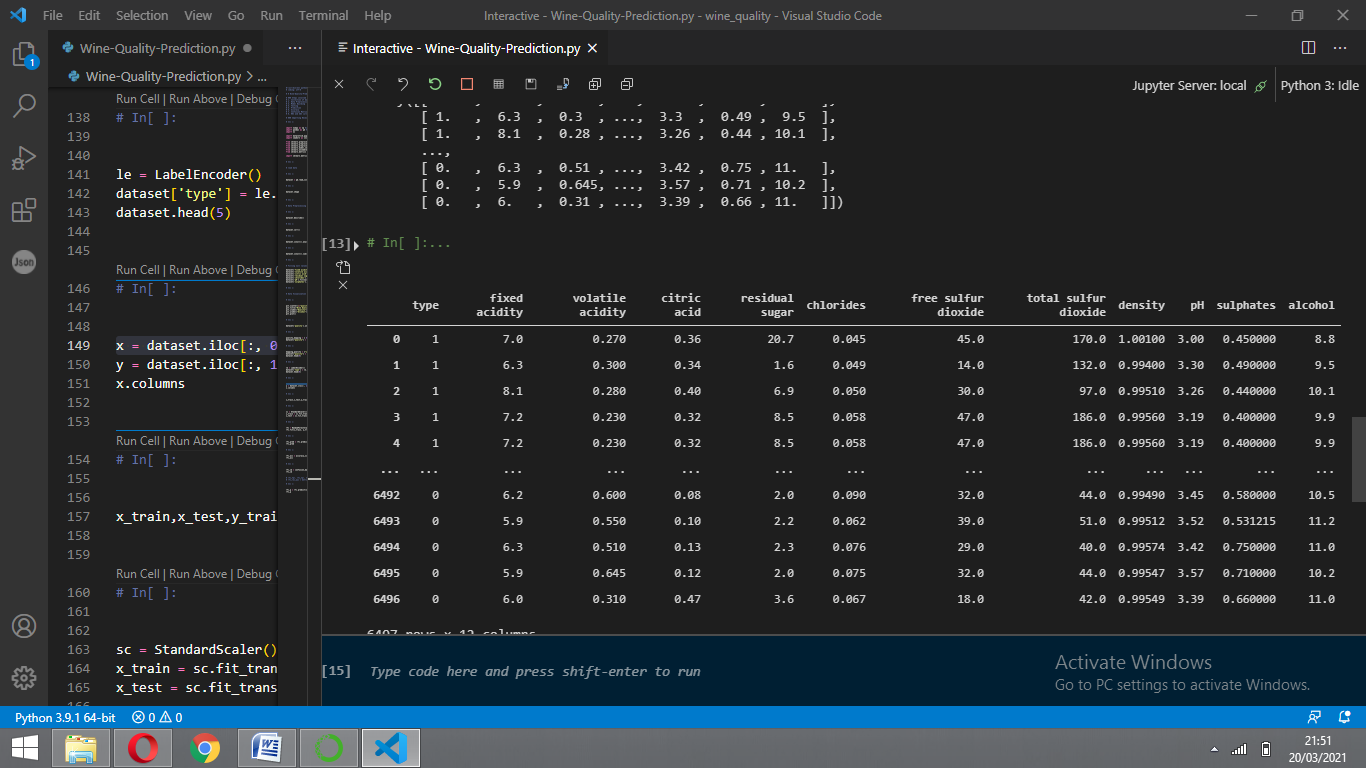
Index(['type', 'fixed acidity', 'volatile acidity', 'citric acid',

'residual sugar', 'chlorides', 'free sulfur dioxide',

'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol'],

dtype='object')

The input features selected are



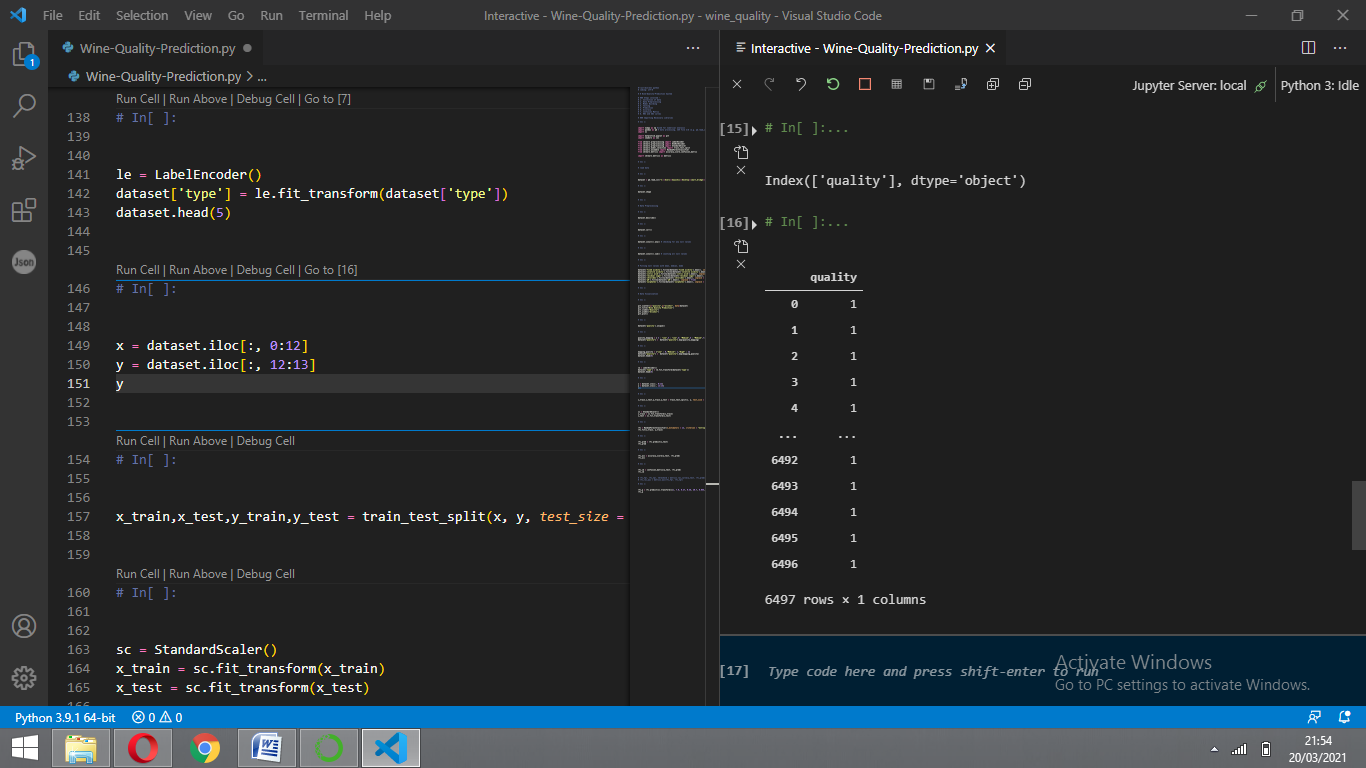
**Dependent\_variables:**

y = dataset.iloc[:, 12:13]

Index(['quality'], dtype='object')

In the above code we are creating DataFrame of the independent variable **x** with our selected columns and for dependent variable **y** we are only taking the **quality** column.

The output feature selected is



**Activity 10: Feature Scaling**

Feature scaling is a method used to normalize the range of independent variables or features of data.

Many machine learning algorithms perform better when numerical input features are scaled to a standard range.

There are two types of scaling..They are

1.Standard Scaling: Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1.

2. Min-Max Scaling: Min-Max is rescaling of the data from the original range so that all values are within the new range of 0 and 1.

Let us scale our data to improve performance. Create an object sc as instance of the class Standard scalar and fit the same to our data frame

sc = StandardScaler()

x\_train = sc.fit\_transform(x\_train)

x\_test = sc.fit\_transform(x\_test)

**Activity 11: Split the dataset into Train set and Test set**

* 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" \t "_blank).’** 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.

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x, y, *test\_size* = 0.2, *random\_state* = 0)

**Milestone 3: Model Building**

Predictive modeling is a mathematical approach to create a statistical model to forecast future behavior based on input test data.

**Steps involved in predictive modeling:**

**Algorithm Selection:**

When we have the structured dataset, and we want to estimate the continuous or categorical outcome then we use supervised machine learning methodologies like regression and classification techniques. When we have unstructured data and want to predict the clusters of items to which a particular input test sample belongs, we use unsupervised algorithms. An actual data scientist applies multiple algorithms to get a more accurate model.

**Train Model:**

After assigning the algorithm and getting the data handy, we train our model using the input data applying the preferred algorithm. It is an action to determine the correspondence between independent variables, and the prediction targets.

**Model Prediction:**

We make predictions by giving the input test data to the trained model. We measure the accuracy by using a cross-validation strategy or ROC curve which performs well to derive model output for test data.

Model building includes the following main tasks

1. Training and testing the model
2. Evaluation of Model
3. Save the model
4. Predicting the output using the model

**Activity 1: Train and Test the Model using Linear Regression.**

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. Multi 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 Random Forest Regression / Classification** **algorithm on our dataset.**

**Random Forest Regression / Classification** is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

**Build the model**

We’re going to use x\_train and y\_train obtained above in train\_test\_split section to train our regression model. We’re using the fit method and passing the parameters as shown below.

rfc = RandomForestClassifier(*n\_estimators* = 12, *criterion* = "entropy", *random\_state* = 0)

rfc.fit(x\_train, y\_train)

rfc\_pred = rfc.predict(x\_test)

# Activity 2: Model evaluation

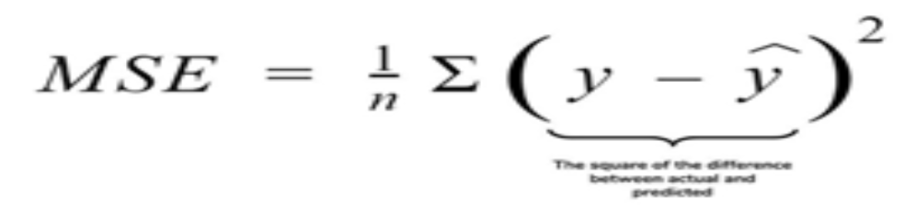
Finally, we need to check to see how well our model is performing on the test data.

**Regression Evaluation Metrics:**

### Mean Squared Error (MSE):

MSE or Mean Squared Error is one of the most preferred metrics for regression problems. It is simply the average of the squared difference between the target value and the value predicted by the regression model.

As it squares the differences, it penalizes even a small error which leads to over-estimation of how bad the model is. It is preferred more than other metrics because it is differentiable and hence can be optimized better.



### RMSE:Root Mean Square Error

### RMSE is the square root of the averaged squared difference between the target value and the value predicted by the model. It is preferred more in some cases because the errors are first squared before averaging which poses a high penalty on large errors. This implies that RMSE is useful when large errors are undesired.

### 

### R2 Score

### Coefficient of Determination or R² is another metric used for evaluating the performance of a regression model. The metric helps us to compare our current model with a constant baseline and tells us how much our model is better.

### The constant baseline is chosen by taking the mean of the data and drawing a line at the mean. R² is a scale-free score that implies it doesn't matter whether the values are too large or too small, the R² will always be less than or equal to 1.



For testing the model we use the below method,

rfc\_acc = accuracy\_score(y\_test, rfc\_pred)

rfc\_acc

0.9361538461538461

**Activity 3: Save the Model**

After building the model we have to save the model.

Pickle is used for serializing and de-serializing Python object structures, also called marshalling or flattening. Serialization refers to the process of converting an object in memory to a byte stream that can be stored on disk or sent over a network. Later on, this character stream can then be retrieved and de-serialized back to a Python object.

This is done by the below code

import pickle

pickle.dump(rfc, open('C:\\Users\\Aayushii\\Desktop\\smart\_bridge\\wine\_quality\\wine\_quality\_model.pkl','wb'))

**Activity 4: Predicting the ouput using the model**

Let us predict the quality by giving input to the model build.

First,load the saved model and standard scalar pickle files .The transformed input is given to model to predict the output.

rfc\_pred = rfc.predict(x\_test)

0.9361538461538461

**Milestone 4: Application Building**

Application Building involves following steps

* 1. Create an HTML file
  2. Build a Python Code
  3. Run the app

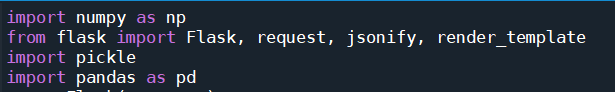
**Activity 1: Create an HTML File**

* We use HTML to create the front end part of the web page.
* Here, we created 2 html pages- Demo2.html, resultnew.html.
* Demo2.html displays home page which accepts the values from the user
* resultnew.html 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)
* We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML pages.
  + **Link :**<https://www.w3schools.com/css/>
  + <https://www.w3schools.com/js/DEFAULT.asp>

**Activity 2: Build python code**

* Let us build flask file ‘app.py’ 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**



Libraries required for the app to run are to be imported.

**Creating our flask app and loading the model**

Now after all the libraries are import we will be creating our flask app and the load our model into our flask app.

model = pickle.load(open("wine\_quality\_model.pkl", "rb"))

app = Flask(\_\_name\_\_) #our flask app

**Routing to the html Page:**

@app.route is used to route the application where it should route to.

Rendering the template. This helps to redirect to index page. In this index page ,we give our input and ask the model to predict

@app.route('/') #default route

*def* hello\_world():

  return render\_template("index.html")

Firstly, we are rendering the index.html template and from there we are navigating to our prediction page that is index.html

@app.route('/login', *methods* = ['POST']) #Main page route

*def* admin():

  p = request.form['type']

  if p =="White":

    p = 0

  elif p == "Red":

    p = 1

  q = *float*(request.form['fa'])

  r = *float*(request.form['va'])

  s = *float*(request.form['ca'])

  t = *float*(request.form['rs'])

  u = *float*(request.form['chl'])

  v = *float*(request.form['fsd'])

  w = *float*(request.form['tsd'])

  x = *float*(request.form['d'])

  y = *float*(request.form['ph'])

  z = *float*(request.form['sp'])

  a = *float*(request.form['ah'])

  sample = [[p, q, r, s, t, u, v, w, x, y, z, a]]

  test = model.predict(sample)

  if test == 0:

    test = "Low"

  elif test == 1:

    test = "Medium"

  elif test == 2:

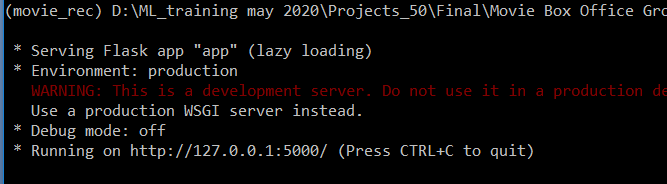
    test = "High"

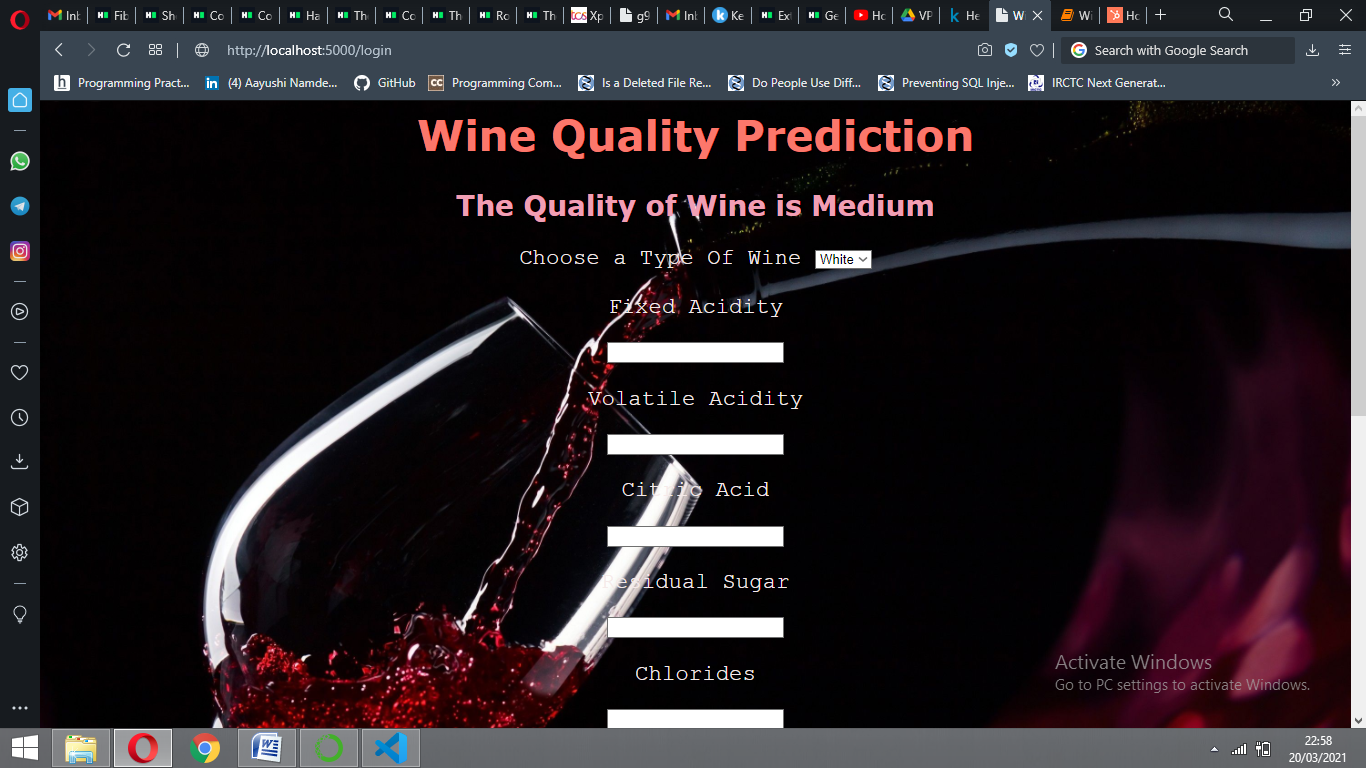
  return render\_template("index.html", *test* = "The Quality of Wine is " + test)

**Activity 3: Run the App**

* + Open new 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
  + Showcasing the output on UI

(base) C:\Users\Aayushii\Desktop\smart\_bridge\wine\_quality>python app.py





This is the prediction page where we get to choose the input from our local system and predict the output.

Finally, the prediction for the given input features is shown.