

FINAL ASSIGNMENT

PREDICTION OF WINE QUALITY USING

RANDOM FOREST MODEL

Muhammad Bilal

S1093347

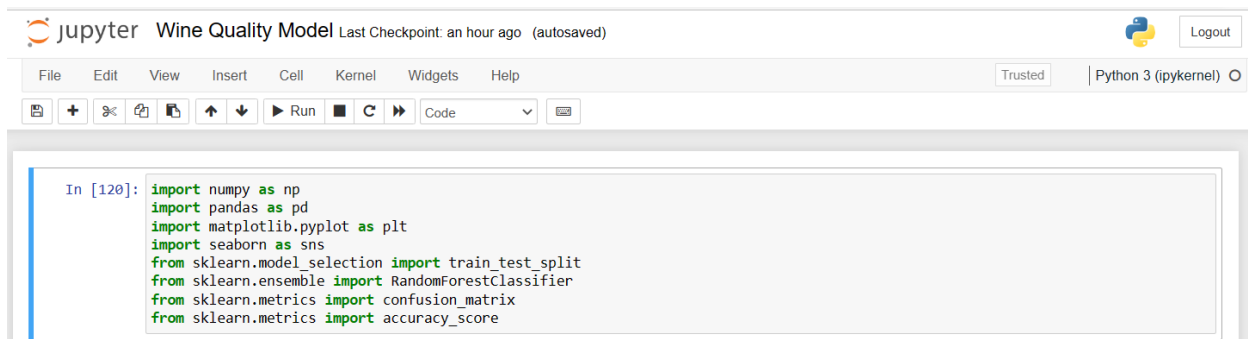
Introduction

In this project I develop a machine learning model that will predict the quality of wine in terms of good or bad for both red and white wine using the given chemical properties of wines that are acidity, citric acid, sugar content and others. In this project I downloaded the label dataset that contains chemical properties of wine from UCI machine learning repository. After getting the data I have done data analysis on this dataset and found out which chemical properties of wine correlate with the quality means if citric acid increase does the quality of wine goes up and down. Then after analyses I have done data preprocessing that means we cannot put raw data to machine learning model so that why data needs to be processed to become compatible for model. Further splitting of dataset into training and testing then train random forest machine learning model based on training dataset and fit that model on test dataset to predict the quality of wine. I have also done an accuracy assessment of my machine learning model in that project by creating confusion matrix and classification report of the model.

Model Development

Importing Dependencies

In this phase I have imported NumPy library that is useful for mathematical calculations and NumPy arrays. Pandas are useful for analyzing and processing data. Matplotlib library for creating static, animated, and interactive visualizations. Seaborn is a data visualization library based on matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics. Finally, Scikit libraries used to split the dataset in training and testing and importing random forest model that will be used for predicting quality at the end.



```
In [120]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

Figure 1

Data Analysis

In this part I import the red and white wine data set from local drive using panda data frame and then merge both dataset into single one and also create a column name type that include the category of wine (red or white). I converted the value and replaced red with 1 and white with 0 because the model does not understand string value as shown in figure 2.

```
In [121]: #wine_data = pd.read_csv(r'E:\Erasmus CDE Courses\Introduction Data Science and Machine Learning\Assignment Final\winequality
red_wine = pd.read_csv(r'E:\Erasmus CDE Courses\Introduction Data Science and Machine Learning\Assignment Final\winequality-r
white_wine = pd.read_csv(r'E:\Erasmus CDE Courses\Introduction Data Science and Machine Learning\Assignment Final\winequality

red_wine['type'] = 1
white_wine['type'] = 0

# Combine the datasets
wine_data = pd.concat([red_wine, white_wine])
```

Figure2

In this code I used shape() function to identify rows and columns in our dataset and head() function is used to print the first 5 rows of the dataset to check that data is in standard format. Then isNull() function is used to check the null values in our dataset as figure 3 shows zero null values in our data

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [122]: wine_data.shape
Out[122]: (6497, 13)
```

```
In [123]: wine_data.head()
Out[123]:
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	type
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	1
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	1
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	1
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	1
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	1

```
In [124]: wine_data.isnull().sum()
Out[124]:
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
type              0
dtype: int64
```

Figure 3

Describe() function is used to find statistical function in our dataset like mean, standard deviation, min etc. as shown in figure 4 below

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [125]: wine_data.describe()
Out[125]:
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	c
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.0
mean	7.215307	0.339666	0.318633	5.443235	0.056034	30.525319	115.744574	0.994697	3.218501	0.531268	10.491801	5.8
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400	56.521855	0.002999	0.160787	0.148806	1.192712	0.8
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000	8.000000	3.0
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.430000	9.500000	5.0
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000	10.300000	6.0
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.600000	11.300000	6.0
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000	14.900000	9.0

In figure 5 below I use seaborn library to draw bar graph that distinct quantity values in our dataset to check how many wines are above 7 quality standards. The value above seven show good quality of wine and vice versa

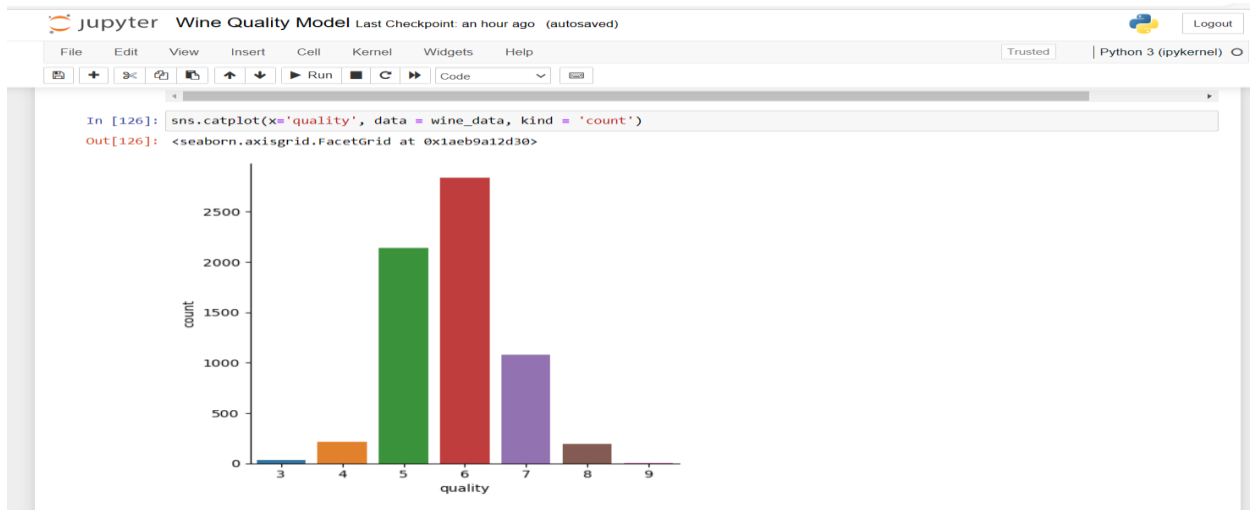


Figure 5

In figure 6 below I compare one of the chemical property name citric acids against quality to check how they correlated with each other as shown below citric acid increases, so the quality of wine also increases.

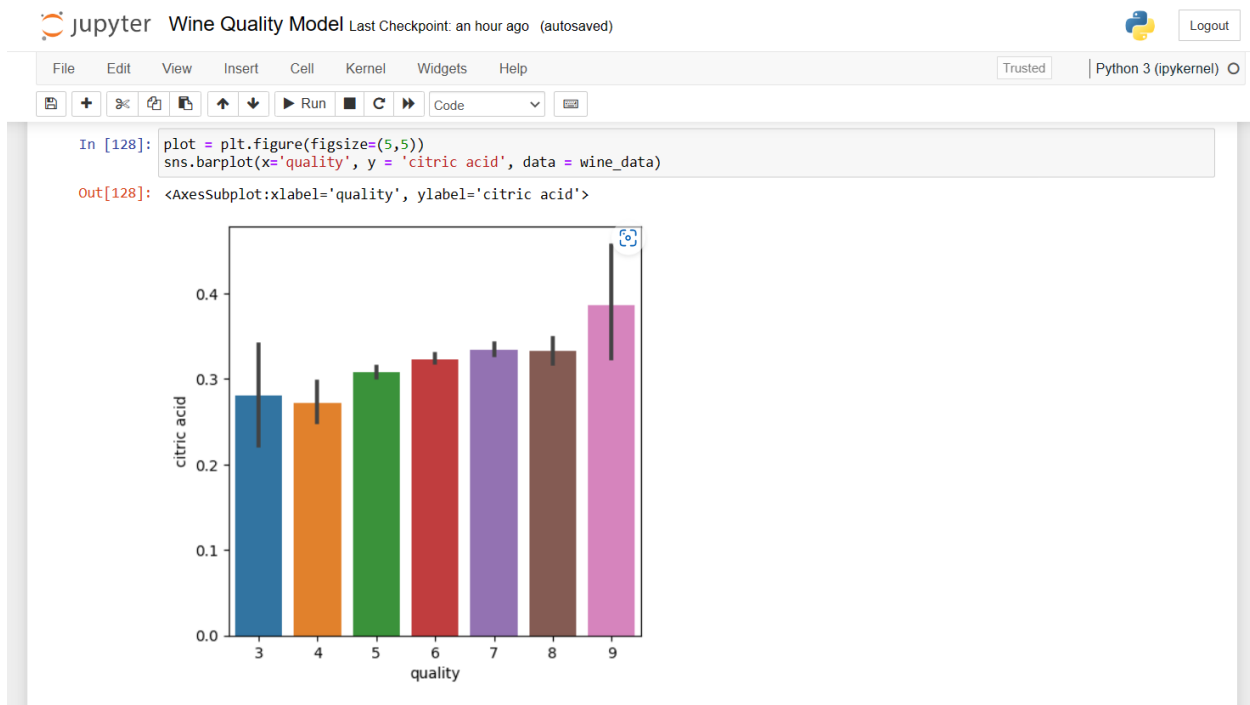


Figure 6

In figure 7 below I generated a correlation matrix between chemical properties of wine that shows positive as well as negative correlation. In this matrix I compare quality column with all other chemical properties of wine to check their correlation. In the last column of matrix, the dark blue cell shows that those chemical properties strongly correlated because with increase of those properties the quality also increase, and light blue cell shows less correlation. So, from this matrix we can identify features that affect the quality.

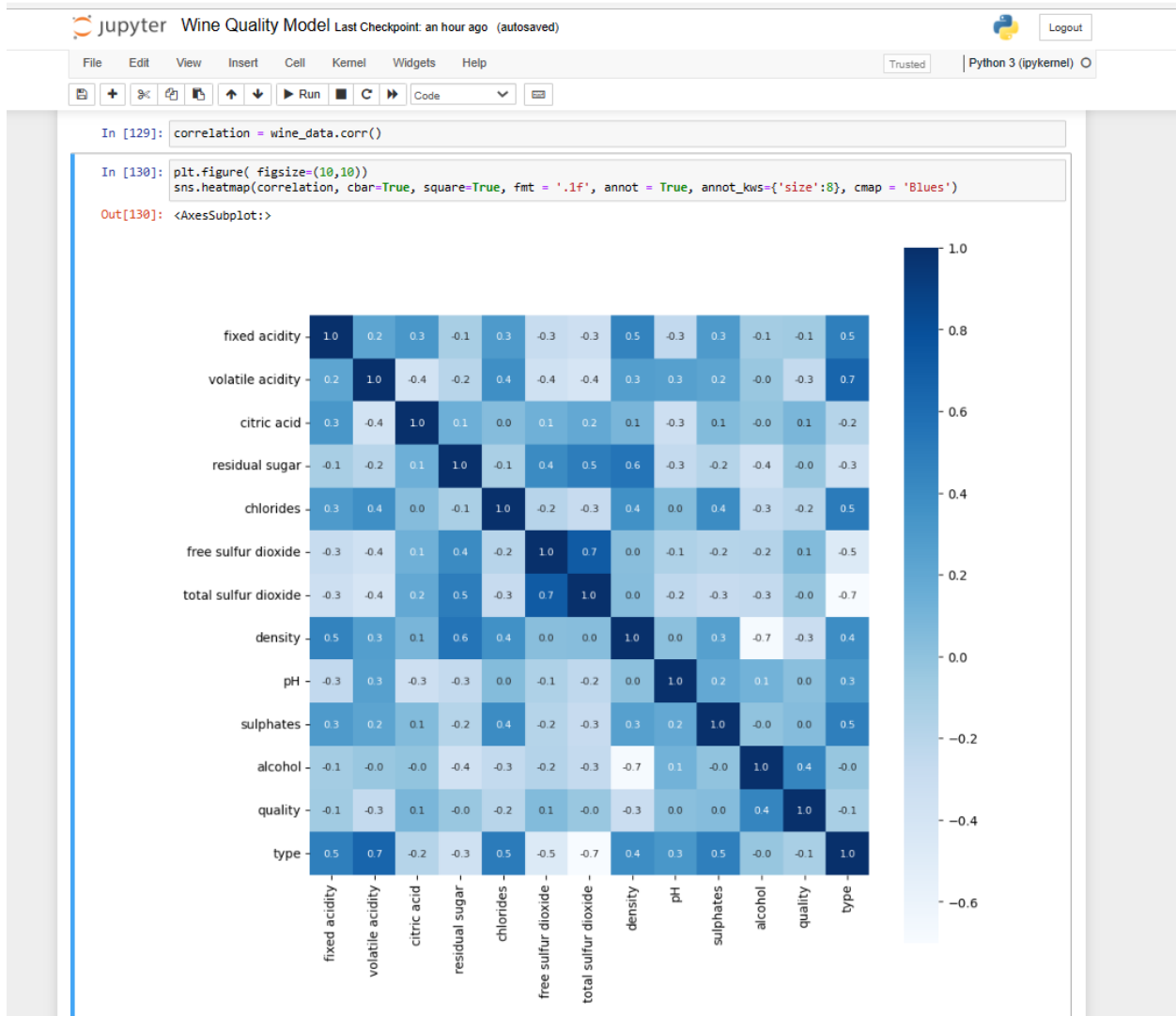


Figure 7

Data Preprocessing

In this I separated the data and label means quality column and store all other columns in X variable to feed this data separately to our machine learning model as shown in figure 8. X variable represents the input variables, which are the independent variables that will be used to predict the output variable. In this case, X consists of all columns in the 'wine_data' data frame except for the 'quality' column.

```

jupyter Wine Quality Model Last Checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipykernel)

In [131]: X = wine_data.drop('quality',axis=1)
In [132]: print(X)

```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides
0	7.4	0.70	0.00	1.9	0.076
1	7.8	0.88	0.00	2.6	0.098
2	7.8	0.76	0.04	2.3	0.092
3	11.2	0.28	0.56	1.9	0.075
4	7.4	0.70	0.00	1.9	0.076
...
4893	6.2	0.21	0.29	1.6	0.090
4894	6.6	0.32	0.36	8.0	0.047
4895	6.5	0.24	0.19	1.2	0.041
4896	5.5	0.29	0.30	1.1	0.022
4897	6.0	0.21	0.38	0.8	0.020
...
0	11.0	34.0	0.99780	3.51	0.56
1	25.0	67.0	0.99680	3.20	0.68
2	15.0	54.0	0.99700	3.26	0.65
3	17.0	60.0	0.99800	3.16	0.58
4	11.0	34.0	0.99780	3.51	0.56
...
4893	24.0	92.0	0.99114	3.27	0.50
4894	57.0	168.0	0.99490	3.15	0.46
4895	30.0	111.0	0.99254	2.99	0.46
4896	20.0	110.0	0.98869	3.34	0.38
4897	22.0	98.0	0.98941	3.26	0.32
...
0	9.4	1			
1	9.8	1			
2	9.8	1			
3	9.8	1			
4	9.4	1			
...			
4893	11.2	0			
4894	9.6	0			
4895	9.4	0			
4896	12.8	0			
4897	11.8	0			
...			

```

[6497 rows x 12 columns]

```

Figure 8

In figure 9 I binarized the values of quality column because I want my labels to be good and bad. The values greater than 7 converted to 1 means good quality and values less than 7 labeled as 0 means bad quality. This label data then stores in separate variable called Y as shown below. Y variable represents the output variable or the dependent variable, which is the column that we want to predict.

```

jupyter Wine Quality Model Last Checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipykernel)

In [133]: Y = wine_data['quality'].apply(lambda y_value:1 if y_value>=7 else 0)
In [134]: print(Y)

```

```

0      0
1      0
2      0
3      0
4      0
..
4893   0
4894   0
4895   0
4896   1
4897   0
Name: quality, Length: 6497, dtype: int64

```

Figure 9

Data splitting

In this part I split the dataset into training and testing with the ratio of 80 and 20 percent. On training dataset our model will be trained and after that model will apply on test dataset to check the accuracy of the model. I used `train_test_split` function to separate data and set 20 percent ratio for test dataset as shown in figure 10.

```
In [135]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=3)

In [136]: print(Y.shape, Y_train.shape, Y_test.shape)
(6497,) (5197,) (1300,)
```

Figure 10

Model Training

I applied random forest model to train on training dataset. Random forests build multiple decision trees using a random subset of features and a random subset of the training data. Then, it averages the predictions from all the decision trees to make the final prediction. This reduces the risk of overfitting and increases the accuracy of the model. Random forest is also capable of handling missing values and maintaining high accuracy even with imbalanced dataset. For training of model, I use fit function that fit the data points to this random forest classifier and give parameter of X and Y train.

After training we need to find model accuracy and apply that model on our test dataset. The confusion matrix has been created for test label against predicted values. The confusion matrix is a 2x2 matrix, where the rows represent the actual values of the target variable ('Y_test') and the columns represent the predicted values ('X_test_prediction').

The matrix shows that there were 1017 true negative predictions (TN) and 164 true positive predictions (TP), which means that the model correctly predicted that 1017 wines were of bad quality and 164 wines were of good quality.

There were also 28 false positive predictions (FP) and 91 false negative predictions (FN). False positives occur when the model predicts a wine to be of good quality, but it is actually of bad quality. False negatives occur when the model predicts a wine to be of bad quality, but it is actually of good quality.

Then I used accuracy score function to check the accuracy of model in percentage by comparing original label dataset with values predicted by our model. The accuracy value of my model is 90 percent that means out 100 its predicted 90 percent of values correctly as shown in figure 11.

```
In [137]: model = RandomForestClassifier()

In [138]: model.fit(X_train, Y_train)

Out[138]: RandomForestClassifier()

In [139]: X_test_prediction = model.predict(X_test)
print(confusion_matrix(Y_test, X_test_prediction))

#test_data_accuracy = accuracy_score(X_test_prediction, Y_test)

[[1017  28]
 [  91 164]]

In [140]: test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy:', test_data_accuracy)

Accuracy: 0.9084615384615384
```

Figure 11

Confusion Matrix

In this case, the classification report shows the performance of the model on the two classes of the target variable ('0' and '1'), where '0' represents bad quality wine and '1' represents good quality wine.

The report shows that the model achieved a precision of 0.92 and a recall of 0.97 for the '0' class, which means that out of all the instances predicted as '0', 92% were actually '0', and out of all the actual '0' instances, 97% were predicted as '0'. The F1 score is 0.94 for the '0' class.

For the '1' class, the model achieved a precision of 0.85 and a recall of 0.64, which means that out of all the instances predicted as '1', 85% were actually '1', and out of all the actual '1' instances, 64% were predicted as '1'. The F1 score is 0.73 for the '1' class.

The accuracy of the model on the test data is 0.91, which means that the model correctly predicted the quality of 91% of the wines in the test data.

```
Accuracy: 0.9084615384615384

In [153]: matrix = classification_report(Y_test, X_test_prediction)
          print (matrix)

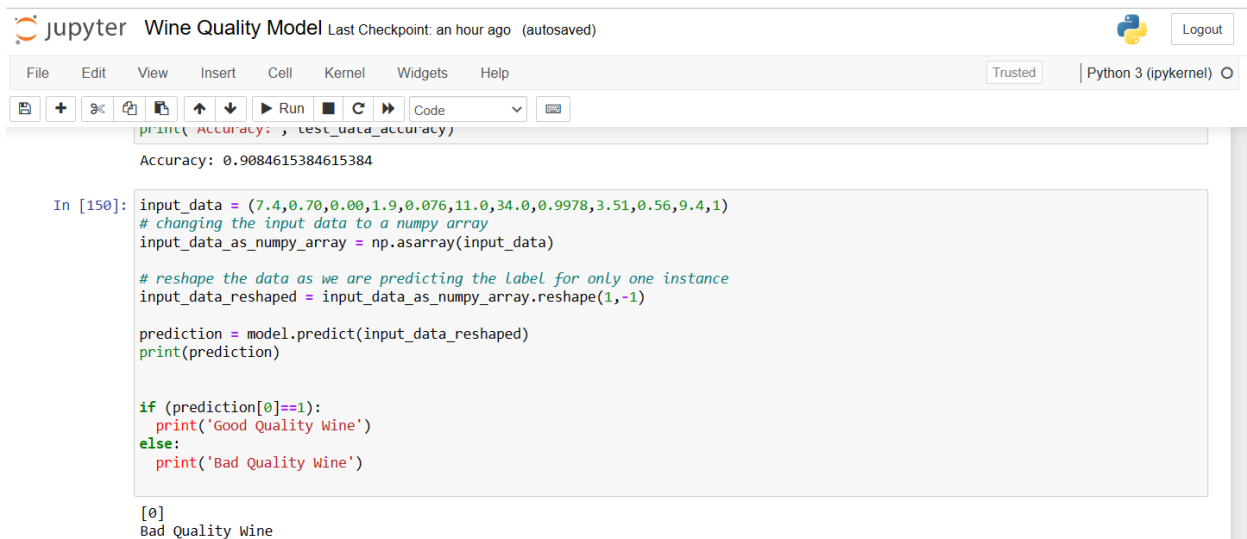
              precision    recall  f1-score   support

     0           0.92       0.97      0.94       1045
     1           0.85       0.64      0.73        255

 accuracy          0.91       0.91      0.91       1300
 macro avg          0.89       0.81      0.84       1300
 weighted avg          0.91       0.91      0.90       1300
```

Model Predictive System

In this I manually gave all the chemical parameters by myself except quality value then this model will find the quality of wine and give output in term of good or bad quality of wine.



The screenshot shows a Jupyter Notebook titled "Wine Quality Model" with a last checkpoint of "an hour ago (autosaved)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a status bar indicating "Trusted" and "Python 3 (ipykernel)".

```
print(accuracy, test_data_accuracy)

Accuracy: 0.9084615384615384

In [150]: input_data = (7.4,0.70,0.00,1.9,0.076,11.0,34.0,0.9978,3.51,0.56,9.4,1)
          # changing the input data to a numpy array
          input_data_as_numpy_array = np.asarray(input_data)

          # reshape the data as we are predicting the label for only one instance
          input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

          prediction = model.predict(input_data_reshaped)
          print(prediction)

          if (prediction[0]==1):
              print('Good Quality Wine')
          else:
              print('Bad Quality Wine')

[0]
Bad Quality Wine
```

Figure 12