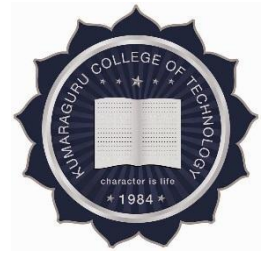




**EXPLORATORY DATA ANALYSIS AND CASE
PREDICTION ON
WINE QUALITY DATASET**



ENGINEERING CLINIC PROJECT REPORT

Submitted by

1.S.MIRUTHUVIKASINI	18BCS021
2.N.ABINAYASRI	18BCS033
3.R.PRAGATHI	18BCS038
4.N.JENFERO	18BCS043
5. S.RAGHAVI	18BCS056

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE-641 049

(An Autonomous Institution Affiliated to Anna University, Chennai)

December 2020

Verified by

(V. Sudha)

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
1.	ABSTRACT	3
	INTRODUCTION	3
	1.1 CONCEPTUAL STUDY OF THE PROJECT	3
	1.2 OBJECTIVE OF THE PROJECT	5
	1.3 SCOPE OF THE PROJECT	5
2.	LITERATURE REVIEW	
	2.1 LITERATURE REVIEW OF JOURNALS	6
3.	PROBLEM DEFINITION	
		11
4.	LOADING THE DATASET	
	4.1 BASIC DATA EXPLORATION	12
	4.2 DATA CLEANING	17
	4.2.1. CHECKING FOR NULL VALUE	17
	4.2.2.CHECKING FOR OUTLIERS	18
	4.2.3.TREATING THE OUTLIER	20
	4.3 DATA VISUALIZATION	21
	4.3.1.HISTOGRAM	22
	4.3.2.STRIPPLOT	23
	4.3.3.COUNTPLOT	24
	4.4 NORMALIZATION	25
	4.5 PREDICTION OF TARGET VARIABLE	27
5.	CONCLUSION	29
6.	REFERENCE LINK	29

ABSTRACT

Machine learning has made dramatic improvements in the past few years, Here we applied various machine learning techniques to predict the Quality of wine based on various physicochemical data. In our Project Wine Quality-red dataset has been used to analyze and infer various information about the data. We use various Python libraries such as pandas, numpy, matplotlib, seaborn and scikit-learn to validate our dataset. Data Visualization had played a vital role to make data more natural for the human mind to comprehend and therefore makes it easier to identify trends, patterns, and outliers within large data sets. We also use that technique to get clear inference and we use it to remove all our outliers. We had trained our data by using a machine learning algorithm called DecisionTreeClassifier so that it would predict the quality of wine based on the available physicochemical data. In this quality prediction testing is done on the 20 percent of the data and the training is done on the 80 percent of the data. The results we have obtained is about the accuracy

1.INTRODUCTION

1.1 INTRODUCTION

The red wine industry shows a recent exponential growth as social drinking is on the rise. Nowadays, industry players are using product quality certifications to promote their products. This is a time-consuming process and requires the assessment given by human experts, which makes this process very expensive. Another vital factor in red wine certification and quality assessment is physicochemical tests, which are laboratory-based and consider factors like acidity, pH level, sugar, and other chemical properties.

Our analysis will use Red Wine Quality Data Set, available on Kaggle <https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009>

The wine Samples was obtained from the north of Portugal to model red wine quality based on physicochemical tests. The dataset contains a total of 12 variables, which were recorded for 1,599 observations.

Attribute Details:**Input variables (based on physicochemical tests):**

- 1 - fixed acidity
- 2 - volatile acidity
- 3 - citric acid
- 4 - residual sugar
- 5 - chlorides
- 6 - free sulfur dioxide
- 7 - total sulfur dioxide
- 8 - density
- 9 - pH
- 10 - sulphates
- 11 - alcohol

Output variable (based on sensory data):

- 12 - quality (score between 0 and 10)

These days with the advent of machine learning techniques it is possible to classify the wines as well as it is possible to figure out the importance of each chemical analyses parameters in the wine and which one to ignore for reduction of cost. The performance comparison with different feature sets will also help to classify it in a more distinctive way. In this paper machine learning approach is proposed to train the dataset and make a test to predict the Quality of wine given the physicochemical data.

1.2 OBJECTIVES OF THE PROJECT

The main objective of this project is to study the wine quality prediction dataset which is available in kaggle and to explore more on python libraries which helps to do exploratory data analysis.

To analyse the data by using various Data Visualization Techniques.

To Preprocess the data by removing the NULL values and treating the missing values.

To identify and remove the Outliers by using various Python techniques.

Finally Prediction is made on the quality of wine by incorporating various machine learning models.

1.3 SCOPE OF THE PROJECT

Learning the attributes of a dataset and understanding the relationship between them.

Cleaning the data by removing the NULL values and treating Outliers.

Visualizing data to get it more precise, by exploring various python libraries such as Numpy, Pandas and seaborn.

To understand the various Machine Learning algorithms and use them accordingly.

Split the data into Train and Test data and make prediction on the quality of wine test by including alternative models on machine learning.

2. LITERATURE REVIEW

2.1 LITERATURE REVIEW OF JOURNALS

Title of the paper: Wine Quality

Research Focus: Exploratory Data Analysis (EDA) in Python for the analysis of Wine Quality dataset.

Student level: Undergraduate

Abstract:

Wine classification is a difficult task since taste is the least understood of the human senses. A good wine quality prediction can be very useful in the certification phase, since currently the sensory analysis is performed by human tasters, being clearly a subjective approach. An automatic predictive system can be integrated into a decision support system, helping the speed and quality of the performance. Furthermore, a feature selection process can help to analyze the impact of the analytical tests. If it is concluded that several input variables are highly relevant to predict the wine quality, since in the production process some variables can be controlled, this information can be used to improve the wine quality.

From the following research journal papers we have included the details related to our dataset.

Research paper 1:

Selection of important features and predicting wine quality using machine learning techniques

Y Gupta - Procedia Computer Science, 2018 – Elsevier

Nowadays, industries are using product quality certifications to promote their products. This is a time taking process and requires the assessment given by human experts which makes this process very expensive. This paper explores the usage of machine learning techniques such as linear regression, neural network and support vector machine for product quality in two ways. Firstly, determine the dependency of target variable on independent variables and secondly, predicting the value of target variable. In this paper, linear regression is used to determine the dependency of target variable on independent variables. On the basis of computed dependency, important variables are selected those make significant impact on dependent variable. Further, neural network and support vector machine are used to predict the values of dependent variable. All the experiments are performed on Red Wine and White Wine datasets. This paper proves that the better prediction can be made if selected features (variables) are being considered rather than considering all the features.

Proposed methodology:

- Linear regression
- Neural network
- Support vector machine

Experimental results and analysis :

- Determining important features for prediction
- Predicting value of dependent variable (Quality)

Conclusion and future directions:

The interest has been increased in wine industry in recent years which demands growth in this industry. Therefore, companies are investing in new technologies to improve wine production and selling. In this direction, wine quality certification plays a very important role for both processes and it requires wine testing by human experts. This paper explores the usage of machine learning techniques in two ways. Firstly, how linear regression determines important features for prediction. Secondly, the usage of neural network and support vector machine in predicting the values. The benchmark Wine dataset is used for all experiments. This dataset has two parts: Red Wine and White Wine data. Red wine contains 1599 samples and white wine contains 4898 samples. Both red and white wine dataset consists of 12 physicochemical characteristics. One (quality) is dependent variable and other 11 are predictors. The experiments shows that the value of dependent variable can be predicted more accurately if only important features are considered in prediction rather than considering all features. In future, large dataset can be taken for experiments and other machine learning techniques may be explored for wine quality prediction.

Reference

link:

<https://www.sciencedirect.com/science/article/pii/S1877050917328053>

Research paper 2:

Assessing wine quality using a decision tree

S Lee, J Park, K Kang - 2015 IEEE International Symposium on ...,
2015 - ieeexplore.ieee.org

Even though wine-drinkers generally agree that wines may be ranked by quality, wine-tasting is famously subjective. There have been many attempts to construct a more methodical approach to the assessment of wines. We propose a method of assessing wine quality using a decision tree, and test it against the wine-quality dataset from the UC Irvine Machine Learning Repository. Results are 60% in agreement with traditional assessment techniques.

Reference link: <https://ieeexplore.ieee.org/abstract/document/7302752>

Research paper 3:

Prediction of Quality for Different Type of Wine based on
Different Feature Sets Using Supervised Machine Learning
Techniques

S Aich, AA Al-Absi, KL Hui... - 2019 21st International ..., 2019 -
ieeexplore.ieee.org

In recent years, most of the industries promoting their products based on the quality certification they received on the products. The traditional way of assessing the product quality is time consuming, however with the invent of machine learning techniques the processes has become more efficient and

consumed less time than before. In this paper we have explored, some of the machine learning techniques to assess the quality of wine based on the attributes of wine that depends on quality. We have used white wine and red wine quality dataset for this research work. We have used different feature selection technique such as genetic algorithm (GA) based feature selection and simulated annealing (SA) based feature selection to check the prediction performance. We have used different performance measure such as accuracy, sensitivity, specificity, positive predictive value, negative predictive value for comparison using different feature sets and different supervised machine learning techniques. We have used nonlinear, linear and probabilistic classifiers. We have found that feature selection-based feature sets able to provide better prediction than considering all the features for performance prediction. We have found accuracy ranging from 95.23% to 98.81% with different feature sets. This analysis will help the industries to access the quality of the products at less time and more efficient way.

Reference link: <https://ieeexplore.ieee.org/abstract/document/8702017>

Research paper 4:

The classification of wine according to their physicochemical qualities

Y Er, A Atasoy - International Journal of Intelligent Systems and ..., 2016 - ijisae.org

The main purpose of this study is to predict wine quality based on physicochemical data. In this study, two large separate data sets which were

taken from UC Irvine Machine Learning Repository were used. These data sets contain 1599 instances for red wine and 4898 instances for white wine with 11 features of physicochemical data such as alcohol, chlorides, density, total sulfur dioxide, free sulfur dioxide, residual sugar, and pH. First, the instances were successfully classified as red wine and white wine with the accuracy of 99.5229% by using Random Forests Algorithm. Then, the following three different data mining algorithms were used to classify the quality of both red wine and white wine: k-nearest-neighbourhood, random forests and support vector machines. There are 6 quality classes of red wine and 7 quality classes of white wine. The most successful classification was obtained by using Random Forests Algorithm. In this study, it is also observed that the use of principal component analysis in the feature selection increases the success rate of classification in Random Forests Algorithm.

Reference link: <https://www.ijisae.org/IJISAE/article/view/914>

3. Problem Definition:

This data will allow us to create different regression models to determine how different independent variables help predict our dependent variable, quality. Knowing how each variable will impact the wine quality will help producers, distributors, and businesses in the wine industry better assess their production, distribution, and pricing strategy.

4.LOADING THE DATASET

We use pandas to analyse the dataset.

```
import pandas as pd
```

We use numpy for scientific computing.

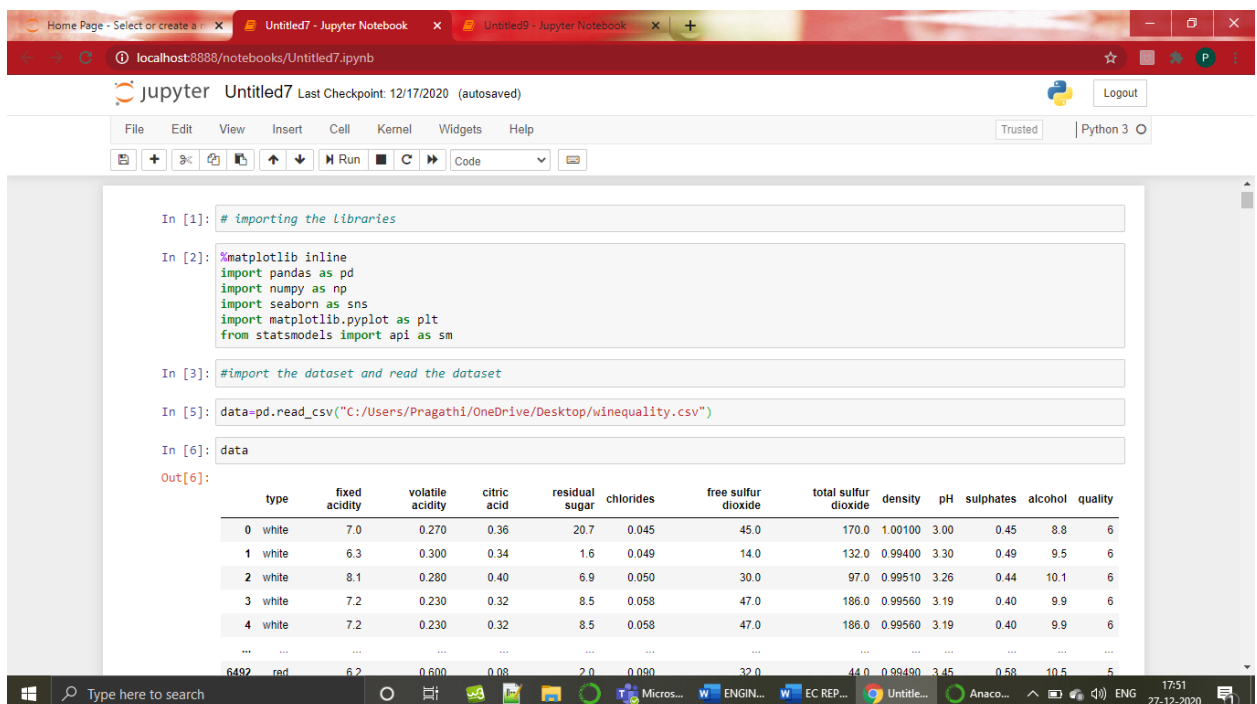
import numpy as np

We use matplotlib for visualization of data generally consists of bars, pies, lines, scatter plots and so on.

import matplotlib.pyplot as plt

We use seaborn for data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

import seaborn as sns



The screenshot shows a Jupyter Notebook window titled 'Untitled7'. The code in the notebook is as follows:

```
In [1]: # importing the libraries

In [2]: %matplotlib inline
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels import api as sm

In [3]: #import the dataset and read the dataset

In [5]: data=pd.read_csv("C:/Users/Pragathi/OneDrive/Desktop/winequality.csv")

In [6]: data
```

The output of the last cell is a preview of the 'data' DataFrame, showing the first 5 rows and the first 13 columns:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.270	0.36	20.7	0.045	45.0	170.0	1.00100	3.00	0.45	8.8	6
1	white	6.3	0.300	0.34	1.6	0.049	14.0	132.0	0.99400	3.30	0.49	9.5	6
2	white	8.1	0.280	0.40	6.9	0.050	30.0	97.0	0.99510	3.26	0.44	10.1	6
3	white	7.2	0.230	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	9.9	6
4	white	7.2	0.230	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	9.9	6

4.1 BASIC DATA EXPLORATION

- Head of the dataset- The head function will display the top records in the data set. By default, python shows you only the top 5 records.

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

```
In [12]: data.head()
```

```
Out[12]:
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6


```
In [13]: data.head(11)
```

```
Out[13]:
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
5	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
6	white	6.2	0.32	0.16	7.00	0.045	30.0	136.0	0.9949	3.18	0.47	9.6	6
7	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
8	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6

- Tail of the dataset- The tail function will display the last records in the data set. By default, python shows you only the last 5 records.

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

```
In [17]: data.tail()
```

```
Out[17]:
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
6492	red	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
6493	red	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	NaN	11.2	6
6494	red	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
6495	red	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
6496	red	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6


```
In [18]: data.tail(11)
```

```
Out[18]:
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
6486	red	7.2	NaN	0.33	2.5	0.068	34.0	102.0	0.99414	3.27	0.78	12.8	6
6487	red	6.6	0.725	0.20	7.8	0.073	29.0	79.0	0.99770	3.29	0.54	9.2	5
6488	red	6.3	0.550	0.15	1.8	0.077	26.0	35.0	0.99314	3.32	0.82	11.6	6
6489	red	5.4	0.740	0.09	1.7	0.089	16.0	26.0	0.99402	3.67	0.56	11.6	6
6490	red	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
6491	red	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.82	9.5	6
6492	red	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
6493	red	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	NaN	11.2	6

- Shape of the dataset- To check the dimension of data.

The screenshot shows a Jupyter Notebook with the following content:

```
6497 rows x 13 columns
```

```
In [15]: data.shape
```

```
Out[15]: (6497, 13)
```

```
In [16]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  -
0   type                   6497 non-null   object
1   fixed acidity          6487 non-null   float64
2   volatile acidity       6489 non-null   float64
3   citric acid            6494 non-null   float64
4   residual sugar         6495 non-null   float64
5   chlorides              6495 non-null   float64
6   free sulfur dioxide    6497 non-null   float64
7   total sulfur dioxide   6497 non-null   float64
8   density                6497 non-null   float64
9   pH                     6488 non-null   float64
10  sulphates              6493 non-null   float64
11  alcohol                6497 non-null   float64
12  quality                6497 non-null   int64
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
```

```
In [12]: data.head()
```

- Info of the dataset- `info()` is used to check the Information about the data and the data types of each respective attribute.

This screenshot is identical to the one above, showing the same Jupyter Notebook content with data shape, info, and head methods.

- Summary of the dataset- The `describe` method will help to see how data has been spread for numerical values.

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

```
In [19]: data.describe()
```

```
Out[19]:
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
count	6487.000000	6489.000000	6494.000000	6495.000000	6495.000000	6497.000000	6497.000000	6497.000000	6488.000000	6493.000000	6497.000000
mean	7.216579	0.339691	0.318722	5.444326	0.056042	30.525319	115.744574	0.994697	3.218395	0.531215	10.491801
std	1.296750	0.164649	0.145265	4.758125	0.035036	17.749400	56.521855	0.002999	0.160748	0.148814	1.192712
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000	8.000000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.430000	9.500000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000	10.300000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.600000	11.300000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000	14.900000

```
In [13]: data.columns
```

```
Out[13]: Index(['type', 'fixed acidity', 'volatile acidity', 'citric acid',
              'residual sugar', 'chlorides', 'free sulfur dioxide',
              'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol',
              'quality'],
              dtype='object')
```

```
In [14]: #to check and display the unique values
```

- Columns of the dataset- The column method will help to see the names of the columns the dataset contains.

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

```
In [13]: data.columns
```

```
Out[13]: Index(['type', 'fixed acidity', 'volatile acidity', 'citric acid',
              'residual sugar', 'chlorides', 'free sulfur dioxide',
              'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol',
              'quality'],
              dtype='object')
```

```
In [14]: #to check and display the unique values
```

```
In [15]: data.nunique()
```

```
Out[15]:
```

type	2
fixed acidity	106
volatile acidity	187
citric acid	89
residual sugar	316
chlorides	214
free sulfur dioxide	135
total sulfur dioxide	276
density	998
pH	108
sulphates	111
alcohol	111
quality	7
dtype:	int64

- Unique values of the dataset- The unique function will help to see the unique values in the specific column of the dataset.

- The nunique function will help to see the no of unique values does the dataset contains.

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

```
In [14]: #to check and display the unique values

In [15]: data.nunique()

Out[15]: type                2
         fixed acidity      106
         volatile acidity    187
         citric acid         89
         residual sugar      316
         chlorides           214
         free sulfur dioxide  135
         total sulfur dioxide 276
         density             998
         pH                  108
         sulphates           111
         alcohol             111
         quality              7
         dtype: int64

In [16]: data['quality'].nunique()

Out[16]: 7

In [17]: data['quality'].unique()

Out[17]: array([6, 5, 7, 8, 4, 3, 9], dtype=int64)

In [18]: data['pH'].nunique()

Out[18]: 108
```

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

```
In [19]: data['pH'].unique()

Out[19]: array([3. , 3.3 , 3.26, 3.19, 3.18, 3.22, 2.99, 3.14, 3.54, 2.98, 3.25,
               3.24, 3.33, 3.12, 3.17, 3.47, 3.05, 3.42, 3.45, 3.38, 3.1 , 3.2 ,
               3.37, 3.13, 3.21, 3.11, 3.16, 3.27, 3.36, 3.35, 3.34, 3.32, 3.31,
               nan, 3.09, 3.03, 3.02, 3.15, 3.69, 2.95, 2.94, 3.39, 3.04, 2.89,
               3.4 , 3.01, 3.63, 3.29, 3.23, 3.28, 3.06, 3.08, 2.87, 2.93, 3.44,
               3.5 , 3.48, 3.72, 3.53, 3.52, 2.96, 3.61, 3.43, 3.49, 3.51, 3.46,
               3.07, 2.97, 3.41, 3.64, 3.56, 2.86, 2.88, 2.85, 3.58, 3.55, 3.66,
               3.59, 2.74, 2.92, 3.82, 3.81, 3.65, 2.9 , 3.77, 2.91, 3.62, 3.74,
               2.8 , 3.6 , 2.72, 2.79, 3.57, 3.8 , 3.68, 2.77, 3.79, 3.7 , 2.84,
               3.76, 2.83, 3.75, 2.82, 3.67, 3.9 , 3.85, 3.71, 3.78, 4.01])

In [20]: #to check for null values and removing that

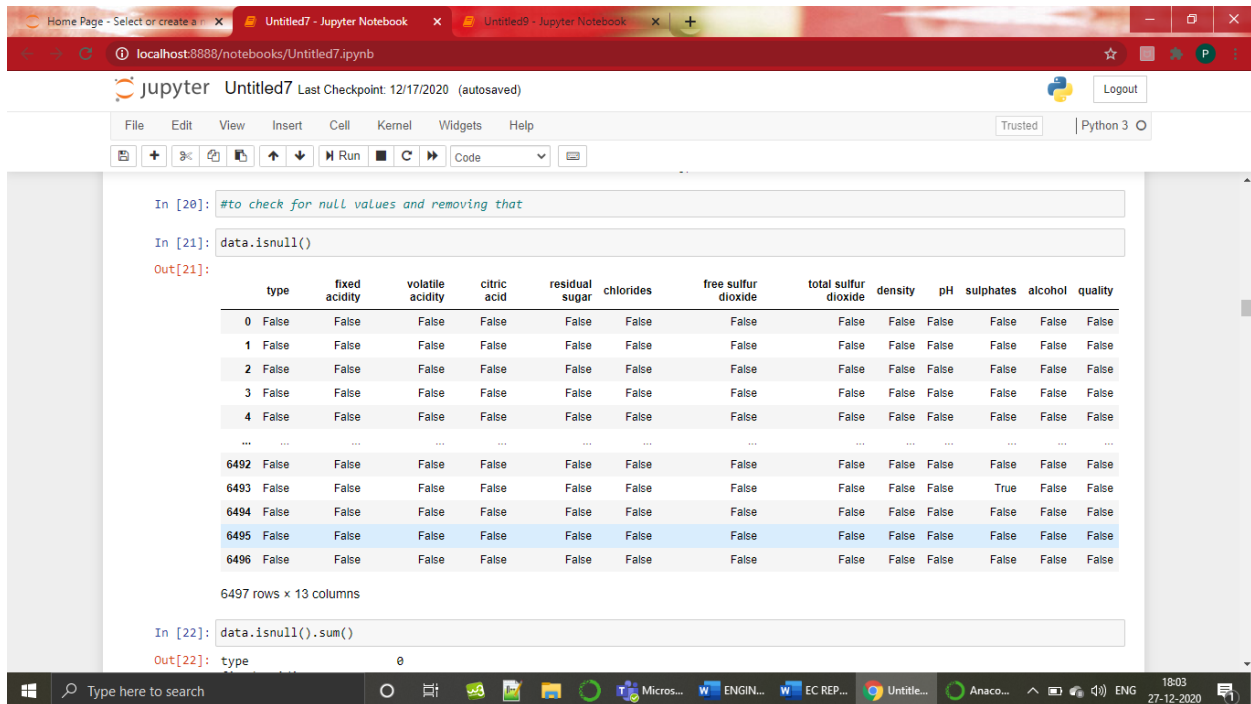
In [21]: data.isnull()

Out[21]:
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False

4.2. DATA CLEANING –

4.2.1 CHECKING FOR NULL VALUES



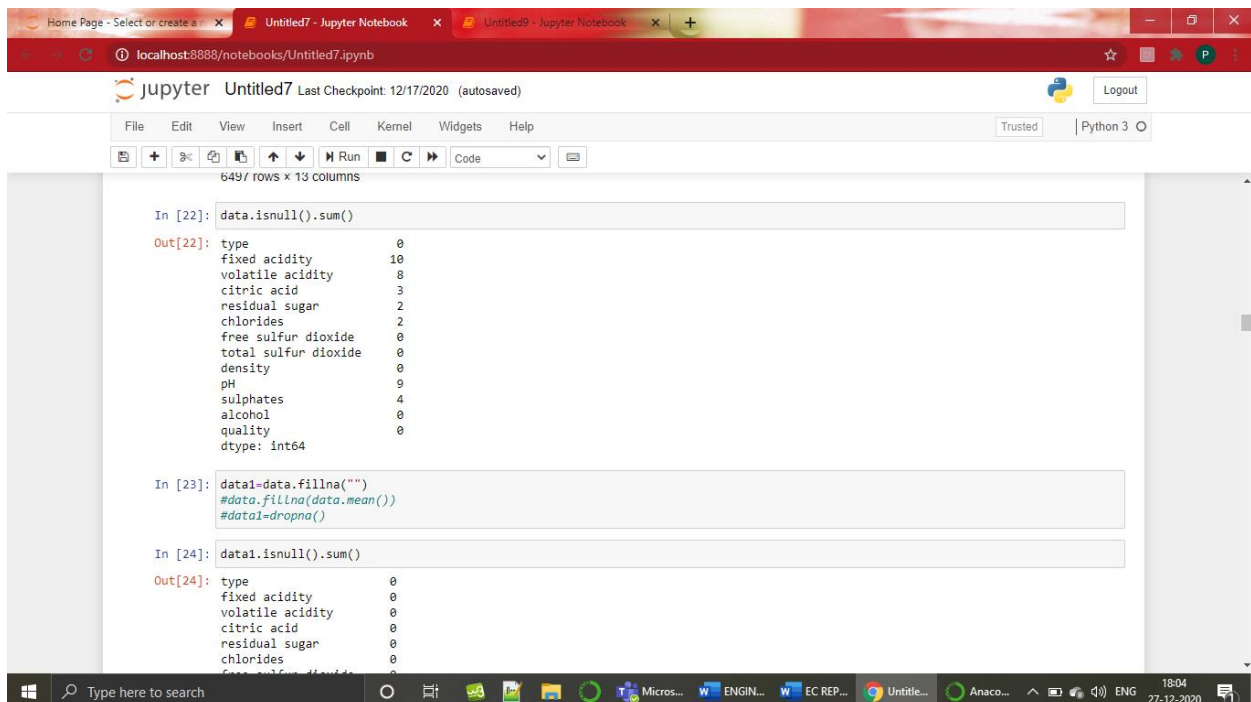
A screenshot of a Jupyter Notebook interface. The browser address bar shows 'localhost:8888/notebooks/Untitled7.ipynb'. The notebook title is 'Untitled7' with a last checkpoint of '12/17/2020 (autosaved)'. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar shows icons for file operations and a 'Run' button. The code cell contains:

```
In [20]: #to check for null values and removing that
In [21]: data.isnull()
```

The output shows a DataFrame with 13 columns: type, fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, and quality. The first few rows are displayed, showing 'False' for all null checks. The output is summarized as '6497 rows x 13 columns'.

```
In [22]: data.isnull().sum()
```

The output shows the sum of null values for each column, all of which are 0.



A screenshot of a Jupyter Notebook interface showing the next steps in handling null values. The code cell contains:

```
In [22]: data.isnull().sum()
Out[22]: 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

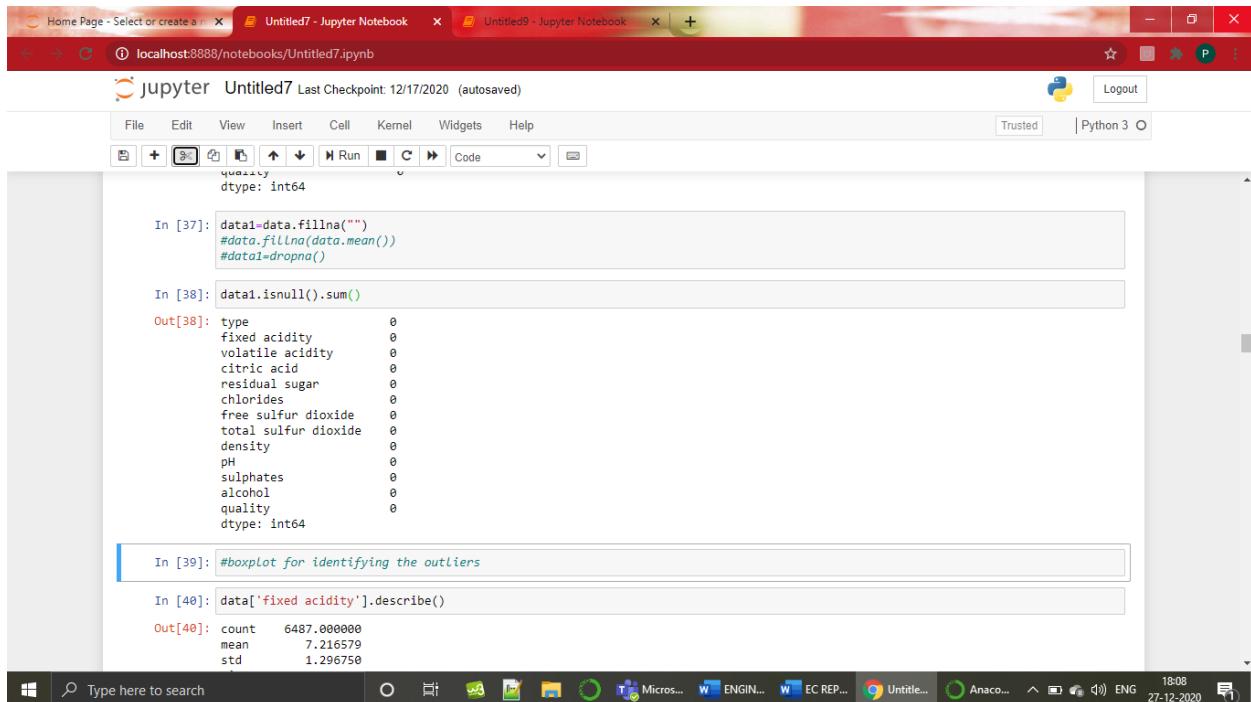
In [23]: data1=data.fillna("")
         #data.fillna(data.mean())
         #data1=dropna()

In [24]: data1.isnull().sum()
```

The output shows the sum of null values for each column after filling them with empty strings. The output is summarized as '6497 rows x 13 columns'.

TREATING THE NULL VALUES:

Here I treated the null values with spaces . (i.e)I replaced the null values with empty space.



```
quality
dtype: int64

In [37]: data1=data.fillna("")
#data1.fillna(data.mean())
#data1.dropna()

In [38]: data1.isnull().sum()

Out[38]: 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

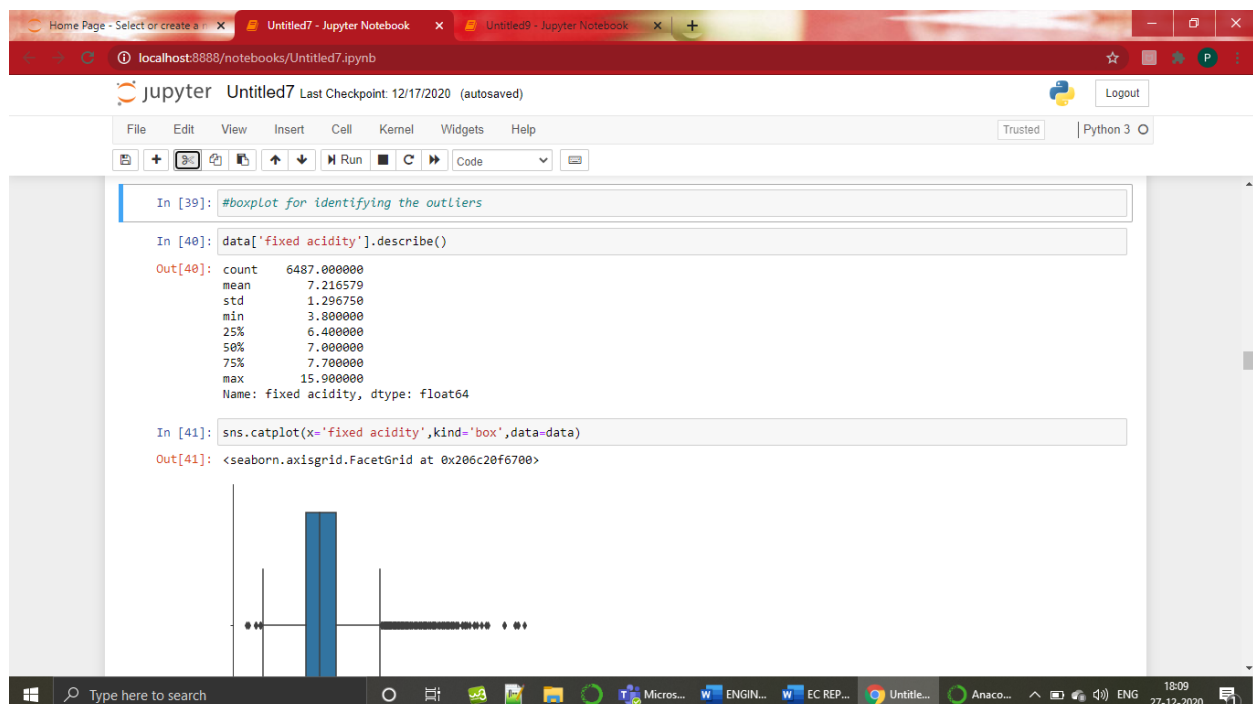
In [39]: #boxplot for identifying the outliers

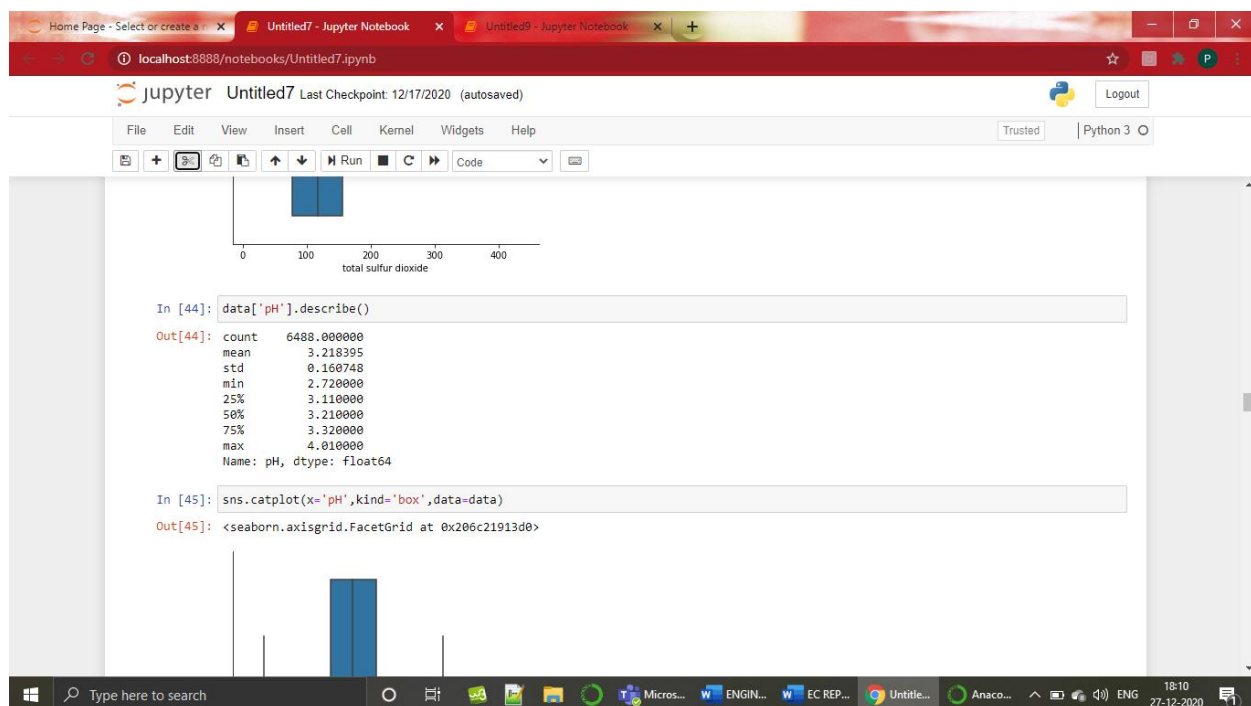
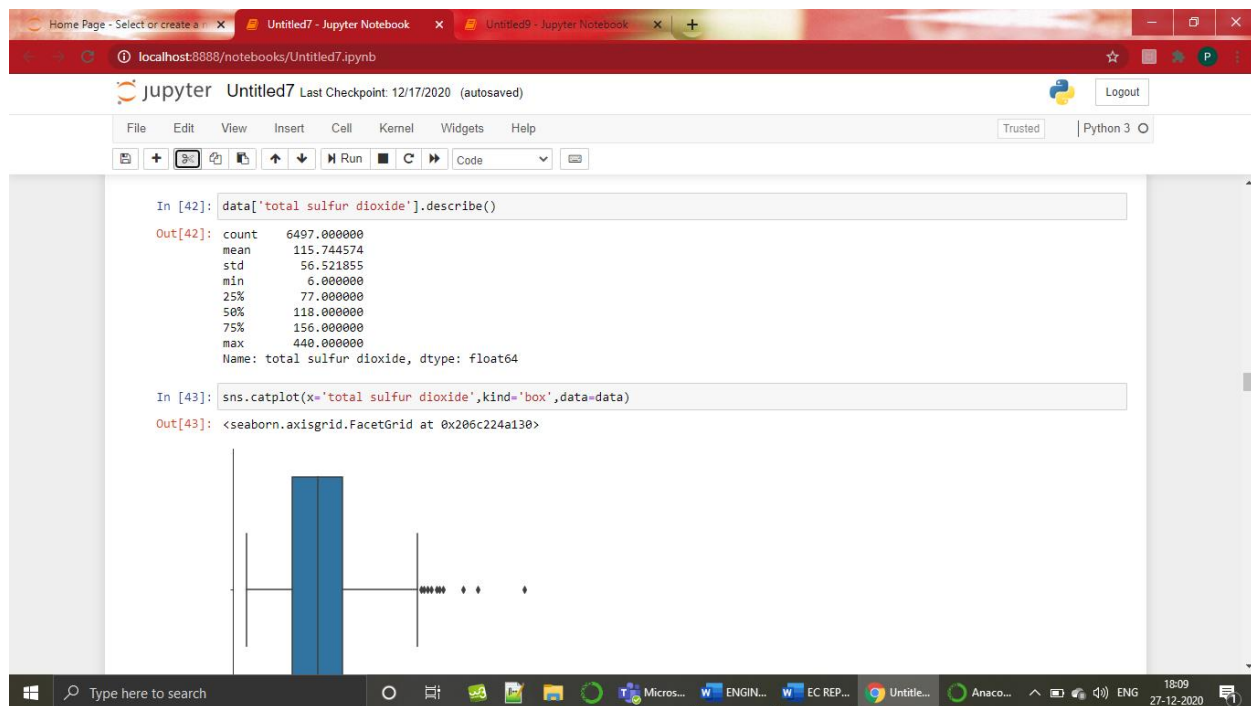
In [40]: data['fixed acidity'].describe()

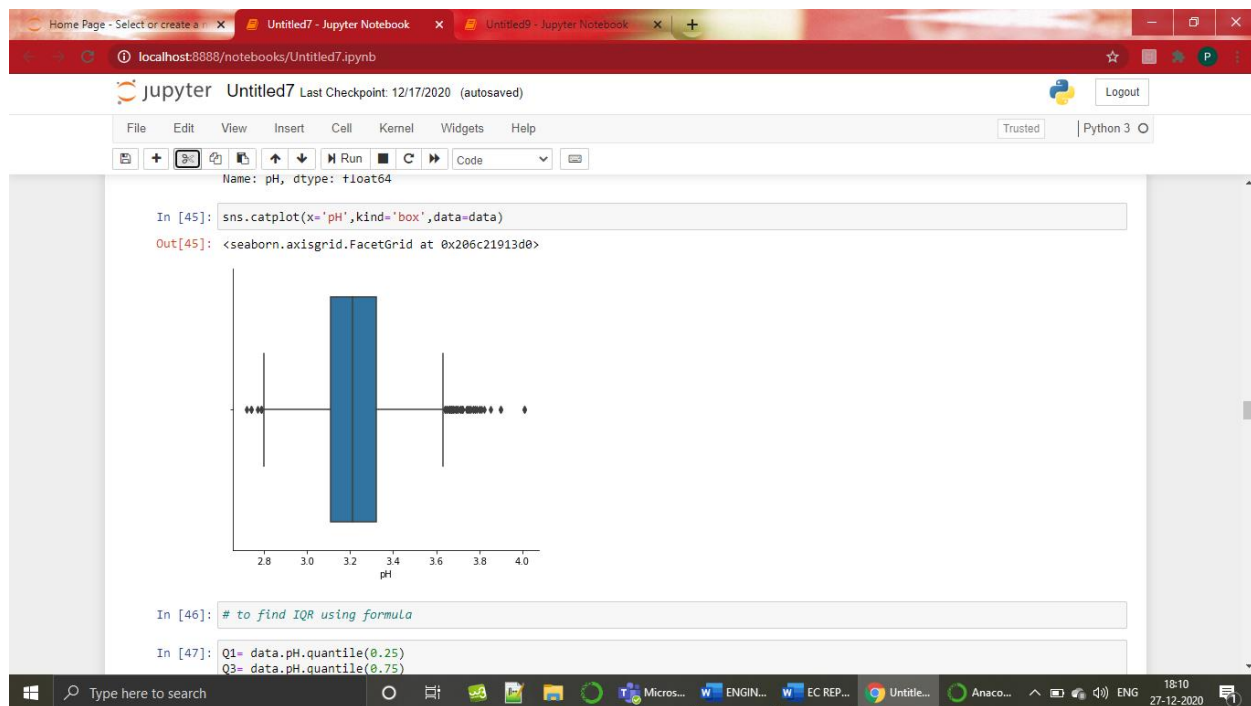
Out[40]: count    6487.000000
mean         7.216579
std          1.296750
min          3.800000
25%          6.400000
50%          7.000000
75%          7.700000
max          15.900000
Name: fixed acidity, dtype: int64
```

4.2.2. CHECKING FOR OUTLIERS

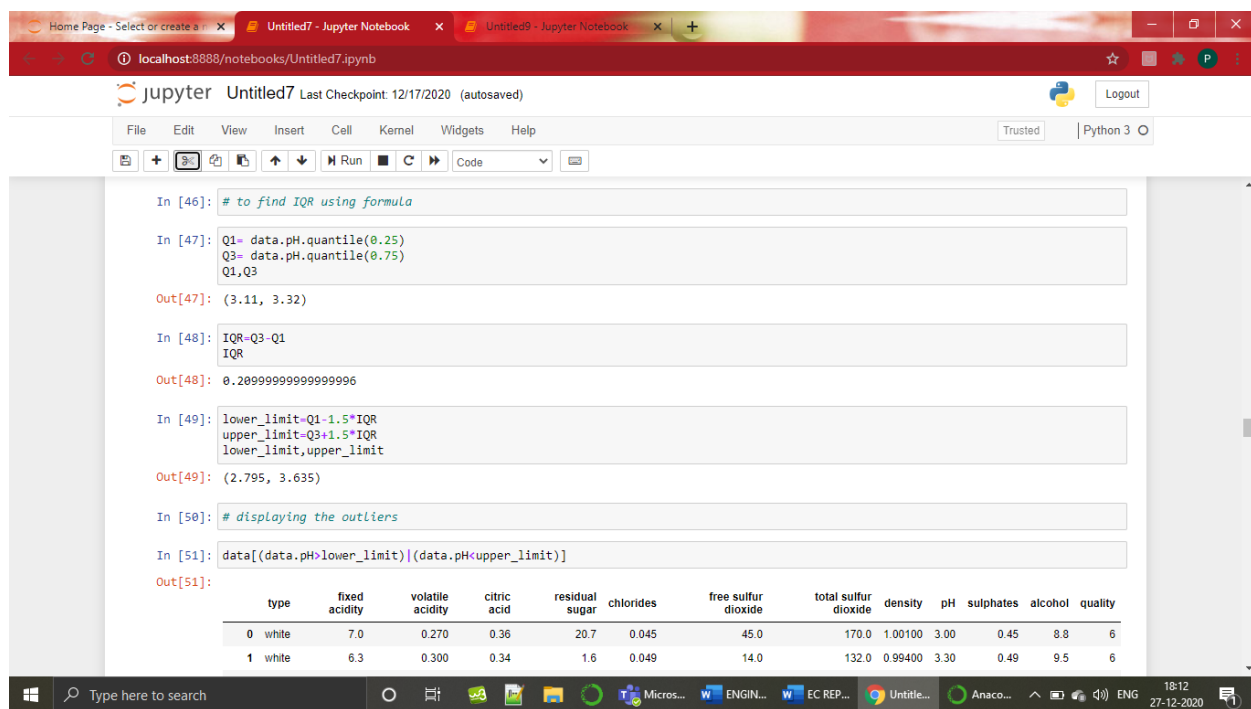
USING BOXPLOT:







4.2.3 TREATING THE OUTLIERS



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

```
In [50]: # displaying the outliers
In [51]: data[(data.pH>lower_limit)|(data.pH<upper_limit)]
Out[51]:
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.270	0.36	20.7	0.045	45.0	170.0	1.00100	3.00	0.45	8.8	6
1	white	6.3	0.300	0.34	1.6	0.049	14.0	132.0	0.99400	3.30	0.49	9.5	6
2	white	8.1	0.280	0.40	6.9	0.050	30.0	97.0	0.99510	3.26	0.44	10.1	6
3	white	7.2	0.230	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	9.9	6
4	white	7.2	0.230	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	9.9	6
...
6492	red	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
6493	red	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	NaN	11.2	6
6494	red	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
6495	red	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
6496	red	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

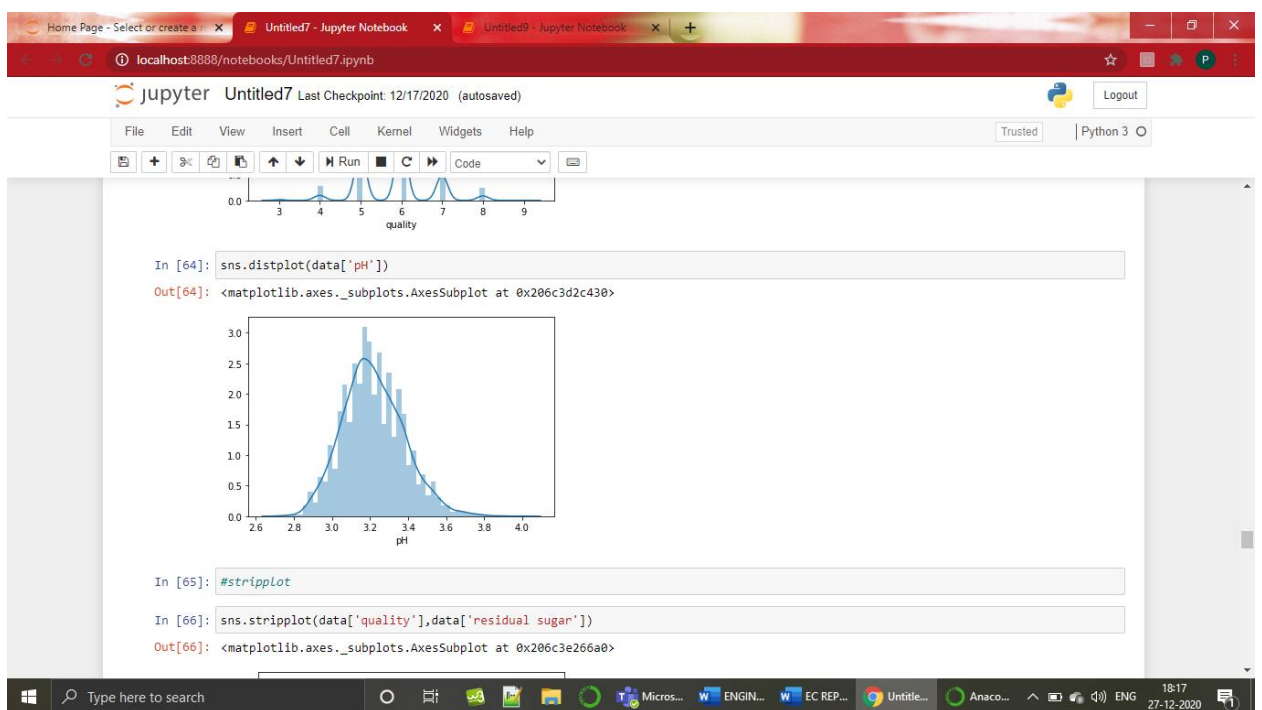
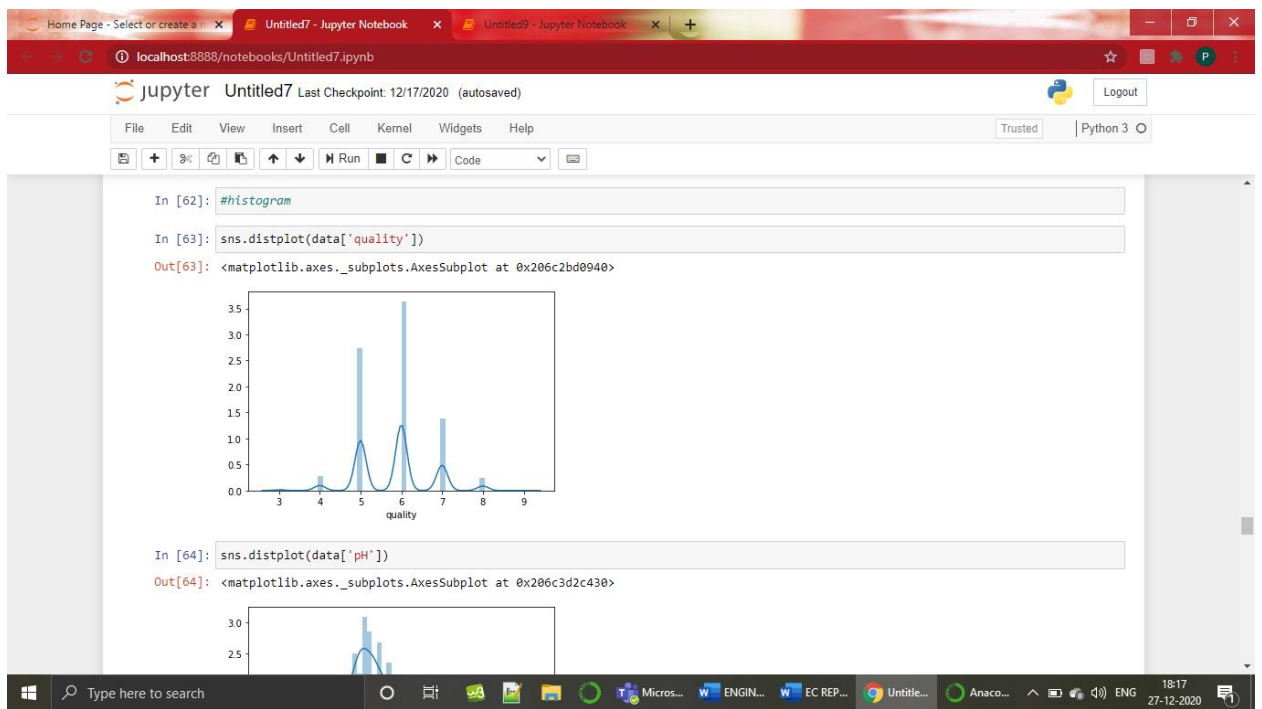
6488 rows x 13 columns

```
In [52]: #dataset without the outliers
In [53]: data2=data[(data.pH>lower_limit)&(data.pH<upper_limit)]
```

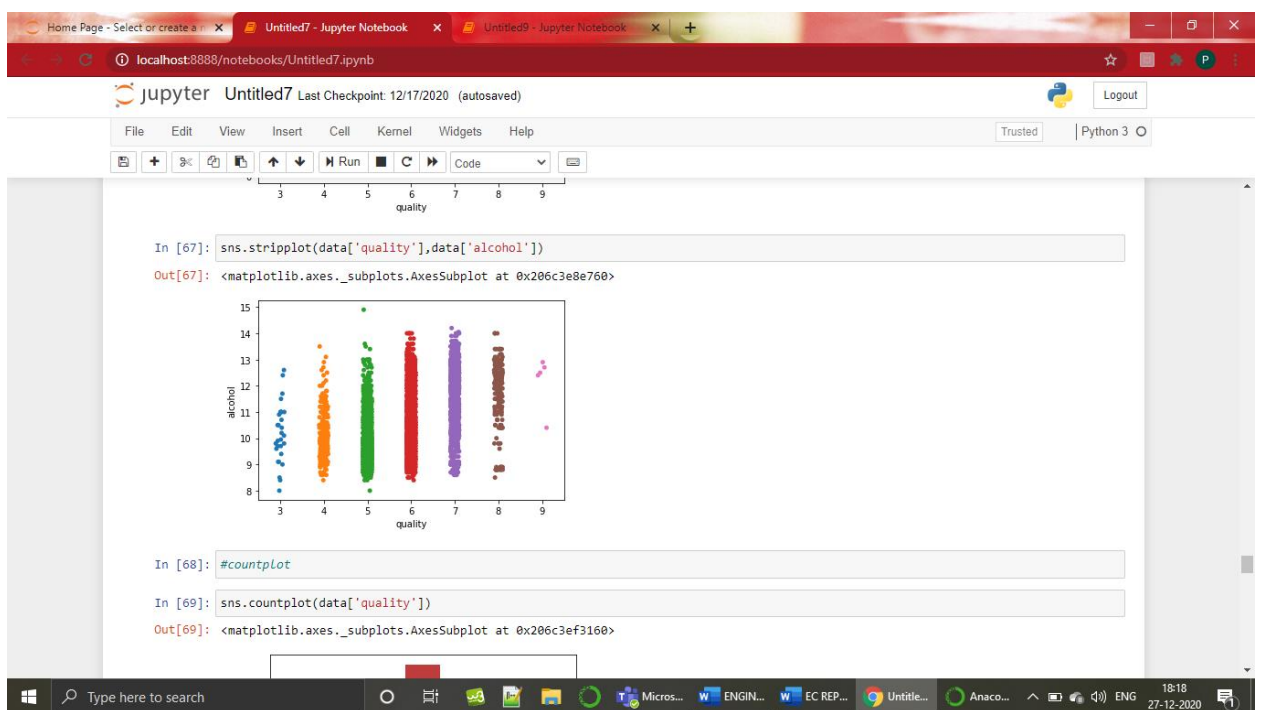
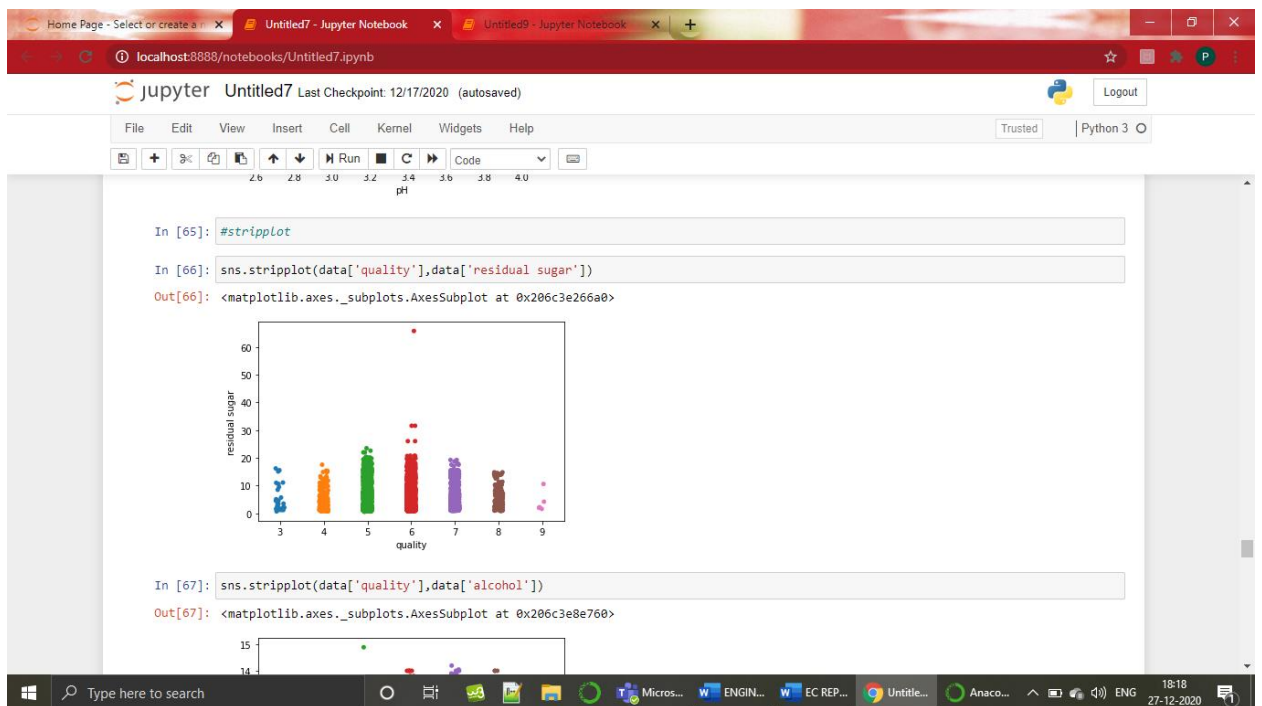
4.3.DATA VISUALIZATION

Data visualization is the graphical representation of information and data. With visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

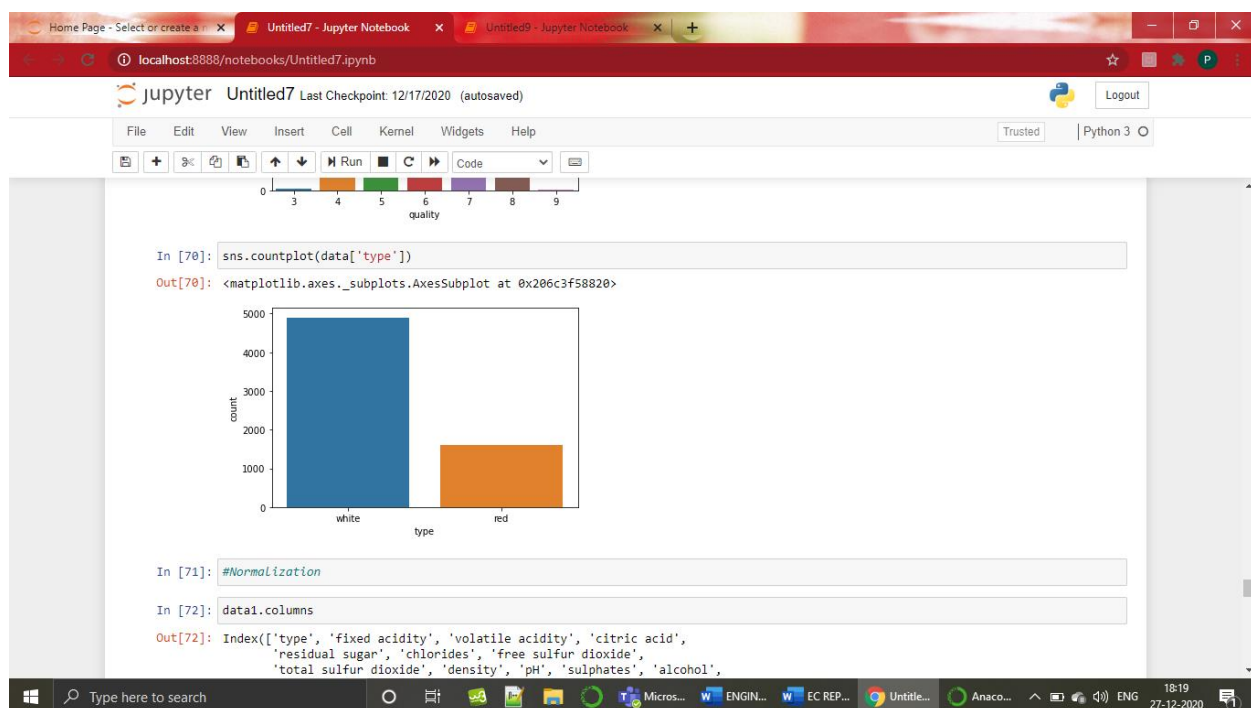
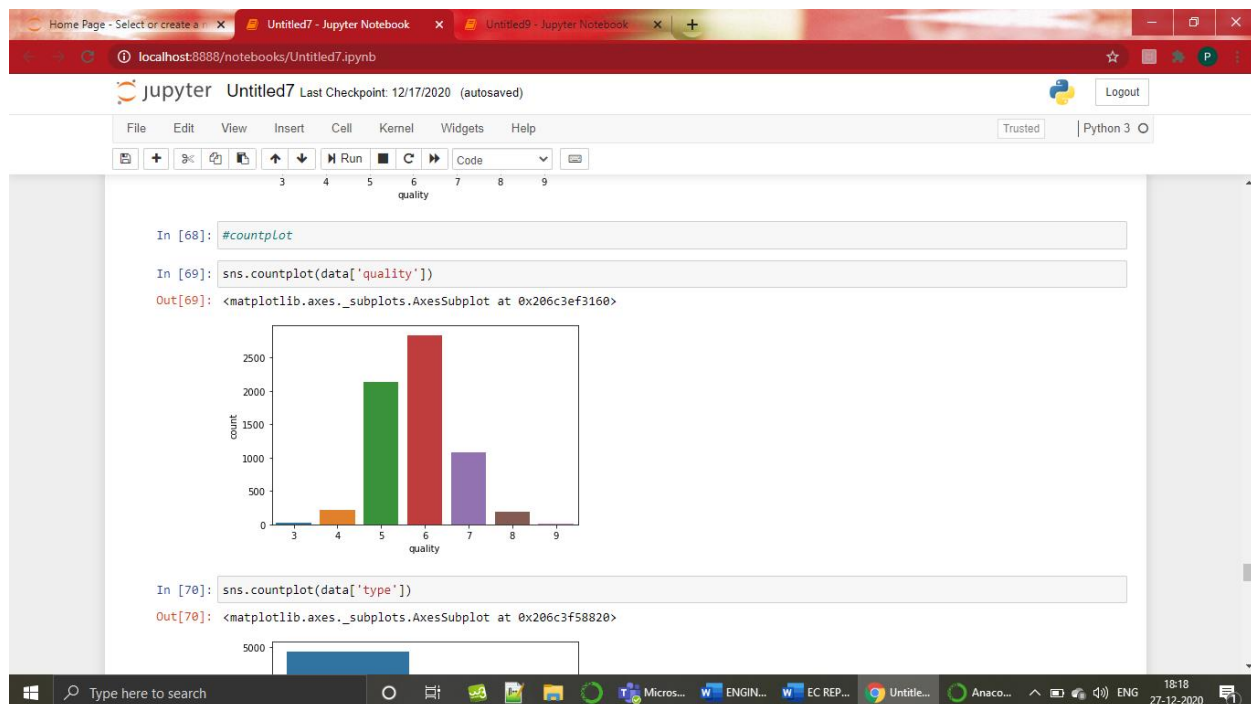
4.3.1 HISTOGRAM:



4.3.2 STRIPLOT:



4.3.3 COUNTPLOT:



4.4. NORMALIZATION

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

The screenshot shows a Jupyter Notebook titled "Untitled7" running on localhost:8888. The notebook contains several code cells performing data preprocessing:

- In [74]: `data2=data1[['free sulfur dioxide','total sulfur dioxide', 'density','alcohol','quality']]`
- In [75]: `x_data=data1[['free sulfur dioxide','total sulfur dioxide', 'density','alcohol','quality']]`
- In [76]: `#using minmax scaler(normalization)`
- In [77]: `from sklearn.preprocessing import MinMaxScaler`
- In [78]: `scaling_data=MinMaxScaler()`
- In [79]: `scaling_data.fit_transform(x_data)`

The output of In [79] is displayed as follows:

```
Out[79]: array([[0.15277778, 0.37788018, 0.26778485, 0.11594203, 0.5],
               [0.04513889, 0.29032258, 0.13283208, 0.2173913 , 0.5],
               [0.10069444, 0.20967742, 0.15403894, 0.30434783, 0.5],
               ...,
               [0.09722222, 0.07834101, 0.16637748, 0.43478261, 0.5],
               [0.10763889, 0.0875576 , 0.16117216, 0.31884058, 0.33333333],
               [0.05902778, 0.08294931, 0.16155774, 0.43478261, 0.5]])
```

In [80]: `x_data=x_data.apply(lambda x:(x-x.min(axis=0))/(x.max(axis=0)-x.min(axis=0)))`

In [81]: `x_data`

The output of In [81] is displayed as a table:

	free sulfur dioxide	total sulfur dioxide	density	alcohol	quality
0	0.152778	0.377880	0.267785	0.115942	0.500000

Home Page - Select or create a notebook x Untitled7 - Jupyter Notebook x Untitled9 - Jupyter Notebook x +

localhost:8888/notebooks/Untitled7.ipynb

jupyter Untitled7 Last Checkpoint: 12/17/2020 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [80]: x_data=x_data.apply(lambda x:(x-x.min(axis=0))/(x.max(axis=0)-x.min(axis=0)))
```

```
In [81]: x_data
```

```
Out[81]:
```

	free sulfur dioxide	total sulfur dioxide	density	alcohol	quality
0	0.152778	0.377880	0.267785	0.115942	0.500000
1	0.045139	0.290323	0.132832	0.217391	0.500000
2	0.100694	0.209677	0.154039	0.304348	0.500000
3	0.159722	0.414747	0.163678	0.275362	0.500000
4	0.159722	0.414747	0.163678	0.275362	0.500000
...
6492	0.107639	0.087558	0.150183	0.362319	0.333333
6493	0.131944	0.103687	0.154425	0.463768	0.500000
6494	0.097222	0.078341	0.166377	0.434783	0.500000
6495	0.107639	0.087558	0.161172	0.318841	0.333333
6496	0.059028	0.082949	0.161558	0.434783	0.500000

6497 rows x 5 columns

```
In [82]: #maximum absolute scaling
```

```
In [83]: from sklearn.preprocessing import MaxAbsScaler
```

Home Page - Select or create a notebook x Untitled7 - Jupyter Notebook x Untitled9 - Jupyter Notebook x +

localhost:8888/notebooks/Untitled7.ipynb

jupyter Untitled7 Last Checkpoint: 12/17/2020 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [82]: #maximum absolute scaling
```

```
In [83]: from sklearn.preprocessing import MaxAbsScaler
```

```
In [84]: absscaler=MaxAbsScaler()
```

```
In [85]: absscaler.fit(data2)
```

```
absscaler.max_abs_
```

```
Out[85]: array([289.      , 440.      , 1.03898, 14.9      , 9.      ])
```

```
In [86]: data5=absscaler.transform(data2)
```

```
scaled_data5=pd.DataFrame(data5,columns=data2.columns)
```

```
In [87]: scaled_data5
```

```
Out[87]:
```

	free sulfur dioxide	total sulfur dioxide	density	alcohol	quality
0	0.155709	0.386364	0.963445	0.590604	0.666667
1	0.048443	0.300000	0.956708	0.637584	0.666667
2	0.103806	0.220455	0.957766	0.677852	0.666667
3	0.162630	0.422727	0.958248	0.664430	0.666667
4	0.162630	0.422727	0.958248	0.664430	0.666667
...
6492	0.110727	0.100000	0.957574	0.704698	0.555556

4.5 PREDICTION OF TARGET VARIABLE 1

```

In [14]: y = data.quality
X = data.drop('quality', axis=1)

In [15]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

In [16]: print(X_train.head())

    fixed acidity  volatile acidity  citric acid  residual sugar  chlorides \
1344          11.5              0.42         0.48             2.6         0.077
120           7.3              1.07         0.09             1.7         0.178
1125           8.8              0.24         0.35             1.7         0.055
1233          10.2              0.23         0.37             2.2         0.057
1389           6.7              0.48         0.02             2.2         0.080

    free sulfur dioxide  total sulfur dioxide  density  pH  sulphates \
1344                8.0                20.0  0.99852  3.09    0.53
120               10.0                89.0  0.99620  3.30    0.57
1125               13.0                27.0  0.99394  3.14    0.59
1233               14.0                36.0  0.99614  3.23    0.49
1389               36.0                111.0  0.99524  3.10    0.53

    alcohol
1344     11.0
120      9.0
1125     11.3
1233      9.3
1389      9.7

In [18]: X_train_scaled = preprocessing.scale(X_train)
print(X_train_scaled)

[[ 1.78453191 -0.59635845  1.06977747 ... -1.4124159 -0.75603198
  0.55376823]
 [-0.59865935  3.12517435 -0.94835989 ... -0.06093465 -0.52056019
 -1.31954779]
 [ 0.25248039 -1.62693677  0.39706501 ... -1.09063465 -0.4028243
  0.83476563]
 ...
 [-0.08797551  0.63461009 -0.89661278 ... -0.1252909  0.53906286
  0.46010243]
 [ 1.10362012  0.37696551  0.7592948 ... -0.18964715  0.24472312
 -0.85121879]
 [-0.48517405 -1.22615631  0.70754768 ...  0.19649035  1.12774233
  0.92843143]]

```

```

1389      9.7

In [18]: X_train_scaled = preprocessing.scale(X_train)
print(X_train_scaled)

[[ 1.78453191 -0.59635845  1.06977747 ... -1.4124159 -0.75603198
  0.55376823]
 [-0.59865935  3.12517435 -0.94835989 ... -0.06093465 -0.52056019
 -1.31954779]
 [ 0.25248039 -1.62693677  0.39706501 ... -1.09063465 -0.4028243
  0.83476563]
 ...
 [-0.08797551  0.63461009 -0.89661278 ... -0.1252909  0.53906286
  0.46010243]
 [ 1.10362012  0.37696551  0.7592948 ... -0.18964715  0.24472312
 -0.85121879]
 [-0.48517405 -1.22615631  0.70754768 ...  0.19649035  1.12774233
  0.92843143]]

In [19]: clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)

Out[19]: DecisionTreeClassifier()

In [20]: confidence = clf.score(X_test, y_test)
print("\nThe confidence score:\n")
print(confidence)

The confidence score:

0.590625

In [21]: y_pred = clf.predict(X_test)

In [23]: x=np.array(y_pred).tolist()

#printing first 5 predictions

```

```
0.590625

In [21]: y_pred = clf.predict(X_test)

In [23]: x=np.array(y_pred).tolist()

#printing first 5 predictions
print("\nThe prediction:\n")
for i in range(0,5):
    print (x[i])

#printing first five expectations
print("\nThe expectation:\n")
print (y_test.head())

The prediction:
7
6
6
5
6

The expectation:
1453    5
262     5
1286    6
482     5
202     5
Name: quality, dtype: int64

In [ ]:

In [ ]:
```

CONCLUSION:

By looking into the details, we can see that good quality wines have higher levels of alcohol on average, have a lower volatile acidity on average, higher levels of sulphates on average, and higher levels of residual sugar on average.

Reference link:

- <https://towardsdatascience.com/predicting-wine-quality-with-several-classification-techniques-179038ea6434>
- <https://www.kaggle.com/scsaurabh/red-wine-quality-analysis-python>
- <https://www.kaggle.com/sgus1318/wine-quality-exploration-and-analysis>

- <https://medium.com/analytics-vidhya/wine-quality-prediction-with-python-695939d34d87>
- <https://medium.com/datadriveninvestor/regression-from-scratch-wine-quality-prediction-d61195cb91c8>
- <https://github.com/vikrantkakad/Red-Wine-Quality-Analysis>
- <https://dzone.com/articles/predicting-wine-quality-with-several-classificatio>
- https://datauab.github.io/red_wine_quality/