Using the CRISP-DM Method for MLN 601 Machine Learning

Assessment 1: Regression Analysis

Adapted from Smart Vision Europe (2020)https://www.sv-europe.com/crisp-dm-methodology/ and Patience (2018) https://grantpatience.com/2018/10/30/applying-crisp-dm-to-data-science-and-a-re-usable-template-in-jupyter/

CRISP-DM stands for cross-industry process for data mining and represents a robust and proven framework available since 1996. While providing a structured approach to planning a data mining project is extremely useful for the documentation, building and deployment of ML projects. The model represents an ideal framework but tasks can be executed in a different sequence and if necessary repeated. The template is a cut down version for use in the completion of your assessment. Any code is shown as comments for example purposes and not necessiarily intended to be completely working. Your final assessment report should follow this template and include your code and narrative text.

The CRISP steps are:

1. Gain an understanding of the business
2. Gain an understanding of the data
3. Prepare the data
4. Complete modeling
5. Evaluate
6. Deploy

1. Stage One - Determine Business Objectives and Assess the Situation

The red wine industry has exponentially growth during the last years. However gaining recognition is crucial to be on top of the industry. Hence, it is key to obtain red wine certification and quality assessment, which is a physicochemical test laboratory-based. Factors such as acidity, pH level, sugar, and other chemical properties are necessary. The red wine market would be of interest if the human quality of tasting can be related to wine's chemical properties so that certification and quality assessment and assurance processes are more controlled. The present project seeks to determine which features are the best quality red wine indicators and generate insights into each of these factors to our model's red wine quality.

1.1 Assess the Current Situation

List of the resources available to the project include:

- Personnel: **Yolanda Carreno Gomez**
- Data: /content/winequality-red.csv taken from https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/
- Computing resources: Google Colab Notebook
- Software: Python 3
- Python Libraries: pandas, matplotlib, numpy and seaborn

2. Stage Two - Data Understanding

My analysis will use Red Wine Quality Data Set, available on the UCI machine learning repository (https://archive.ics.uci.edu/ml/datasets/wine+quality). I obtained the red wine samples 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. 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 red wine quality will help producers, distributors, and businesses in the red wine industry better assess their production, distribution, and pricing strategy.

2.1 Initial Data Acquisition

The following code shows the libraries required to run the project and the the data source and location is detailed in the comments bellow:

```
# Import Libraries Required
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

```
#Data source: The exercise focuses on predicting only the quality of the red wine dataset extracted from https://archive.i
#Source Query location: /content/winequality-red.csv
#path =  '/content/winequality-red.csv'
path = 'https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv'
# reads the data from the file - denotes as CSV, it has no header, sets column headers
df =  pd.read_csv(path, sep=';')
```

2.2 Describe Data

Whitin this step is very important to clean and prepare the data for analysis, which I have done it. I went through different steps of data cleaning. First, I checked the data types, identified the number of columns and rows, as well as the headings so I nknow what variables describe the red wine dataset.

In order of highest correlation, these variables are:

1. Alcohol: the amount of alcohol in wine
2. Volatile acidity: are high acetic acid in wine which leads to an unpleasant vinegar taste
3. Sulphates: a wine additive that contributes to SO2 levels and acts as an antimicrobial and antioxidant
4. Citric Acid: acts as a preservative to increase acidity (small quantities add freshness and flavor to wines)
5. Total Sulfur Dioxide: is the amount of free + bound forms of SO2
6. Density: sweeter wines have a higher density

7. Chlorides: the amount of salt in the wine
8. Fixed acidity: are non-volatile acids that do not evaporate readily
9. pH: the level of acidity
10. Free Sulfur Dioxide: it prevents microbial growth and the oxidation of wine
11. Residual sugar: is the amount of sugar remaining after fermentation stops. The key is to have a perfect balance between — sweetness and sourness (wines > 45g/ltrs are sweet)

```
#df.columns, df.shape, df.dtypes, df.describe(), df.info() and df.head(10) Use Pandas to explore and clean up your tabular
```

```
df.columns
```

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

```
df.shape
```

```
(1599, 12)
```

```
df.dtypes
```

```
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide     float64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
alcohol                 float64
quality                   int64
dtype: object
```

```
df.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	d
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
```

```
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

```
df.head(5)
```

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

```
df['quality'].value_counts()
```

```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

```
df.isnull().sum()
```

```
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
```

2.3 Verify Data Quality

Examine the quality of the data:

- Is the data complete (does it cover all that you require)?
- Is it correct, or does the data contain errors ?
- Are there missing values in the data? If so, where do they occur?

all good! Y

2.3.1. Outliers

At this point, we may also want to remove any outliers. These can be due to typos in data entry, mistakes in units, or they could be legitimate but extreme values or rare events. For this assessment, you don't need to worry about outliers. However, you would remove anomalies based on the definition of extreme outliers:

https://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm

- Below the first quartile − 3 $*$ interquartile range
- Above the third quartile + 3 $*$ interquartile range

2.4 Initial Data Exploration

During this stage, address data questions using querying, data visualization and reporting techniques. These may include:

- **Distribution** of key attributes (for example, the target attribute of a prediction task)
- **Relationships** between pairs or small numbers of attributes
- Results of **simple aggregations**

- **Properties** of significant sub-populations
- **Simple** statistical analyses

These analyses may contribute to or refine the data description and quality aspects of your report, and feed into other data preparation steps needed for further analysis.

- **Data exploration component of your report** - Describe results of your data exploration, including first findings or initial hypothesis and their impact on the remainder of the project. Include graphs and plots here to indicate data characteristics that suggest further examination of interesting data subsets.

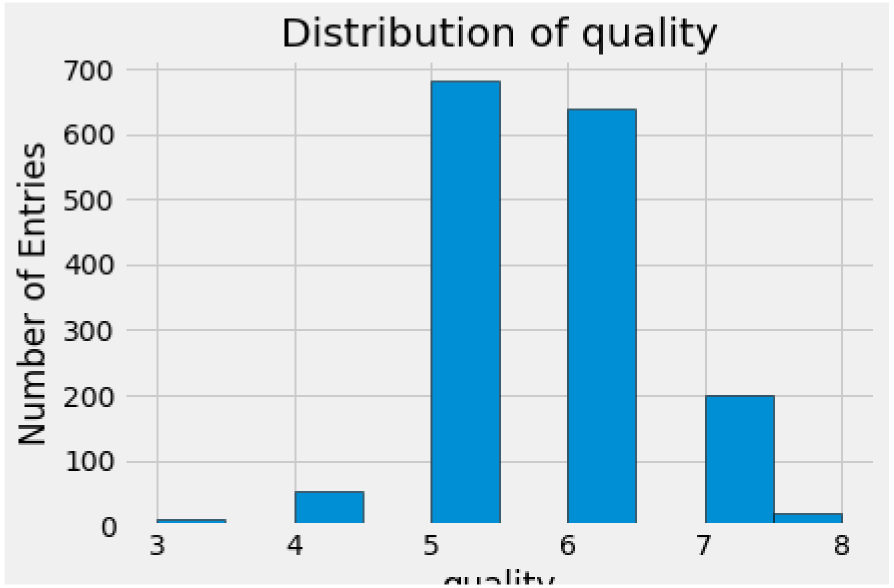
2.4.1 Distributions

```
def count_values_table(df):
        count_val = df.value_counts()
        count_val_percent = 100 * df.value_counts() / len(df)
        count_val_table = pd.concat([count_val, count_val_percent.round(1)], axis=1)
        count_val_table_ren_columns = count_val_table.rename(
        columns = {0 : 'Count Values', 1 : '% of Total Values'})
        return count_val_table_ren_columns
```

```
# Histogram
def hist_chart(df, col):
        plt.style.use('fivethirtyeight')
        plt.hist(df[col].dropna(), edgecolor = 'k');
        plt.xlabel(col); plt.ylabel('Number of Entries');
        plt.title('Distribution of '+col);
```

```
col = 'quality'
#Histogram & Results
hist_chart(df, col)
count_values_table(df.quality)
```

	quality	quality
5	681	42.6
6	638	39.9
7	199	12.4
4	53	3.3
8	18	1.1
3	10	0.6



The histogram shows that the majority of red wine have score 5 and 6, meaning that wine with quality score of 3, 4 and 8 represent the outliers in this dataset.

2.4.2 Correlations

To see which variables are likely to affect the quality of red wine the most, I ran a correlation analysis of independent variables against the dependent variable, **quality**. The graphs bellow shows a list of variables of interest that had the highest correlations with quality.

Any correlation from this data-set can be derived. I have used Pairplot chart which gives us correlations, distributions and regression path.

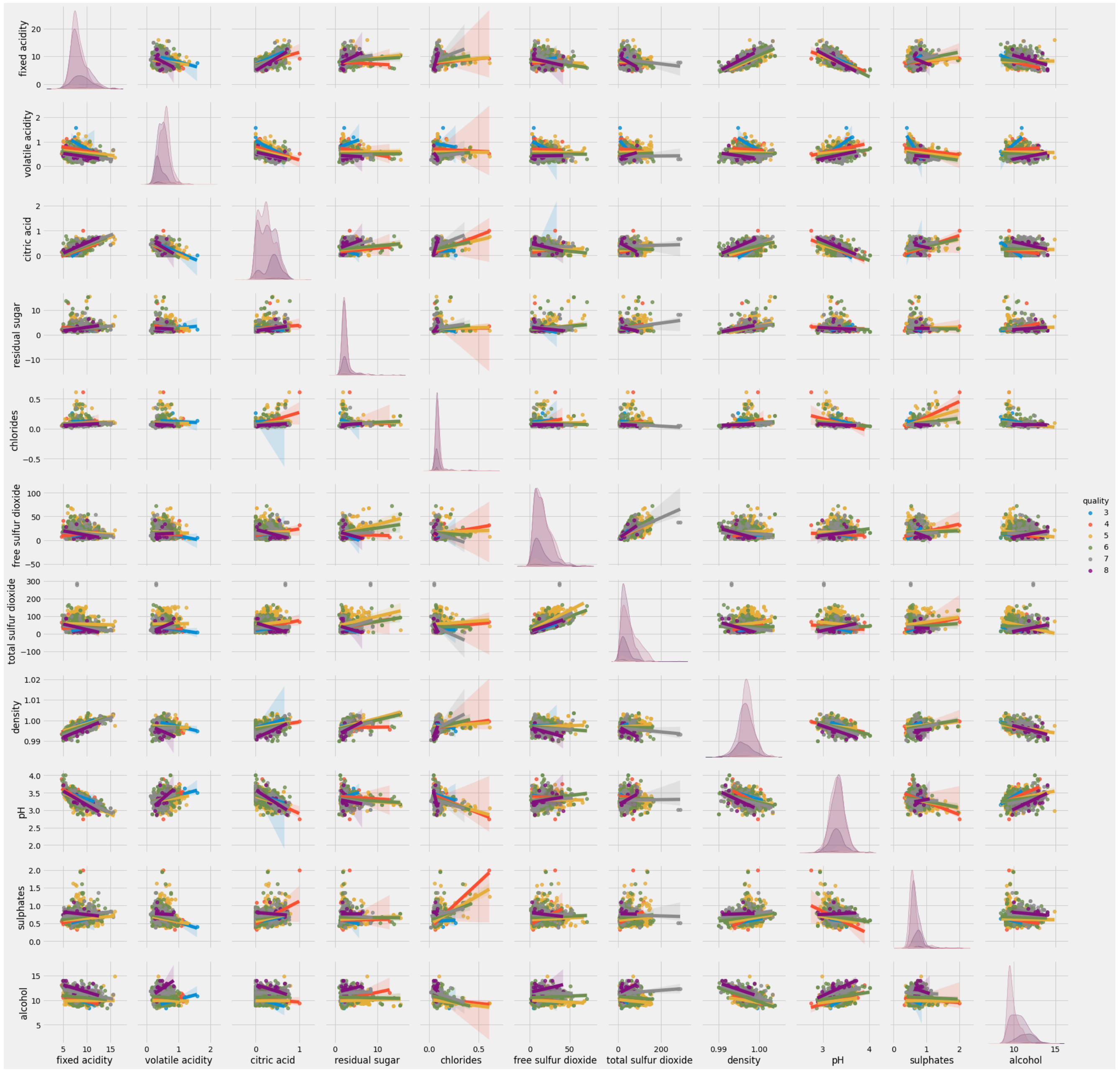
Correlogram are awesome for exploratory analysis. It allows to quickly observe the relationship between every variable of your matrix. It is easy to do it with seaborn: just call the pairplot function

Pairplot documentation is found here: https://seaborn.pydata.org/generated/seaborn.pairplot.html

```
# get all correlations of data table
correlations = df.corr()

# get relationship with target column -> quality
quality_corr = correlations['quality']

# remove target column from the target correlations
quality_corr = quality_corr.drop('quality')

# print all relationship values
print(quality_corr)
```

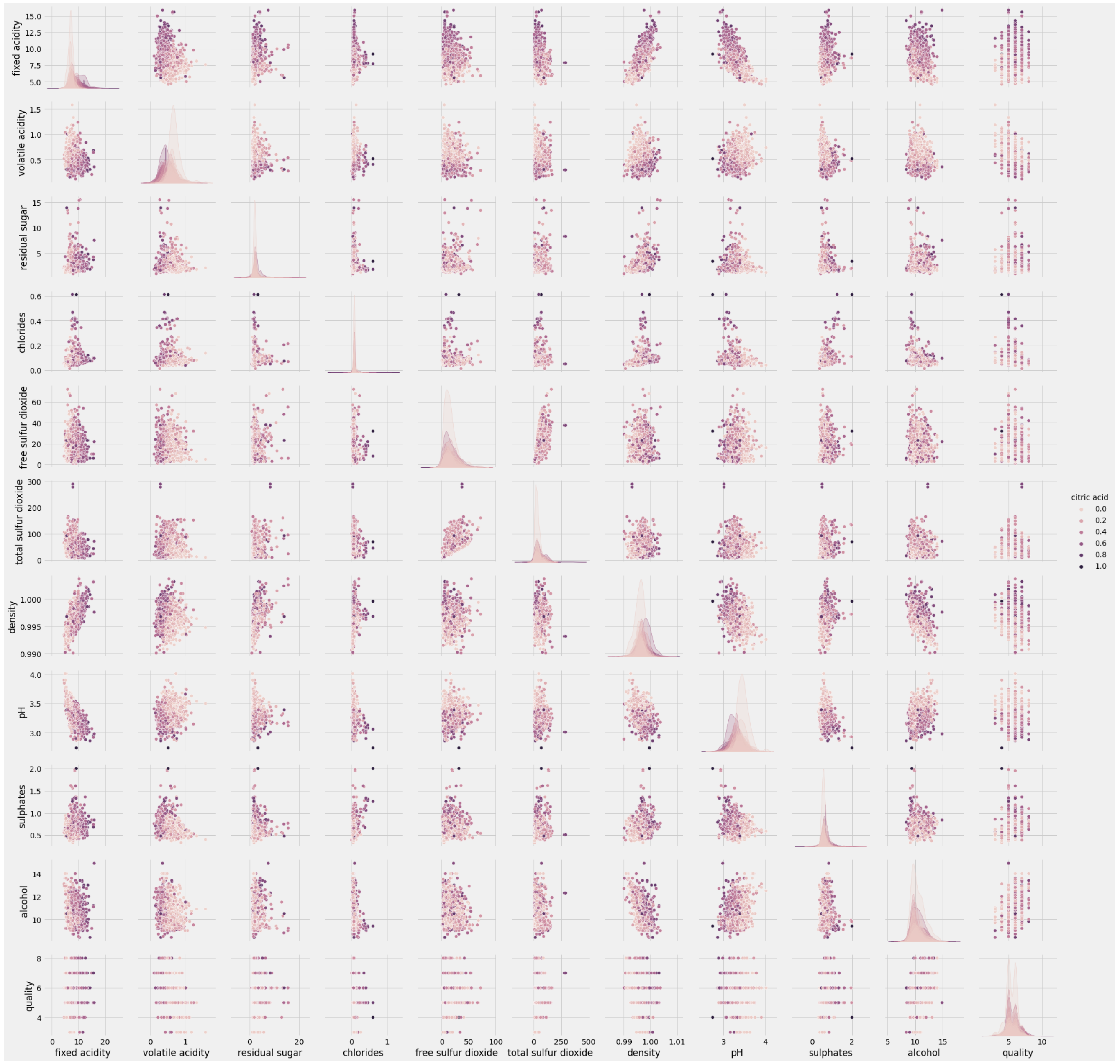
```
fixed acidity            0.124052
volatile acidity        -0.390558
citric acid              0.226373
residual sugar           0.013732
chlorides               -0.128907
free sulfur dioxide     -0.050656
total sulfur dioxide    -0.185100
density                 -0.174919
pH                      -0.057731
sulphates                0.251397
alcohol                  0.476166
Name: quality, dtype: float64
```

```
#Seaborn allows to make a correlogram or correlation matrix really easily.
sns.pairplot(df, hue='quality', kind ='reg')

plt.show()
```


fixed acidity
volatile acidity
citric acid
residual sugar
chlorides
free sulfur dioxide
total sulfur dioxide
density
pH
sulphates
alcohol
quality
3
4
5
6
7
8

The graph above shows the relation of each property of the red wine against the quality, it is shown in 8 different colours which denotates the score of quality vs other properties. The plot has a regression format.

```
sns.pairplot(df, hue='citric acid', kind ='scatter')

plt.show()
```


fixed acidity
volatile acidity
residual sugar
chlorides
free sulfur dioxide
total sulfur dioxide
density
pH
sulphates
alcohol
quality
citric acid
0.0
0.2
0.4
0.6
0.8
1.0

```
#agg means agregate
#df_agg = df.drop(['quality'], axis=1).groupby(['fixed acidity']).sum() #data columns to drop: quality and fixed acidity
df_agg = df.groupby(['quality']).sum() #grouping the data by quality and making a sum of it
df_agg
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	densi
quality								
3	83.6	8.845	1.71	26.35	1.225	110.0	249.0	9.97
4	412.3	36.780	9.23	142.80	4.806	650.0	1921.0	52.81
5	5561.9	392.965	165.95	1722.15	63.153	11566.0	38486.0	679.02
6	5325.5	317.395	174.70	1580.45	54.202	10024.0	26075.0	635.84
7	1765.6	80.380	74.66	541.40	15.241	2795.0	6969.0	198.22

This table agregates the data per quality

3. Stage Three - Data Preparation

This is the stage of the project where you decide on the data that you're going to use for analysis. The criteria you might use to make this decision include the relevance of the data to your data mining goals, the quality of the data, and also technical constraints such as limits on data volume or data types. Note that data selection covers selection of attributes (columns) as well as selection of records (rows) in a table.

3.1 Select Your Data

This is the stage of the project where you decide on the data that you're going to use for analysis. The criteria you might use to make this decision include the relevance of the data to your machine learning goal, the quality of the data, and also technical constraints such as limits on data volume or data types. Note that data selection covers selection of attributes (columns) as well as selection of records (rows) in a table.

Rationale for inclusion/exclusion - List the data to be included/excluded and the reasons for these decisions.

```
x = df.iloc[:, :-1]
# pick quality for training model
y = df.iloc[:, -1]

print(x.shape)
print(y.shape)
```

```
(1599, 11)
(1599,)
```

```
#dividing the dataset in training 75% and testing 25% set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25, random_state=4)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(1199, 11)
(1199,)
(400, 11)
(400,)
```

3.2 Clean The Data

This task involves raising the data quality to the level required by the analysis techniques that you've selected. This may involve selecting clean subsets of the data, the insertion of suitable defaults, or more ambitious techniques such as the estimation of missing data by modelling.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
x_test = sc.fit_transform(x_test)
```

4. Stage Four - Modelling

As the first step in modelling, you'll select the actual modelling technique that you'll be using e.g.Linear Regression

4.1. Modelling technique

Document the actual modelling technique that is to be used.

Import Models in your code below:

```
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

4.2. Modelling assumptions

Many modelling techniques make specific assumptions about the data, for example that all attributes have uniform distributions, no missing values allowed, class attribute must be symbolic etc. Record any assumptions made.

4.3. Build Model

Run the modelling tool on the prepared dataset to create your model.

Parameter settings - With any modelling tool there are often a large number of parameters that can be adjusted. List the parameters and their chosen values, along with the rationale for the choice of parameter settings.

Model - This is the actual model produced by the modelling tool, not a report on the model.

Model description - Describe the resulting model, report on the interpretation of the model and document any difficulties encountered with their meanings.

```
#creating the model
regressor = LinearRegression()
#feeding the training set into the model
regressor.fit(x_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
#predicting hte results for the test set
y_pred = regressor.predict(x_test)
print(y_pred[:20])
```

```
[5.68598931 5.34666283 6.13697353 6.14204803 5.33448077 5.13307534
 4.85568134 6.35176096 5.12387676 6.34155348 6.24152411 6.38227318
 5.45726253 5.63108402 6.34728055 5.57787493 5.28737328 5.64592507
 5.22376534 4.74989695]
```

4.4. Assess Model

Interpret the models according to your knowledge, your prediction success criteria and your desired test design. Judge the success of the application of modelling and discovery techniques technically to discuss the machine learning results in the business context. This task only considers models, whereas the evaluation phase also takes into account all other results that were produced in the course of the project.

At this stage you should rank the models and assess them according to the evaluation criteria. You should take the business objectives and business success criteria into account as far as you can here. In most ML projects a single technique is applied more than once and results are generated with several different techniques.

Model assessment - Summarise the results of this task, list the qualities of your generated models (e.g.in terms of accuracy) and rank their quality in relation to each other.

Revised parameter settings - According to the model assessment, revise parameter settings and tune them for the next modelling run. Iterate

```
df_test = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df_test
```

	Actual	Predicted
289	5	5.685989
962	5	5.346663
826	7	6.136974
495	8	6.142048
57	5	5.334481
...	...	...
225	6	5.708679
322	5	5.301800
677	5	5.058051
1182	6	6.188560
1561	5	5.126900

400 rows × 2 columns

```
#calculating the training and testing accuracies
accuracy = regressor.score(x_test,y_test)
print(accuracy)
```

```
0.3460417550418452
```

5. Stage 5 - Evaluate

Previous steps deal with the accuracy and generality of the model. During this step you should assesses the degree to which the model meets your business objectives and seek to determine if there is some business reason why this model is deficient.

Model evaluation is very important in data science. It helps understand the performance of the models. There are many different evaluation metrics out there but only some of them are suitable to be used for regression analysis as Mean Absolute Error, Mean Squared Error and Root Mean Squared Error.

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 0.49331616571013653
Mean Squared Error: 0.40577700375750403
Root Mean Squared Error: 0.6370062823532465
```

6. Stage 6 - Deploy

In conclusion, it can be observed that the value of root mean squared error is 0.637 which is 0.11% of the mean value 5.636023 of the red wine quality whose score is between 1 and 10. The algorithm did a great job. However, the accuracy is not that significant.