Machine Learning Mini Project – Report

Problem Statement:

The dataset is related to the red variant of the Portuguese "Vinho Verde" wine.
These datasets can be viewed as regression tasks. The classes are ordered and not balanced (e.g. there are much more normal wines than excellent or poor ones). Apply Regression and find the quality of Wine

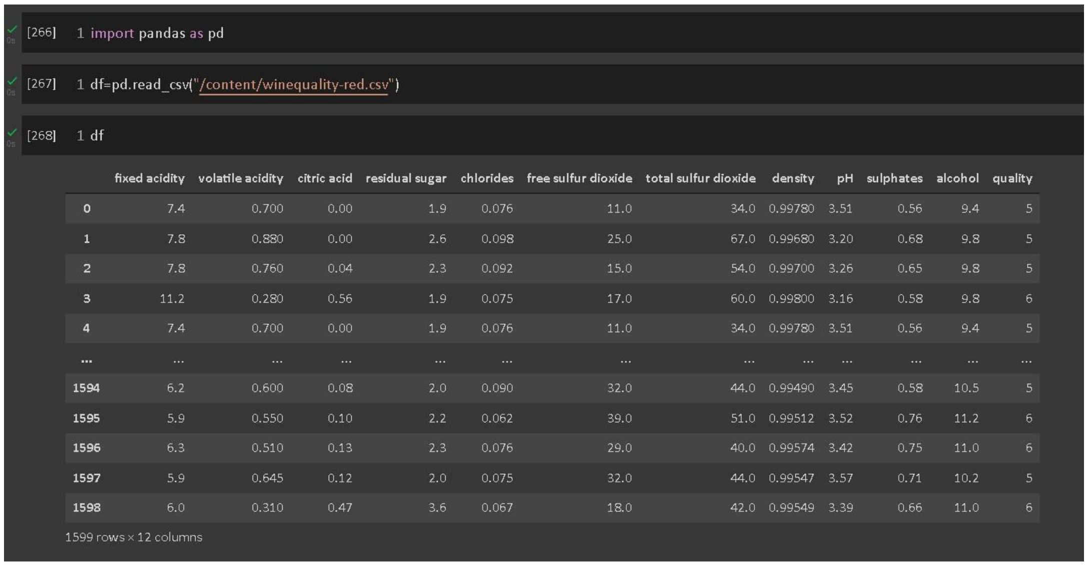
Approach used:

The task is to predict the quality of red wine with the provided dada. I have solved the problem using **Linear Regression**.

Dataset information:

Input variables are fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol. And the output variable (based on sensory data) is quality.

Below is a screenshot of the dataset.



```
[266] import pandas as pd
```

```
[267] df=pd.read_csv("/content/winequality-red.csv")
```

```
[268] df
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

Tools used:

- Google colaboratory
- Numpy Pandas
- Matplotlib
- Seaborn

- Sklearn
- Math

Why linear regression?

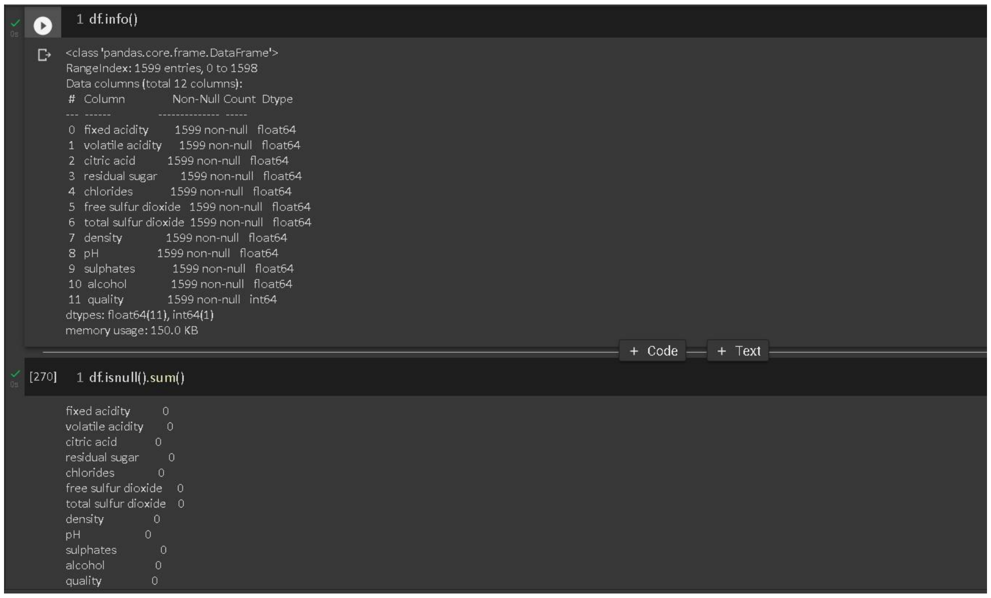
From the data provided, we can see that the values are in a continuous form and the input and output has linear relationship within it so i thought of going with linear regression approach of solving the problem **Algorithm:**

- **Analysing data:**

With the pandas library we read the provided csv data and store it in a variable called df.

We check the data and count the missing values using the functions info() and isnull ()

Below is the screen shot of the data



```
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
```

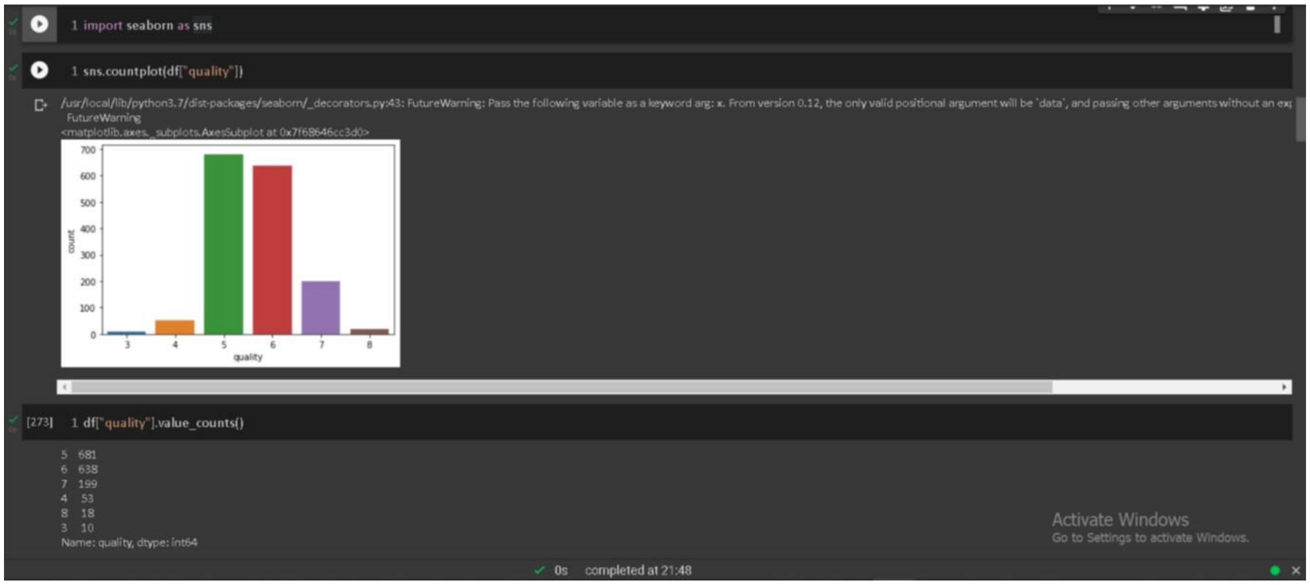
```
[270]  df.isnull().sum()
```

```
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
```

Fortunately we find there are no null values and all datatypes seem to be right.

- **Data Exploration:**

we plot important information that will help us check how features behave and how they are correlated.
Knowing our target variable is **"quality"**, we are now going to plot some information about it. Let's see which values this column contains and how many of them there are.



```
import seaborn as sns
```

```
sns.countplot(df["quality"])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an exp
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f68646cc3d0>
```

```
[273]  df["quality"].value_counts()
```

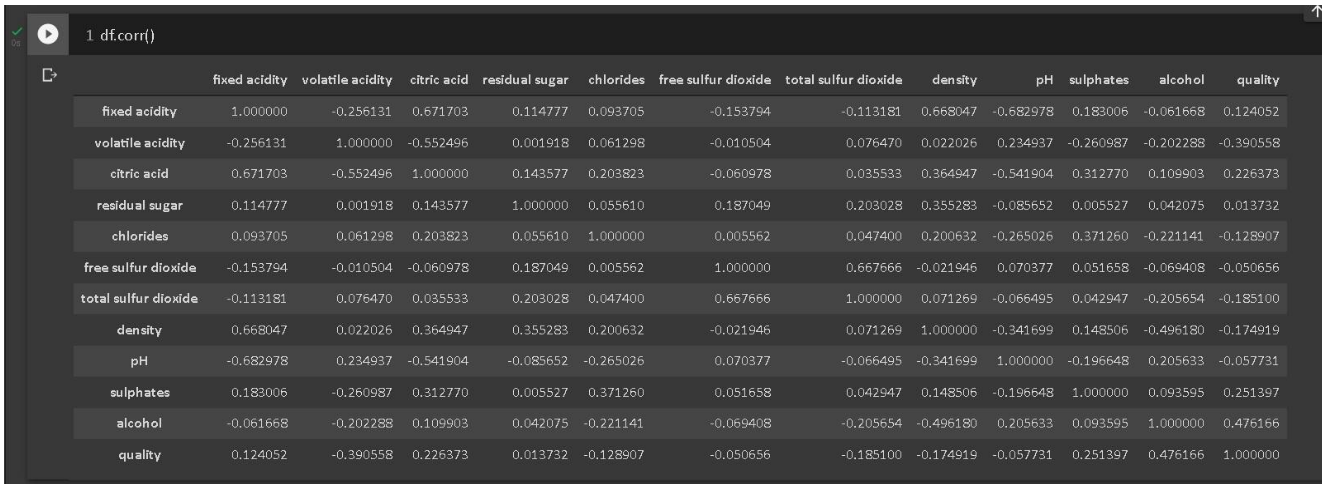
```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

0s completed at 21:48

- **Data cleaning:**

To study the correlation between quality and other features and see which are the ones that play an important role in deciding the quality of a wine we use a function called corr() which gives the relationship between the features and lables and also gives relationship between the respective features

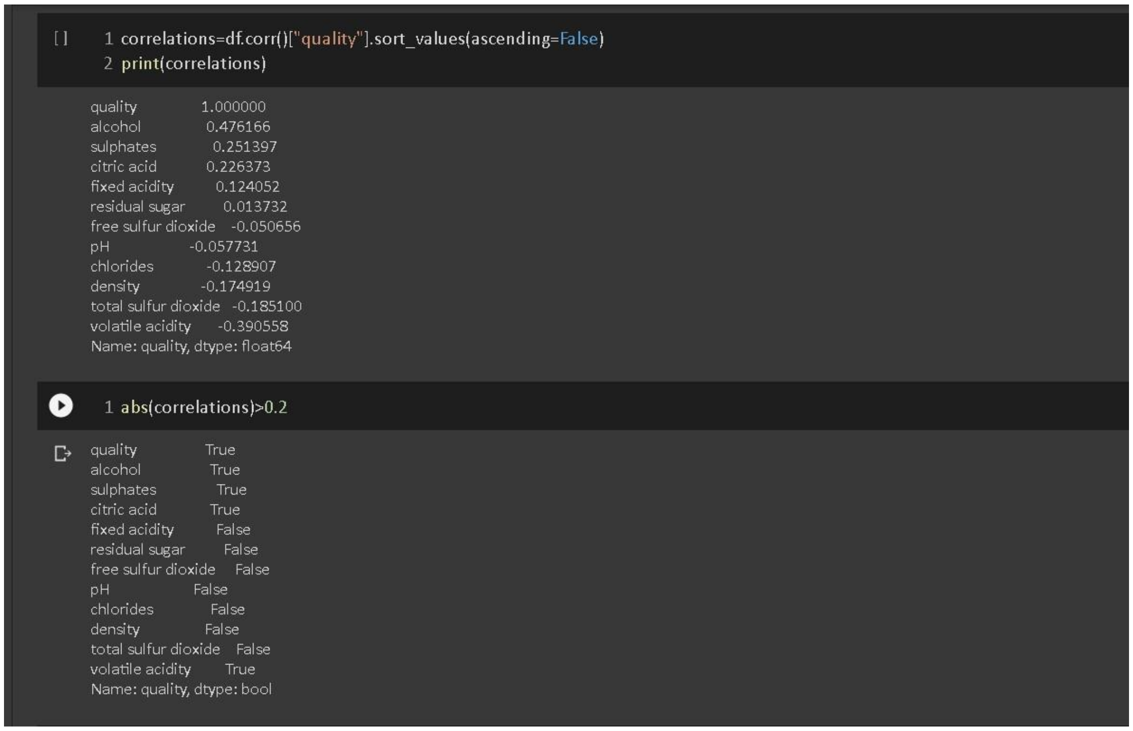
Below is the screenshot which shows the relationship between features and lables



```
df.corr()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181	0.668047	-0.682978	0.183006	-0.061668	0.124052
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.022026	0.234937	-0.260987	-0.202288	-0.390558
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.364947	-0.541904	0.312770	0.109903	0.226373
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028	0.355283	-0.085652	0.005527	0.042075	0.013732
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400	0.200632	-0.265026	0.371260	-0.221141	-0.128907
free sulfur dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666	-0.021946	0.070377	0.051658	-0.069408	-0.050656
total sulfur dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000	0.071269	-0.066495	0.042947	-0.205654	-0.185100
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269	1.000000	-0.341699	0.148506	-0.496180	-0.174919
pH	-0.682978	0.234937	-0.541904	-0.085652	-0.265026	0.070377	-0.066495	-0.341699	1.000000	-0.196648	0.205633	-0.057731
sulphates	0.183006	-0.260987	0.312770	0.005527	0.371260	0.051658	0.042947	0.148506	-0.196648	1.000000	0.093595	0.251397
alcohol	-0.061668	-0.202288	0.109903	0.042075	-0.221141	-0.069408	-0.205654	-0.496180	0.205633	0.093595	1.000000	0.476166
quality	0.124052	-0.390558	0.226373	0.013732	-0.128907	-0.050656	-0.185100	-0.174919	-0.057731	0.251397	0.476166	1.000000

After analysing the correlation data i selected the ones with bigger numbers as correlation with the quality since these are the ones that will give us more information and we considered a minimum threshold of correlation approximately around 0.2 (absolut value) since we do not have to take into account features whose values might be redundant and not provide information at all



```
correlations=df.corr()["quality"].sort_values(ascending=False)
print(correlations)
```

```
quality                 1.000000
alcohol                 0.476166
sulphates               0.251397
citric acid             0.226373
fixed acidity           0.124052
residual sugar          0.013732
free sulfur dioxide    -0.050656
pH                     -0.057731
chlorides              -0.128907
density                -0.174919
total sulfur dioxide   -0.185100
volatile acidity       -0.390558
Name: quality, dtype: float64
```

```
abs(correlations)>0.2
```

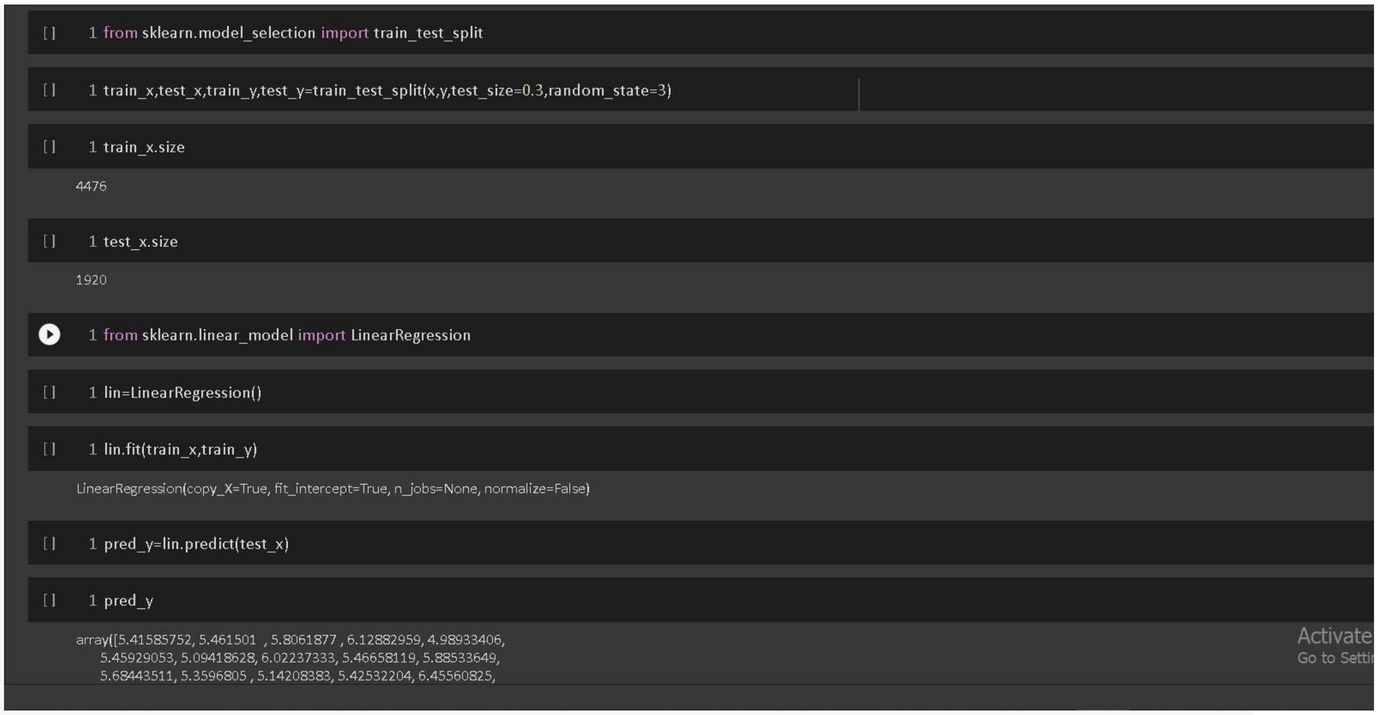
```
quality                  True
alcohol                  True
sulphates                True
citric acid              True
fixed acidity           False
residual sugar          False
free sulfur dioxide     False
pH                      False
chlorides               False
density                 False
total sulfur dioxide    False
volatile acidity         True
Name: quality, dtype: bool
```

From all the values, we are selecting **alcohol**, **sulphates**, **citric_acid** and **volatile_acidity**

- ## Train test split the data:

On this section, after having understood our data and dropped some useless features, we are going to make an estimation of quality based on the other features. To do so we are going to use Linear Regression

We separate our features from our target feature (quality) and we split data into training and test



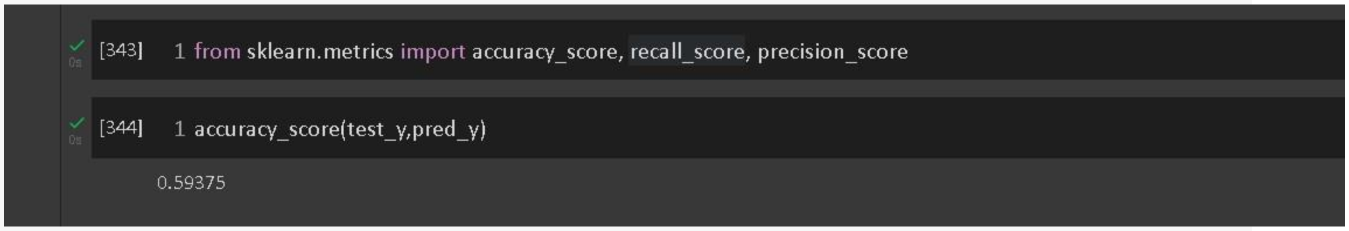
```
[ ]  from sklearn.model_selection import train_test_split

[ ]  train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=0.3,random_state=3)

[ ]  train_x.size

     4476

[ ]  test_x.size

     1920

     from sklearn.linear_model import LinearRegression

[ ]  lin=LinearRegression()

[ ]  lin.fit(train_x,train_y)

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

[ ]  pred_y=lin.predict(test_x)

[ ]  pred_y

     array([5.41585752, 5.461501  , 5.8061877 , 6.12882959, 4.98933406,
            5.45929053, 5.09418628, 6.02237333, 5.46658119, 5.88533649,
            5.68443511, 5.3596805 , 5.14208383, 5.42532204, 6.45560825,
```

Activate
Go to Setti

After splitting the data into train and test, we import linear regression library and fit our training data.

Now we predict the result (pred_y) with the function predict () by passing in the test x data

Now with accuracy_score() function we can find the accuracy of our model

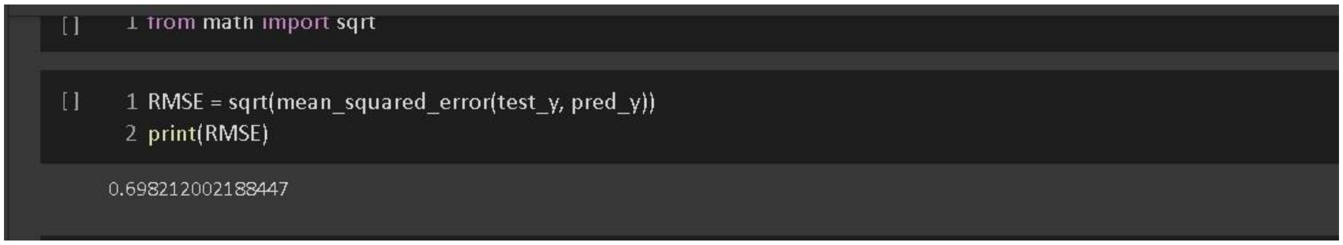


```
[343]  from sklearn.metrics import accuracy_score, recall_score, precision_score

[344]  accuracy_score(test_y,pred_y)

       0.59375
```

Our model is now **59.3% accurate**

- ## RMSE of Models:

After having prepared our models, it's now time to evaluate them. To do so we are going to use RMSE (Root Mean Square Error) which is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are so RMSE is a measure of how spread out these residuals are.

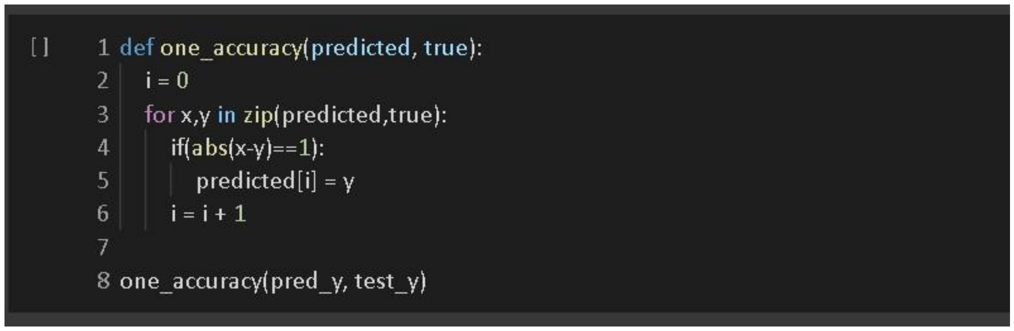


```
[ ]    from math import sqrt

[ ]    RMSE = sqrt(mean_squared_error(test_y, pred_y))
       print(RMSE)

    0.698212002188447
```

We get an **error of 69.8%**

- **Improving the results with 1-Off Accuracy:**

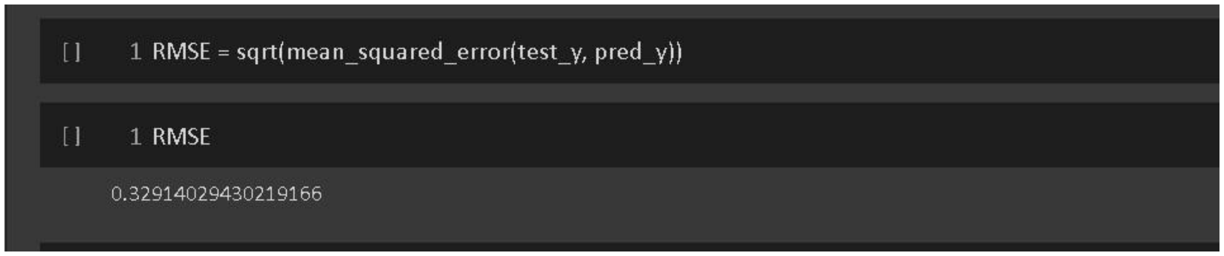
As we can above our predictions aren't bad at all but in order to "improve" them we are going to apply a concept called 1-off accuracy, which states that if the distance between our predicted quality and the true quality is 1 (in absolut value), we will accept it as a correct prediction.
We will now create a function that will transform our predicted value into the true value if the distance between them is equal to 1. Afterwards we are going to plot the new correlation matrices and test the new values with some metrics.



```
[ ]    def one_accuracy(predicted, true):
         i = 0
         for x,y in zip(predicted,true):
           if(abs(x-y)==1):
             predicted[i] = y
           i = i + 1

       one_accuracy(pred_y, test_y)
```

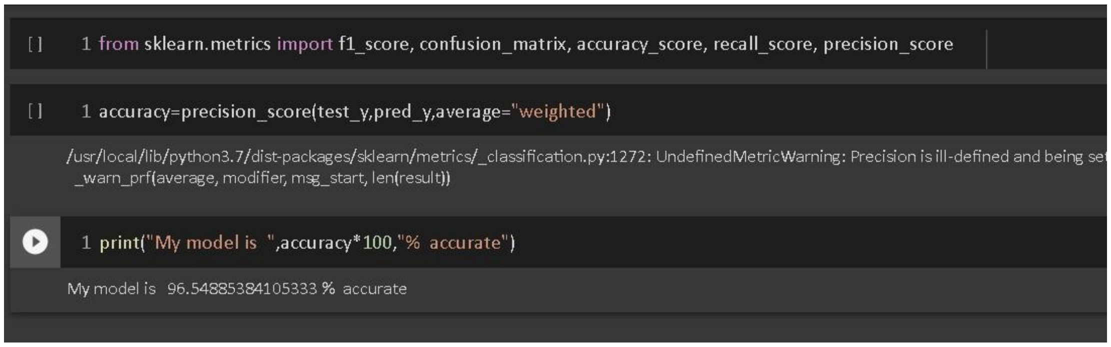
1 off accuracy decreases the error and increase the accuracy of our model

RMSE after 1 off accuracy is shown below



```
[ ]    RMSE = sqrt(mean_squared_error(test_y, pred_y))

[ ]    RMSE

    0.32914029430219166
```

Accuracy after 1 off accuracy is shown below


```
[ ]  from sklearn.metrics import f1_score, confusion_matrix, accuracy_score, recall_score, precision_score

[ ]  accuracy=precision_score(test_y,pred_y,average="weighted")

     /usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision is ill-defined and being se
       _warn_prf(average, modifier, msg_start, len(result))

     print("My model is  ",accuracy*100,"% accurate")

     My model is  96.54885384105333 % accurate
```

After using 1 off accuracy we are getting an

RMSE error of 32.9%

Accuracy of 96.5%

- **Conclusions:**

After having obtained all the results through our models and plots, these are some things we can say about this problem and solution:

- The vast majority of wines get a quality rating of five or six, while having good and bad wines seems more unlikely. There seem not to be any excellent wines (>8) on this database.
- From the very first moment we saw there weren't strong correlations between features and quality, that's why it's hard to make an accurate prediction using regression algorithms. That said, alcohol, sulphates, citric_acid features are the ones that correlate the most positively while volatile_acidity is the one correlating the most negatively.
- Applying the concept 1-off Accuracy gives us much better results.
- Linear Regression seems to be the best fitting models when solving this problem using regression.
- Since there are only six different quality values in this dataset, it would be clever treating this problem as a multiclass classification and we might even get better results.
- **Model Accuracy : 96.5%**