Assignment 2

LOGISTIC REGRESSION

Submitted By: Neel Vijay Patil (11436004)

Submitted To: Dr. Junhua Ding

INFO 5505

Applied Machine Learning for Data Science

UNT

I downloaded the dataset (Red Wine Quality) from the Kaggle. The main goal of project is designing as well as evaluating the Logistic Regression Algorithm/Model which helps to decide the quality of the wine whether it is (high/low) quality based on other attributes of the dataset. I have evaluated the test score as well.

The features that affect the wine Quality are chlorides, total Sulphur dioxide, free Sulphur dioxide, PH, density, citric acid, volatile acidity, fixed acidity, residual sugar, alcohol.

1)Importing the required libraries

```
#Importing libraries
import pandas as pd
import missingno as mn
import seaborn as sb
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
from scipy.stats import skew
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
```

2) Importing the wine quality-red dataset

```
#importing data
from google.colab import files
uploaded = files.upload()
```

```
Choose Files winequality-red.csv
• winequality-red.csv(text/csv) - 100951 bytes, last modified: 7/16/2022 - 100% done
Saving winequality-red.csv to winequality-red.csv
```

3) Reading dataset and creating the data frames

```
[3] data = pd.read_csv('winequality-red.csv')
```

```
data
```

	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

The dataset has 1599 observations and 12 different features that deals with wine quality.

```
[5] data.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
```

The dataset has float64, int64 datatypes and used around 150 kb of memory the wine quality dataset does not have any NULL values.

```
# Checking for missing values in the data
data.isnull()
```

	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
1	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
3	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
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	False	False	False	False	False	False	False	False	False	False	False	False
1595	False	False	False	False	False	False	False	False	False	False	False	False
1596	False	False	False	False	False	False	False	False	False	False	False	False
1597	False	False	False	False	False	False	False	False	False	False	False	False
1598	False	False	False	False	False	False	False	False	False	False	False	False

1599 rows × 12 columns

4) Checking the null values using is null function.

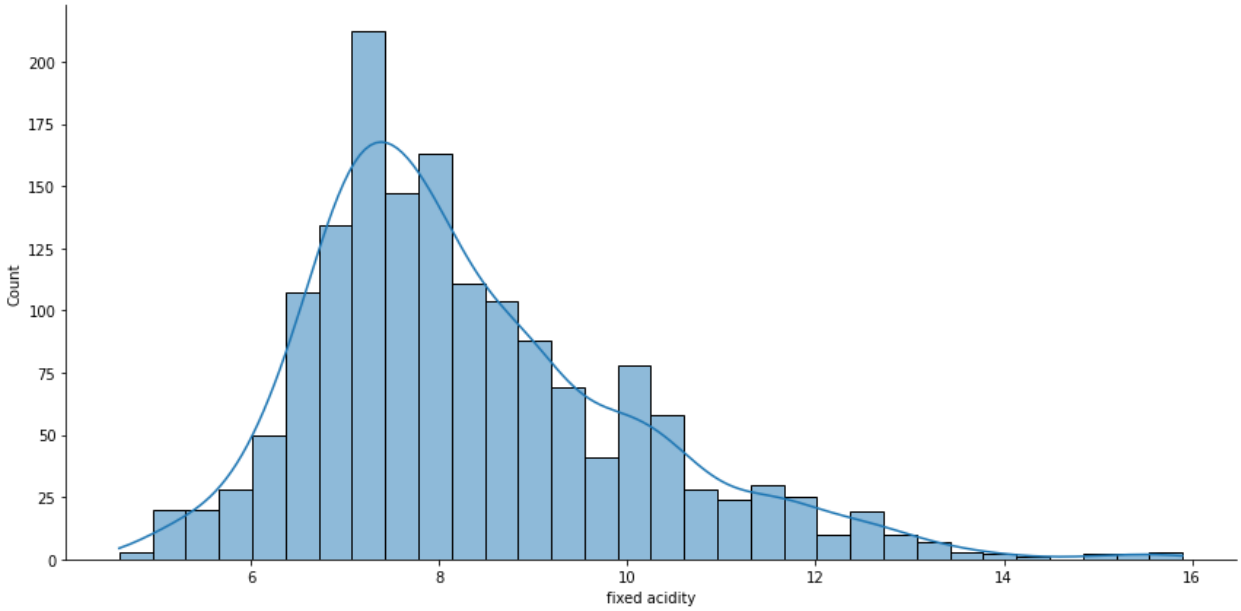
```
data.isnull().sum()
```

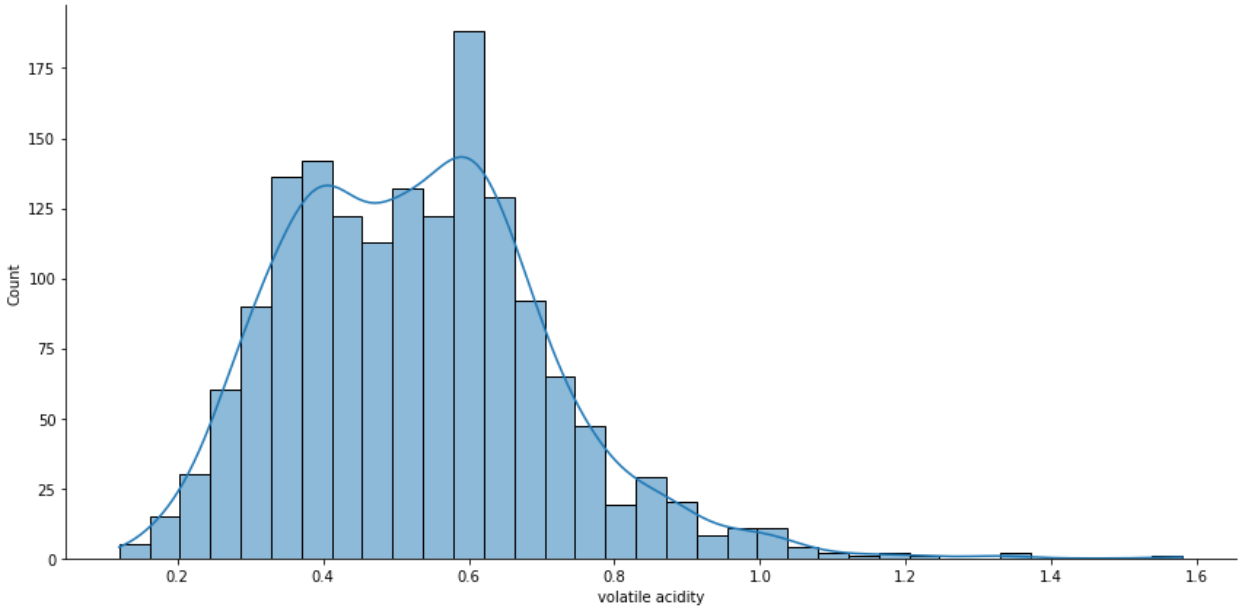
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

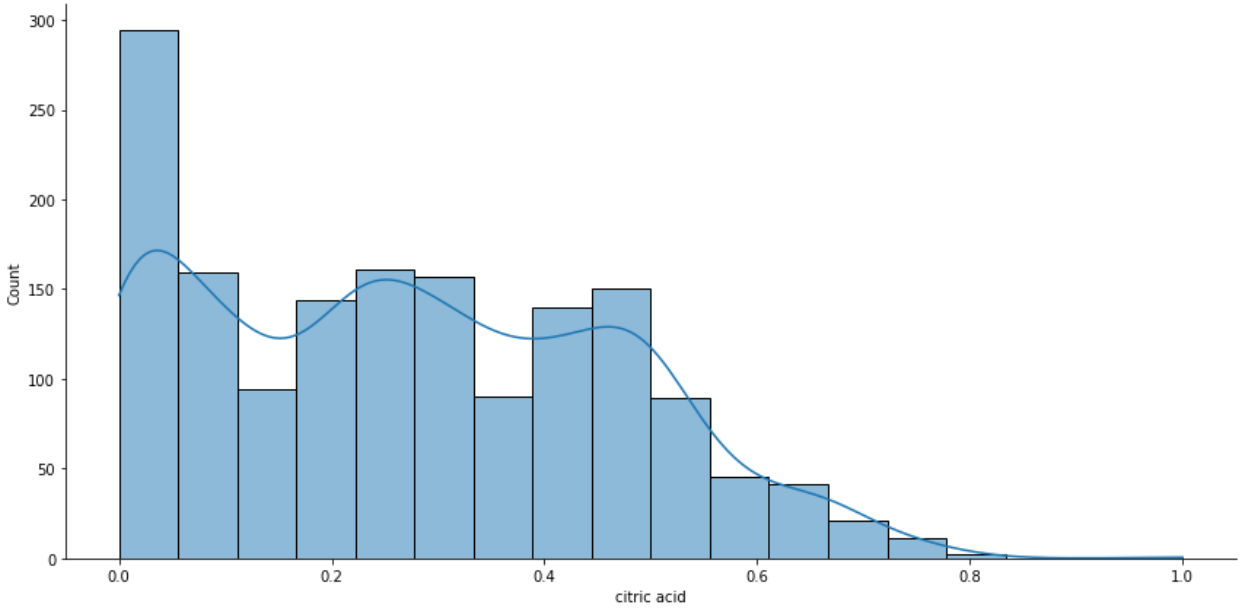
5) Checking whether the features are normally distributed or skewed in nature by plotting the displot.

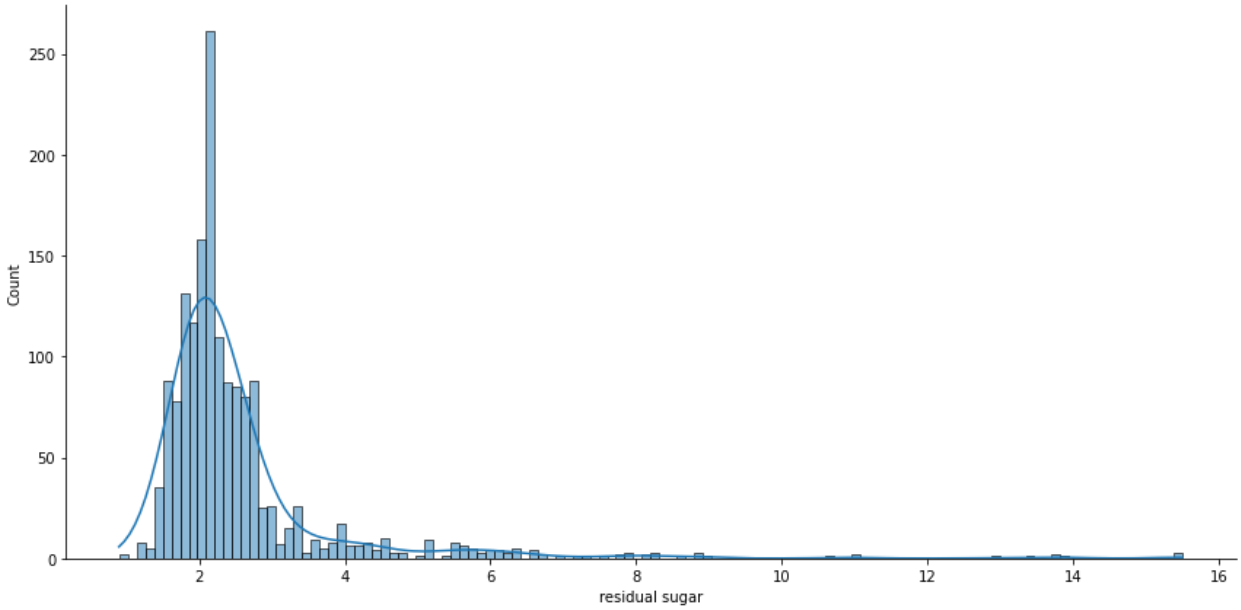
```
# Checking the distrubution of variables

# fig, ax = plt.subplots(figsize=(10,10))
sb.displot(data, x = 'fixed acidity', kde=True, height=6, aspect=2)
sb.displot(data, x = 'volatile acidity', kde=True, height=6, aspect=2)
sb.displot(data, x = 'citric acid', kde=True, height=6, aspect=2)
sb.displot(data, x = 'residual sugar', kde=True, height=6, aspect=2)
sb.displot(data, x = 'free sulfur dioxide', kde=True, height=6, aspect=2)
sb.displot(data, x = 'total sulfur dioxide', kde=True, height=6, aspect=2)
sb.displot(data, x = 'density', kde=True, height=6, aspect=2)
sb.displot(data, x = 'pH', kde=True, height=6, aspect=2)
sb.displot(data, x = 'sulphates', kde=True, height=6, aspect=2)
sb.displot(data, x = 'alcohol', kde=True, height=6, aspect=2)
sb.displot(data, x = 'quality', kde=True, height=6, aspect=2)
sb.displot(data, x = 'chlorides', kde=True, height=6, aspect=2)
```

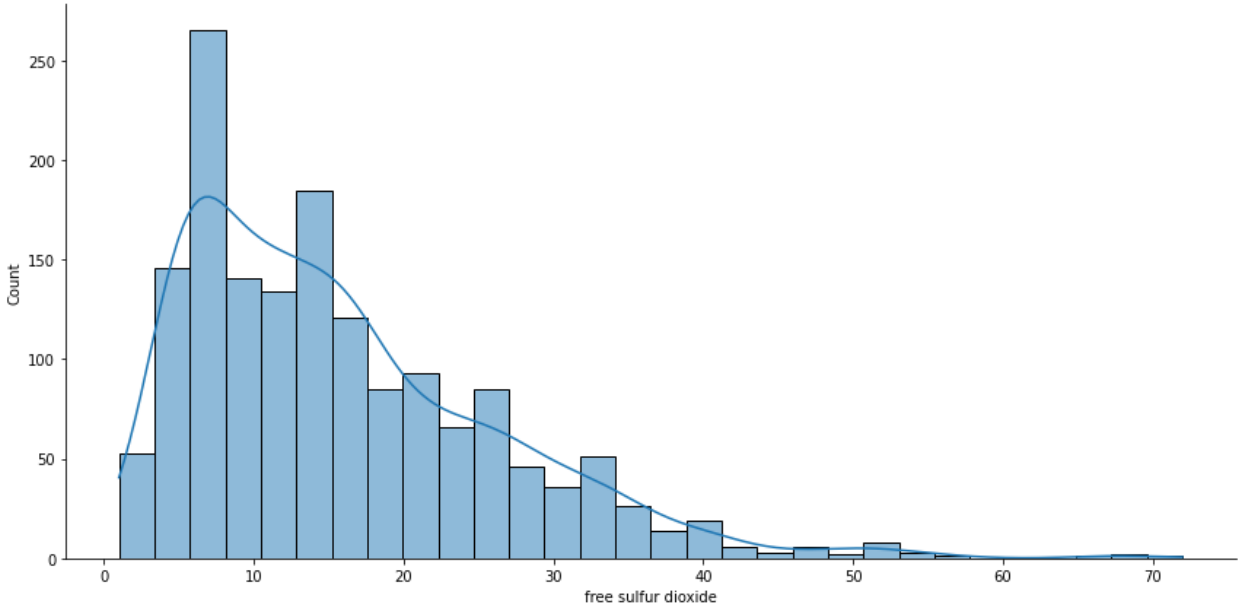
```
<seaborn.axisgrid.FacetGrid at 0x7fb845012f90>
```

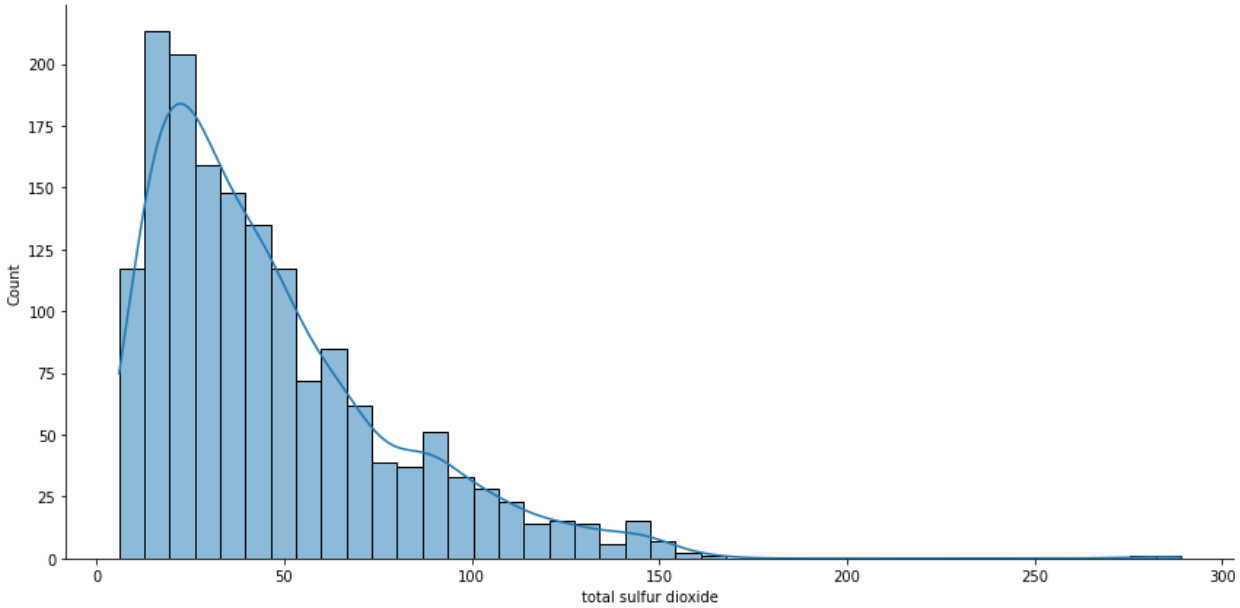


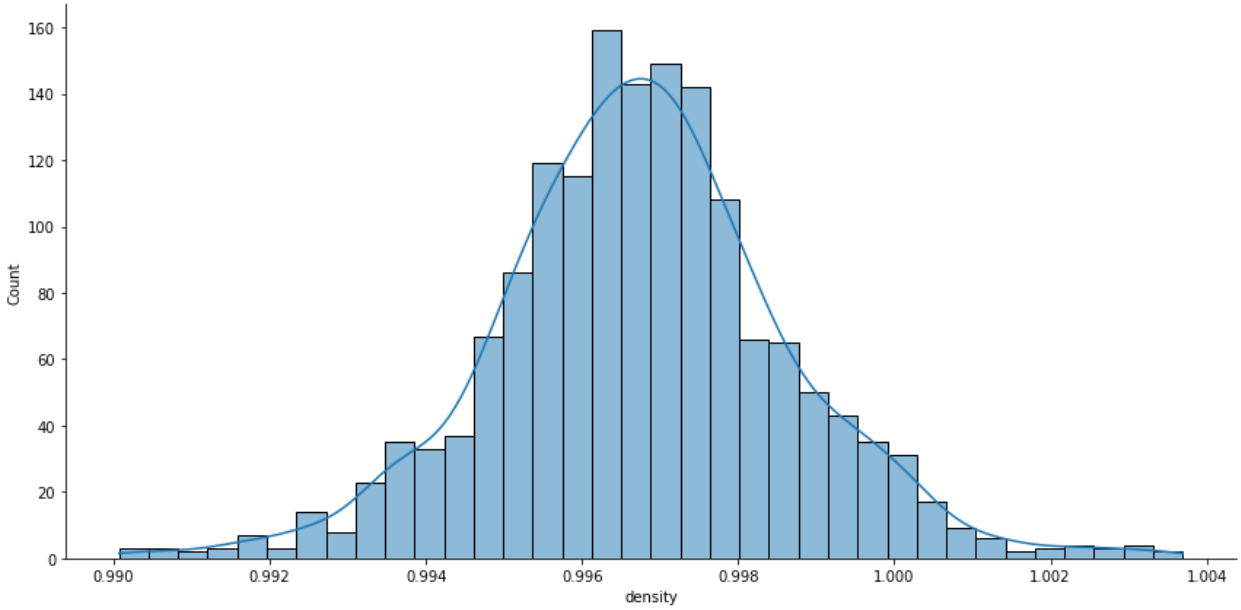


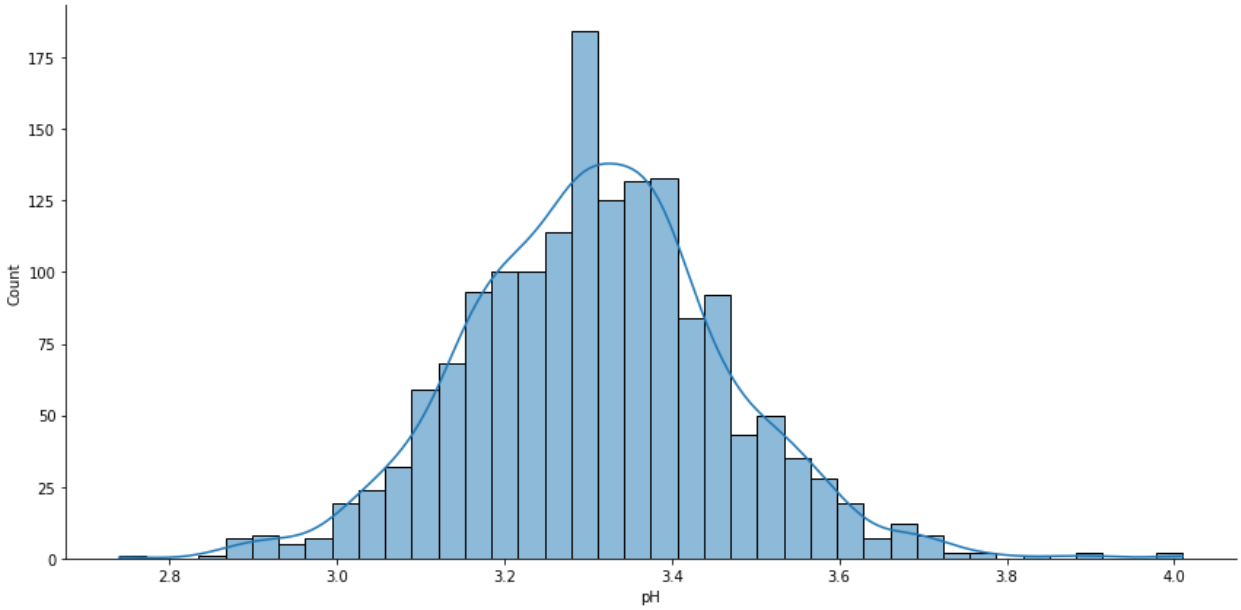

Count
citric acid

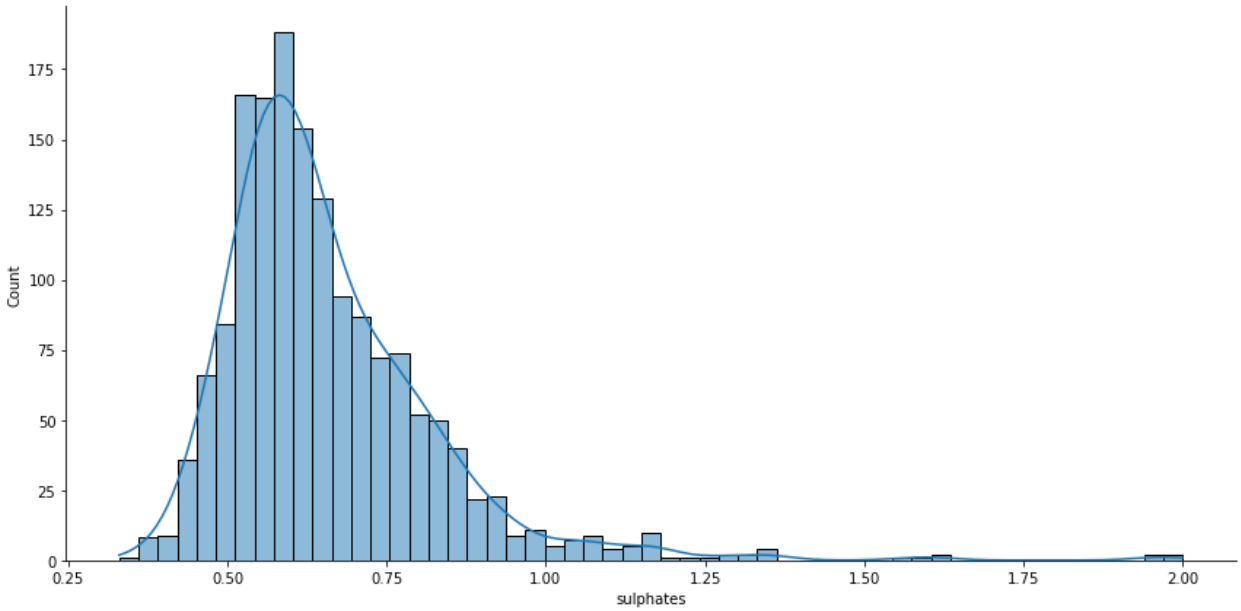

Count
residual sugar

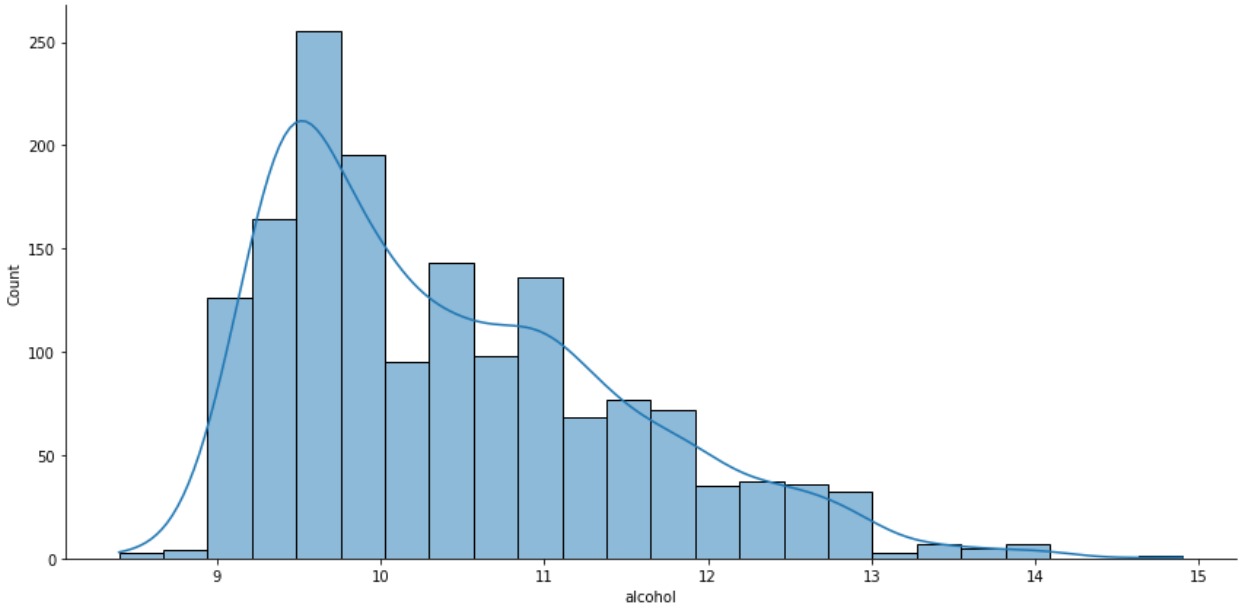

Count
250
200
150
100
50
0
0
10
20
30
40
50
60
70
free sulfur dioxide

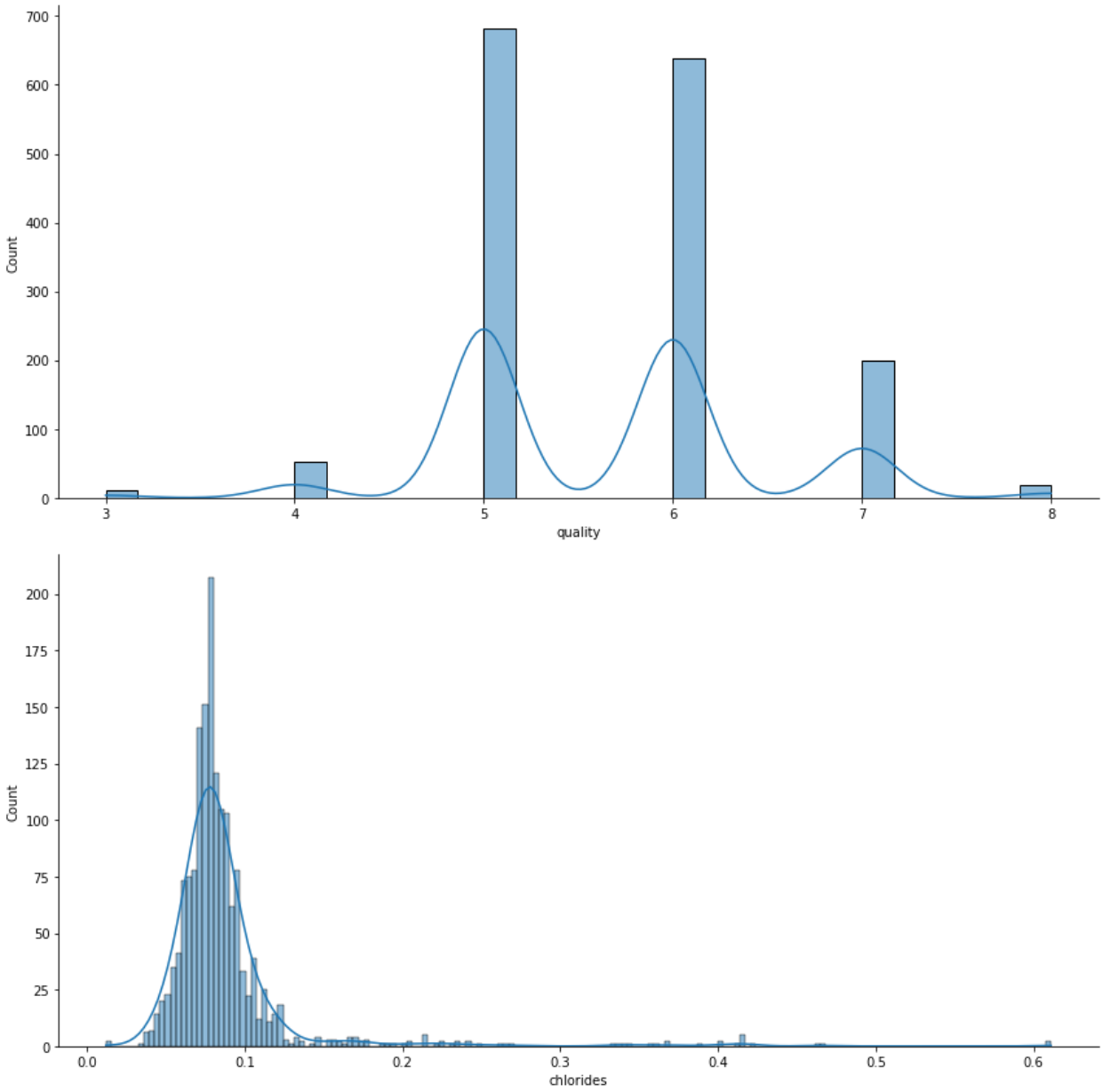

Count
200
175
150
125
100
75
50
25
0
0
50
100
150
200
250
300
total sulfur dioxide


Count
density
0
20
40
60
80
100
120
140
160
0.990
0.992
0.994
0.996
0.998
1.000
1.002
1.004


Count
pH
0
25
50
75
100
125
150
175
2.8
3.0
3.2
3.4
3.6
3.8
4.0


Count
sulphates


Count
alcohol



The above displots states that chlorides and sulphates are right skewed while most of the other features are normally distributed.

The skewness coefficient for the Chlorides is found to be 5.67 which indicates it to be rightly skewed as well as sulphates have skewness coefficient of 2.42 which indicates it to be slightly right skewed.

```
# Finding Skewness of the variables

for column in data.columns:
  if data.dtypes[column] != np.object:
    print(column ,' : ' , skew(data[column], axis=0, bias=True, nan_policy='omit'))
```

```
fixed acidity  :  0.9818292953262073
volatile acidity  :  0.6709623963499574
citric acid  :  0.3180385895475358
residual sugar  :  4.536394788805638
chlorides  :  5.675016527504258
free sulfur dioxide  :  1.249393847434253
total sulfur dioxide  :  1.5141091878506638
density  :  0.07122077153539946
pH  :  0.19350175891005525
sulphates  :  2.426393455449087
alcohol  :  0.8600210646566755
quality  :  0.21759720553467285
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To silence this warning, use `object` by itself.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  after removing the cwd from sys.path.
```

6) Log Transformation of Chloride and Sulphur features

I carried log transformation to convert the skewed data into normalize form and concatenated it to the original data frame using concat().

```
# log transform chlorides column
chlorides_log = data[['chlorides']].applymap(lambda x: np.log(x))
chlorides_log.columns = 'log_' + chlorides_log.columns
```

```
[12] chlorides_log.skew()
```

```
log_chlorides    1.745215
dtype: float64
```

```
# log transform sulphates column
sulphates_log = data[['sulphates']].applymap(lambda x: np.log(x))
sulphates_log.columns = 'log_' + sulphates_log.columns
```

```
[14] sulphates_log.skew()
```

```
log_sulphates    0.921867
dtype: float64
```

```
[15] df = pd.concat([data, chlorides_log,sulphates_log], axis=1)
```

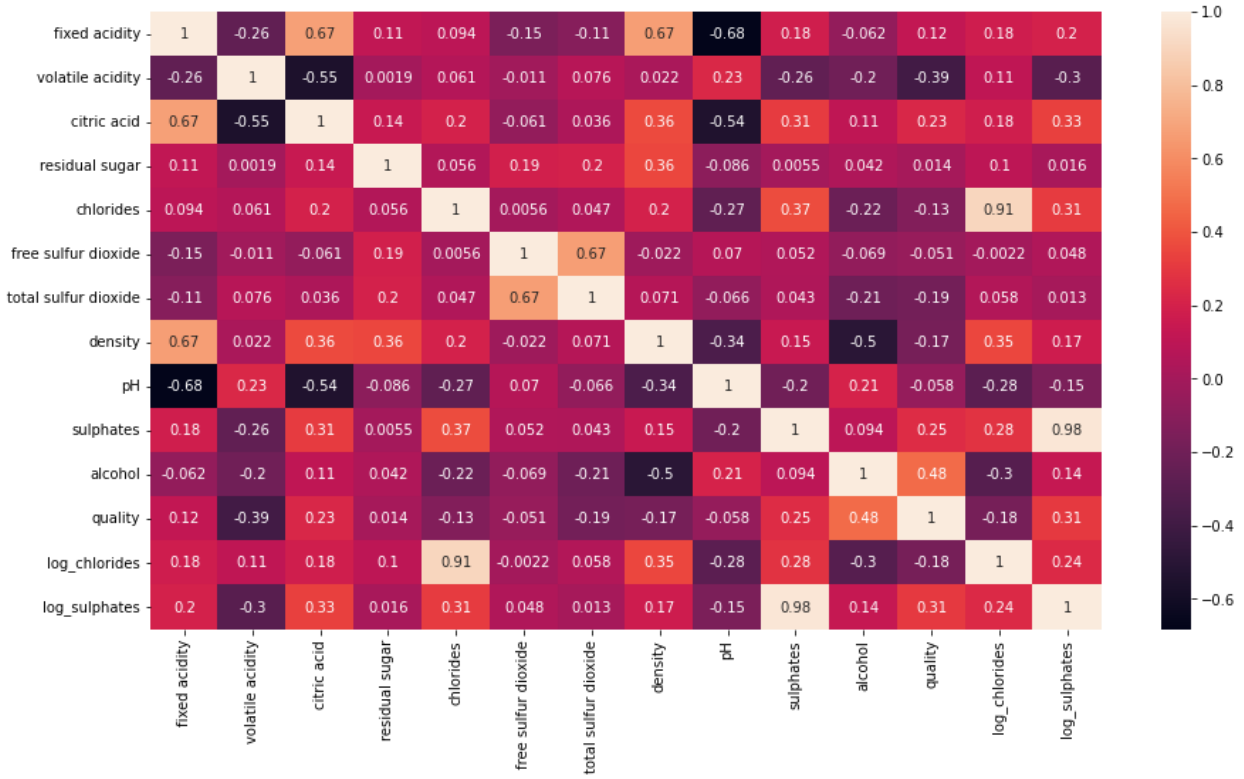
```
[16] df
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	log_chlorides	log_sulphates
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	-2.577022	-0.579818
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5	-2.322788	-0.385662
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5	-2.385967	-0.430783
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6	-2.590267	-0.544727
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	-2.577022	-0.579818
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5	-2.407946	-0.544727
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6	-2.780621	-0.274437
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6	-2.577022	-0.287682
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5	-2.590267	-0.342490
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6	-2.703063	-0.415515

1599 rows × 14 columns

6) Correlation Matrix between variables

```
# Finding correlation between variables
fig, ax = plt.subplots(figsize=(15,8))
sb.heatmap(df.corr(), annot=True)
```



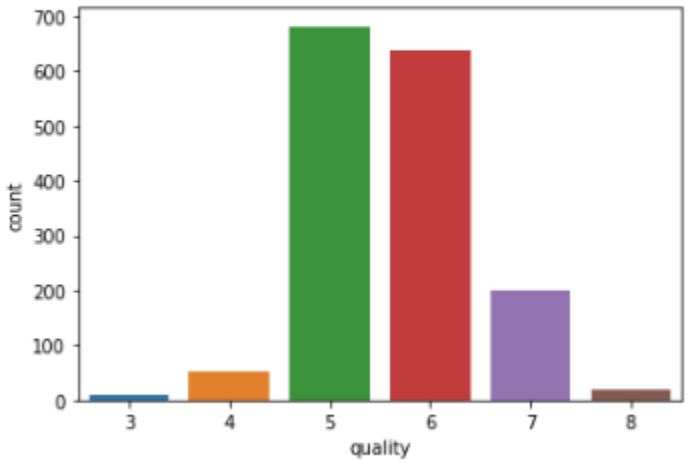
The above heatmap demonstrates that the features Quality and Alcohol are highly correlated to each other which accounts (0.48) as well as density and chlorides are correlated with (0.35) correlation coefficient. Off course the features and their log values are highly correlated.

7) Data preprocessing

The quality of wine is ranged from 2 to 8 the count plot below depicts the count of respective categories.

```
[18] # Count of each category of wine quality
     sb.countplot(x="quality", data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb84491d110>
```



However, I am developing a predictive model for wine quality which distinguish into 2 categories low, high quality. I have transformed the categories into the binary formats that is in 0,1.

Here

1's are assigned to values that are greater than 7.

0's are assigned to remaining values.

```
# Preprocessing the data

df['quality'] = [1 if x>=7 else 0 for x in df['quality']]
df
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	log_chlorides	log_sulphates
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	0	-2.577022	-0.579818
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	0	-2.322788	-0.385662
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	0	-2.385967	-0.430783
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	0	-2.590267	-0.544727
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	0	-2.577022	-0.579818
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	0	-2.407946	-0.544727
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	0	-2.780621	-0.274437
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	0	-2.577022	-0.287682
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	0	-2.590267	-0.342490
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	0	-2.703063	-0.415515

1599 rows × 14 columns

8) Designing the logistic regression model

```
# Developing Logistic Regression model
X = df.drop(labels='quality', axis=1).values
y = df['quality'].values
```

```
[21] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

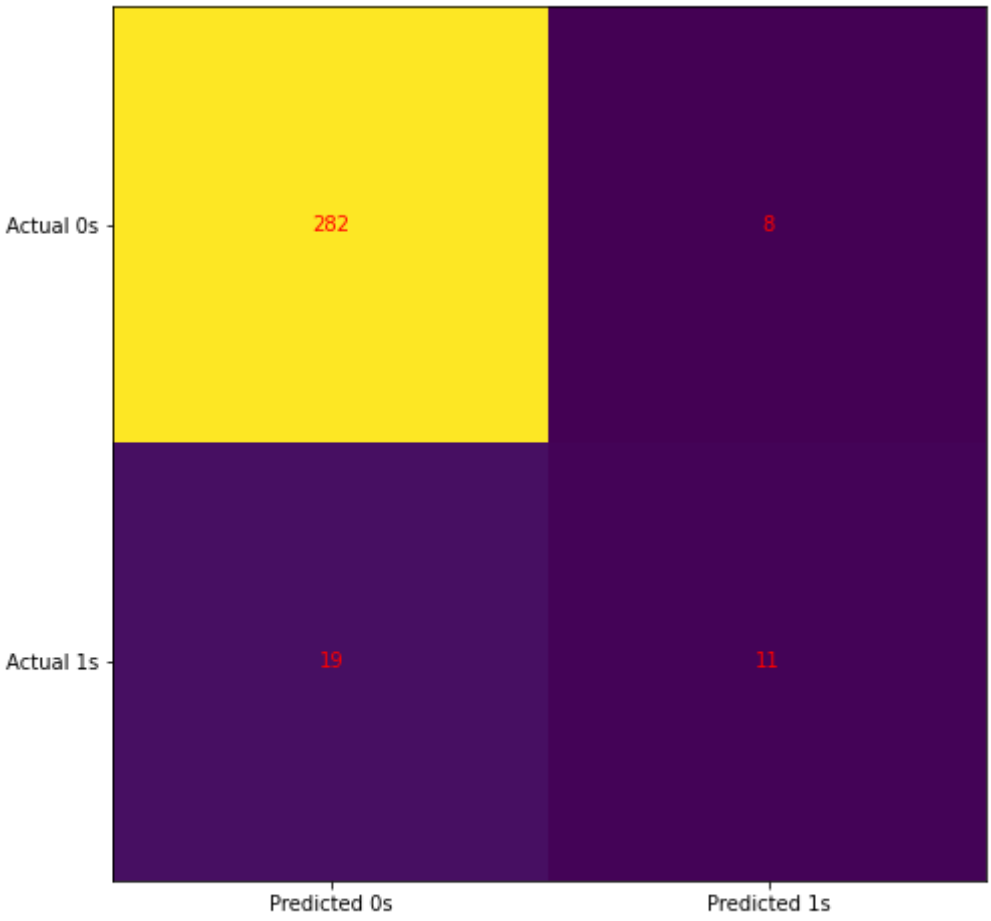
```
[22] logitmodel = LogisticRegression(solver='liblinear', random_state=0)
```

```
[23] logitmodel.fit(X_train, y_train)

     LogisticRegression(random_state=0, solver='liblinear')
```

```
[24] y_pred = logitmodel.predict(X_test)
```

9)Evaluating the test score using the confusion Matrix

```
[25] # Finding test scores using confusion matrix
     cm = confusion_matrix(y_test, y_pred)

     fig, ax = plt.subplots(figsize=(8, 8))
     ax.imshow(cm)
     ax.grid(False)
     ax.xaxis.set(ticks=(0, 1), ticklabels=('Predicted 0s', 'Predicted 1s'))
     ax.yaxis.set(ticks=(0, 1), ticklabels=('Actual 0s', 'Actual 1s'))
     ax.set_ylim(1.5, -0.5)
     for i in range(2):
         for j in range(2):
             ax.text(j, i, cm[i, j], ha='center', va='center', color='red')
     plt.show()
```



The confusion matrix illustrates that test data has
Low quality wine = 300

High quality wine = 8

Prediction of Logistic Regression Model:

1) **The logistic regression model has predicted 8 - high Quality wine out of 300- lowest quality**
2) **19 has been predicted as low quality out of the 30- high quality.**

10) Classification report

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.97      0.95       290
           1       0.58      0.37      0.45        30

    accuracy                           0.92       320
   macro avg       0.76      0.67      0.70       320
weighted avg       0.90      0.92      0.91       320
```

11)Accuracy Score

The count of correctly predicted models on the scale of 100 is referred as accuracy of that model.

Here, The Accuracy score was found to be 91.56 percentage. So we can say that out of 100 our model has predicted 91.56 times correctly.

```
# Accuracy_Score
logitmodel.score(X_test, y_test)
```

```
0.915625
```

```
[28] from sklearn import  metrics
```

```
print(metrics.confusion_matrix(y_test, y_pred, labels=[0,1]))
```

```
[[282   8]
 [ 19  11]]
```

12)Conclusion

I have designed a logistic regression model with 91.56% accuracy and also have better precision, recall, f-1, support coefficients.