ML LAB 2

Explore and implement Linear regression algorithm in a given business scenario and comment on its efficiency and performance.

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
%matplotlib inline
```

In [5]:
```
df=pd.read_csv("E:\DS\Datasets\winequalityN.csv")
```

In [7]:
```
df.head(20)
```

Out[7]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	a
0	white	7.0	0.27	0.36	20.70	0.045	45.0	170.0	1.0010	3.00	0.45	
1	white	6.3	0.30	0.34	1.60	0.049	14.0	132.0	0.9940	3.30	0.49	
2	white	8.1	0.28	0.40	6.90	0.050	30.0	97.0	0.9951	3.26	0.44	
3	white	7.2	0.23	0.32	8.50	0.058	47.0	186.0	0.9956	3.19	0.40	
4	white	7.2	0.23	0.32	8.50	0.058	47.0	186.0	0.9956	3.19	0.40	
5	white	8.1	0.28	0.40	6.90	0.050	30.0	97.0	0.9951	3.26	0.44	
6	white	6.2	0.32	0.16	7.00	0.045	30.0	136.0	0.9949	3.18	0.47	
7	white	7.0	0.27	0.36	20.70	0.045	45.0	170.0	1.0010	3.00	0.45	
8	white	6.3	0.30	0.34	1.60	0.049	14.0	132.0	0.9940	3.30	0.49	
9	white	8.1	0.22	0.43	1.50	0.044	28.0	129.0	0.9938	3.22	0.45	
10	white	8.1	0.27	0.41	1.45	0.033	11.0	63.0	0.9908	2.99	0.56	
11	white	8.6	0.23	0.40	4.20	0.035	17.0	109.0	0.9947	3.14	0.53	
12	white	7.9	0.18	0.37	1.20	0.040	16.0	75.0	0.9920	3.18	0.63	
13	white	6.6	0.16	0.40	1.50	0.044	48.0	143.0	0.9912	3.54	0.52	
14	white	8.3	0.42	0.62	19.25	0.040	41.0	172.0	1.0002	2.98	0.67	
15	white	6.6	0.17	0.38	1.50	0.032	28.0	112.0	0.9914	3.25	0.55	
16	white	6.3	0.48	0.04	1.10	0.046	30.0	99.0	0.9928	3.24	0.36	
17	white	NaN	0.66	0.48	1.20	0.029	29.0	75.0	0.9892	3.33	0.39	
18	white	7.4	0.34	0.42	1.10	0.033	17.0	171.0	0.9917	3.12	0.53	
19	white	6.5	0.31	0.14	7.50	0.044	34.0	133.0	0.9955	3.22	0.50	

In [12]:
```
df.columns
```

Out[12]:
```
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 [13]:
```
df.shape
```

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

In [14]:
```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
type                    6497 non-null object
fixed acidity           6487 non-null float64
volatile acidity        6489 non-null float64
citric acid             6494 non-null float64
residual sugar          6495 non-null float64
chlorides               6495 non-null float64
free sulfur dioxide     6497 non-null float64
total sulfur dioxide    6497 non-null float64
density                 6497 non-null float64
pH                      6488 non-null float64
sulphates               6493 non-null float64
alcohol                 6497 non-null float64
quality                 6497 non-null int64
dtypes: float64(11), int64(1), object(1)
memory usage: 659.9+ KB
None
```

In [15]:
```
df.isna().sum()
```

Out[15]:
```
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
```

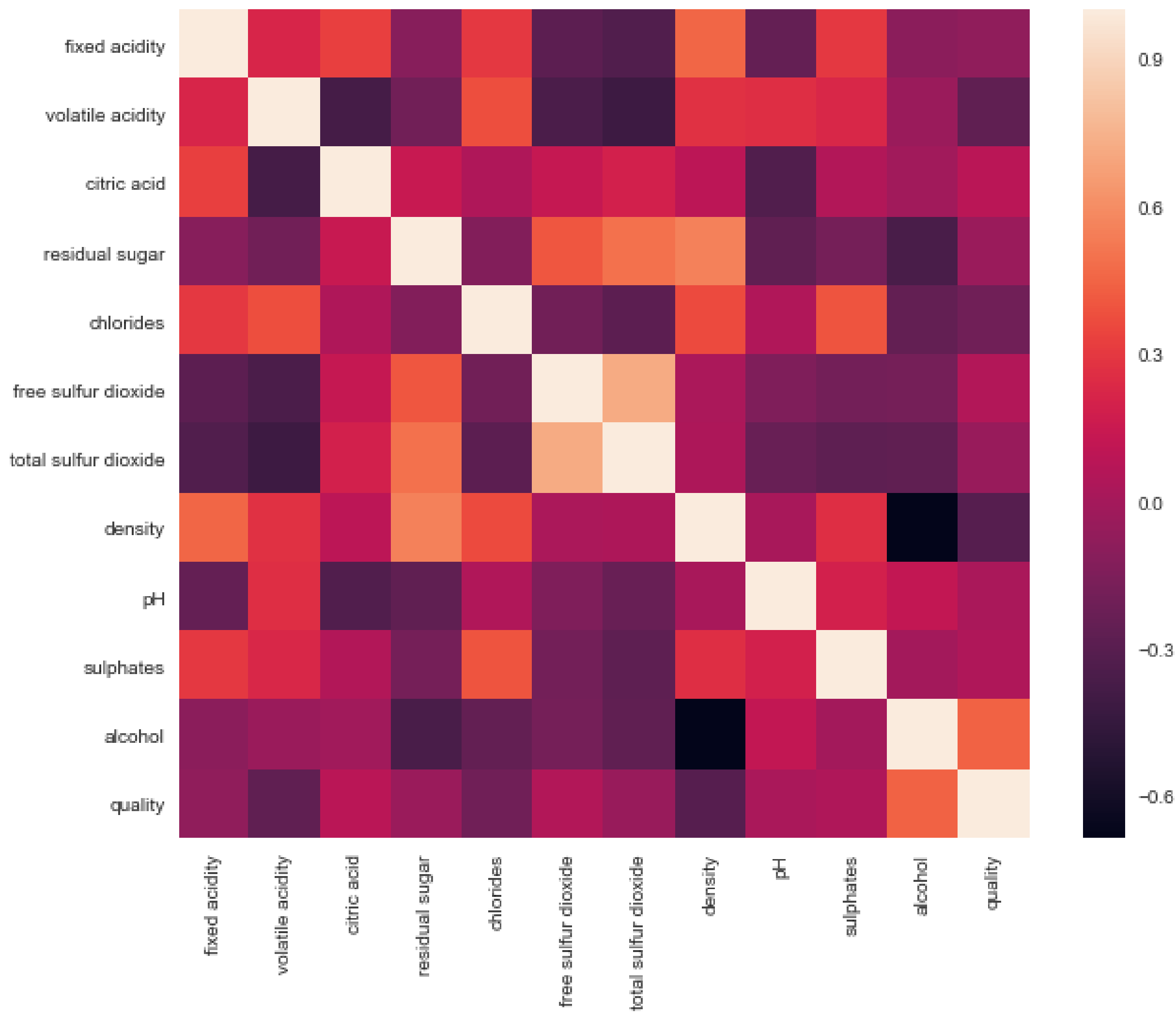
In [16]:
```
df=df.fillna(df.mean())
```

In [17]:
```
df.describe()
```

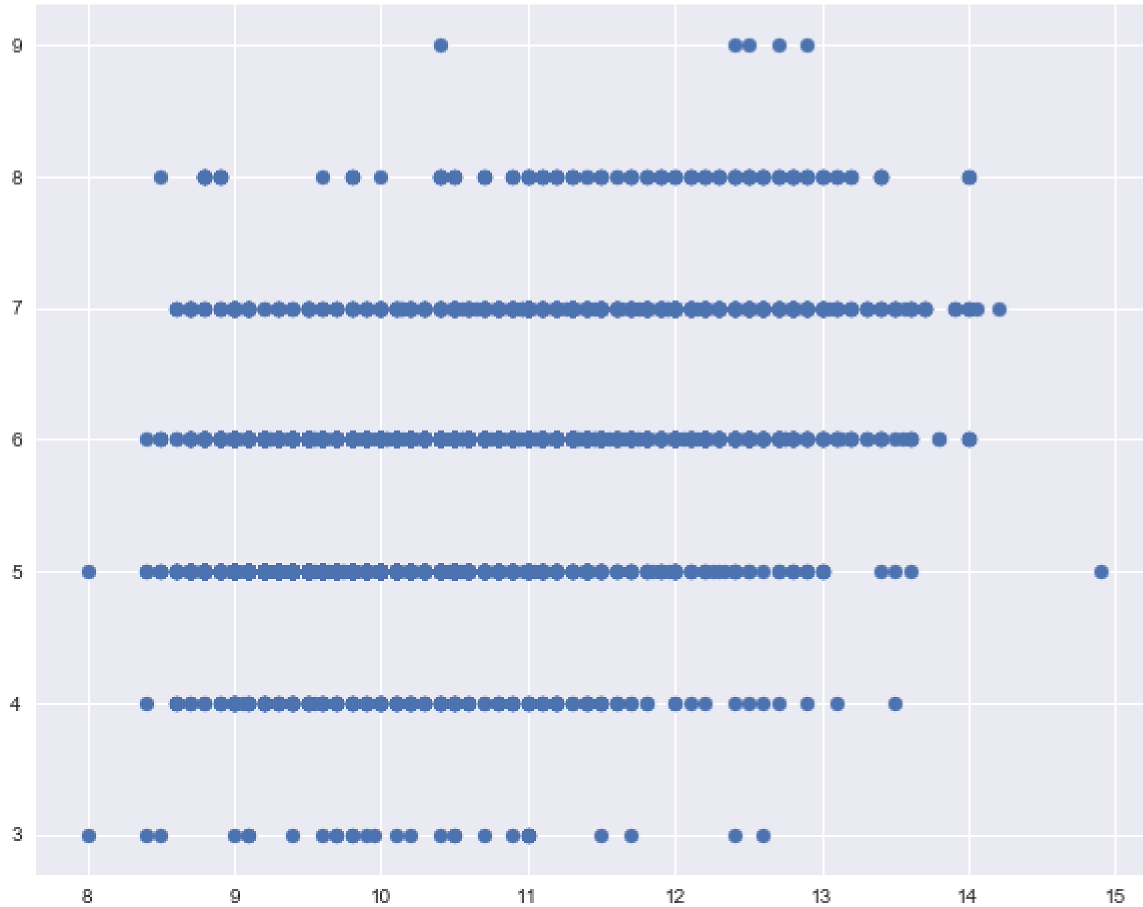
Out[17]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000
mean	7.216579	0.339691	0.318722	5.444326	0.056042	30.525319	115.744574
std	1.295751	0.164548	0.145231	4.757392	0.035031	17.749400	56.521855
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000

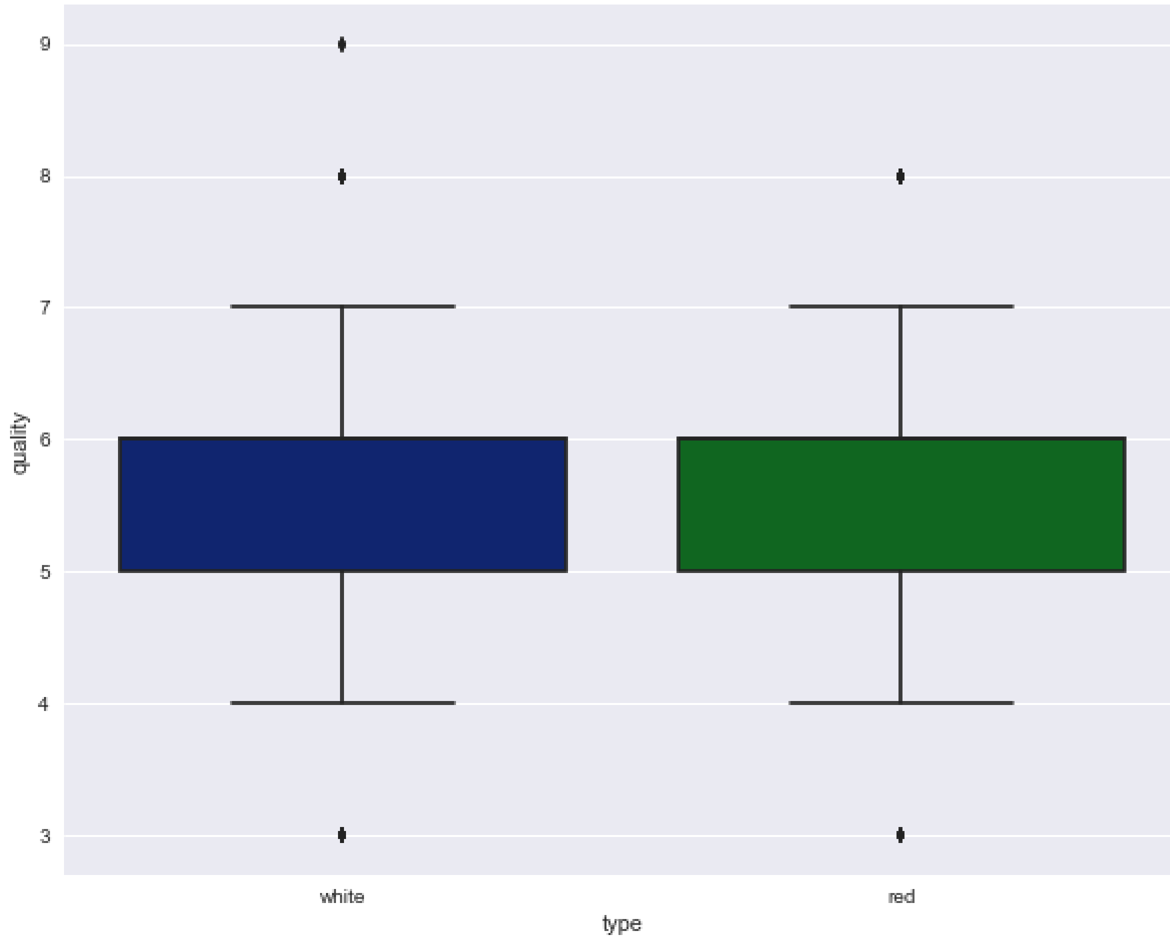
In [18]:
```
import seaborn as sns
sns.set(rc={'figure.figsize':(10,8)})
corr = df.corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
plt.show()
```



In [73]:
```
plt.scatter("alcohol","quality",data=df)
plt.show()
```



In [19]:
```
sns.boxplot(x="type",y="quality",data=df, palette="dark")
plt.show()
```



In [20]:
```
df=df[df.columns.drop('type')]
```

In [21]:
```
df.head(5)
```

Out[21]:

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

In [22]:
```
print(df.nunique())
```

```
fixed acidity           107
volatile acidity        188
citric acid              90
residual sugar          317
chlorides               215
free sulfur dioxide     135
total sulfur dioxide    276
density                 998
pH                      109
sulphates               112
alcohol                 111
quality                   7
dtype: int64
```

In [23]:
```
from sklearn.model_selection import train_test_split
training, testing =train_test_split(df, test_size= 0.30, random_state=24)
```

In [24]:
```
training.shape
```

Out[24]: (4547, 12)

In [25]:
```
testing.shape
```

Out[25]: (1950, 12)

In [28]:
```
X = training['alcohol']
```

In [29]:
```
X.shape
```

Out[29]: (4547,)

In [30]:
```
x= np.array(X)
```

In [31]:
```
x = x.reshape(4547,1)
```

In [32]:
```
x.shape
```
Out[32]: (4547, 1)

In [33]:
```
Y = training['quality']
```

In [34]:
```
Y.shape
```
Out[34]: (4547,)

In [35]:
```
Y= np.array(Y)
```

In [36]:
```
y = Y.reshape(4547,1)
```

In [37]:
```
y.shape
```
Out[37]: (4547, 1)

In [38]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
model=lr.fit(x, y)
```

In [39]:
```
print(model)
```
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [50]:
```
print(model.coef_[0][0]) ## Printing the coefficients
print(model.intercept_[0]) ### printing the Intercept term

print("The linear model is: Y = {:.5} + {:.5}X".format(model.intercept_[0], model
```
```
0.32546629798314014
2.4029999180573034
The linear model is: Y = 2.403 + 0.32547X
```

In [52]:
```
 X_test=testing['alcohol']
```

In [53]:
```
X_test.shape
```
Out[53]: (1950,)

In [82]:
```
X_test = X_test.reshape(1950,1)
```

In [83]:
```
X_test.shape
```
Out[83]: (1950, 1)

In [84]:
```
Y_test=testing['quality']
```

In [85]:
```
Y_test.shape
```
Out[85]: (1950,)

In [86]:
```
Y_test = Y_test.reshape(1950,1)
```

```
C:\Users\PRANAV\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWarn
ing: reshape is deprecated and will raise in a subsequent release. Please use .
values.reshape(...) instead
```

In [88]:
```
Y_test.shape
```

Out[88]:
```
(1950, 1)
```

In [94]:
```
Y_test
```

Out[94]:
```
array([[5],
       [4],
       [5],
       ...,
       [5],
       [7],
       [5]], dtype=int64)
```

In [89]:
```
Y_pred = lr.predict(X_test)
```

In [90]:
```
Y_pred
```

Out[90]:
```
array([[5.72275616],
       [5.52747638],
       [5.3321966 ],
       ...,
       [5.56002301],
       [5.75530279],
       [5.46238312]])
```

In [91]:
```
 from sklearn.metrics import mean_squared_error
```

In [92]:
```
LR_score= mean_squared_error(Y_test,Y_pred)
```

In [93]:
```
LR_score
```

Out[93]:
```
0.610874859296884
```

Interpretation:

The wine quality has been predicted using Linear Regression, with LR score of 61%

In [ ]:
```

```