

# Problem Statement:

A real estate agent want to help to predict the house price for regions in USA. He gave us the dataset to work on to use Linear Regression model. Create a Model that helps him to estimate of what the house would sell for.

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [3]: df=pd.read_csv("winequality.csv")
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	
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	
...	...	...	...	...	...	...	...	...	...	...	...	
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	

1599 rows × 12 columns

```
In [4]: 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

```

In [5]:

`df.head()`

Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	

## Data cleaning and Pre-Processing

In [6]:

`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

```

In [7]:

`df.describe()`

Out[7]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	quality
<b>mean</b>	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.
<b>std</b>	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.
<b>min</b>	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.
<b>25%</b>	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.
<b>50%</b>	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.
<b>75%</b>	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.
<b>max</b>	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.

In [8]:

```
df.dropna(axis='columns')
```

Out[8]:

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

1599 rows × 12 columns

In [9]:

```
a = df.dropna(axis='columns')
a.columns
```

Out[9]:

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

In [10]:

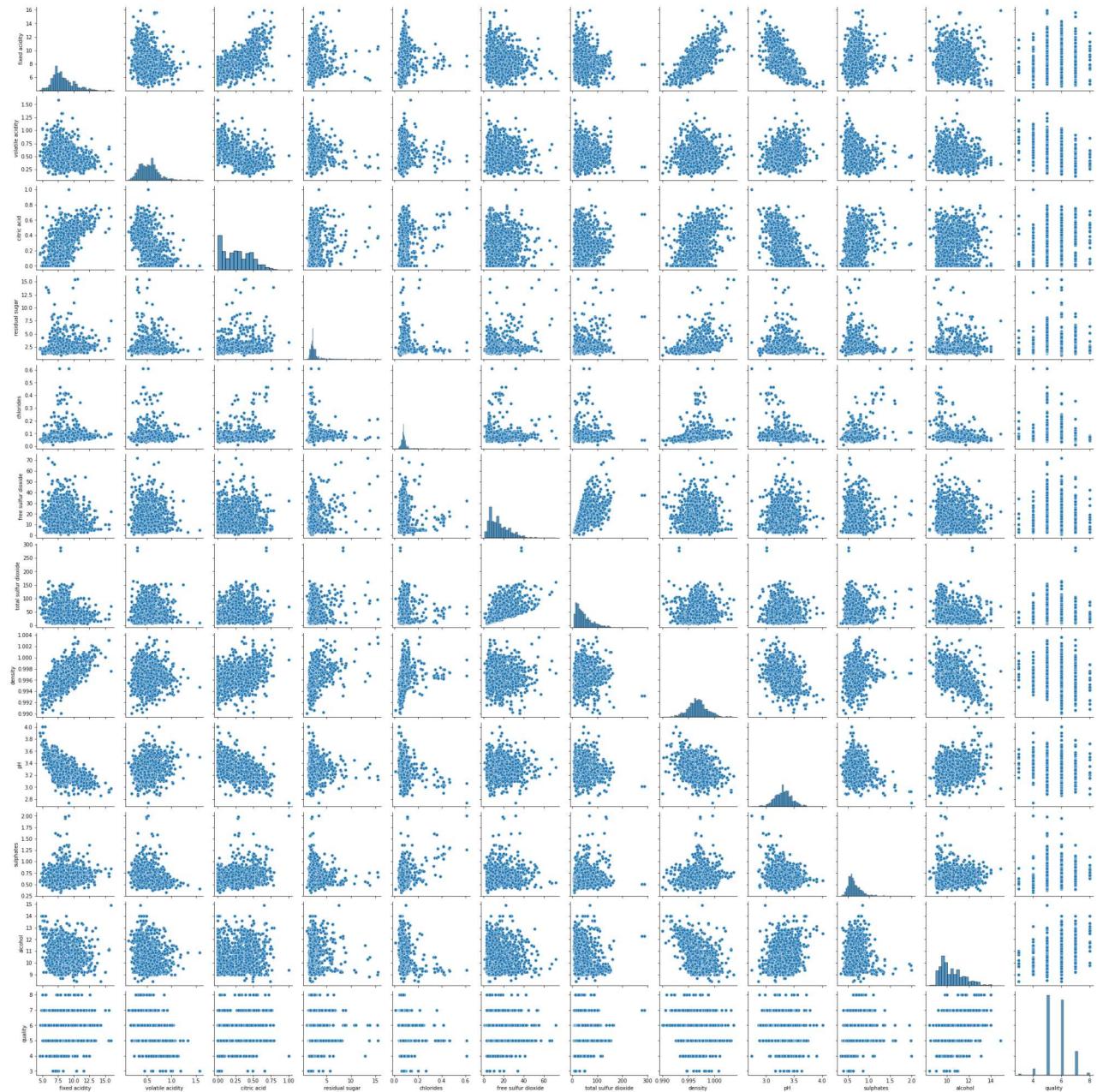
```
df.columns
```

```
Out[10]: Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
   'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
   'pH', 'sulphates', 'alcohol', 'quality'],
  dtype='object')
```

## EDA and VISUALIZATION

```
In [11]: sns.pairplot(df)
```

```
Out[11]: <seaborn.axisgrid.PairGrid at 0xec5b1c6eb0>
```

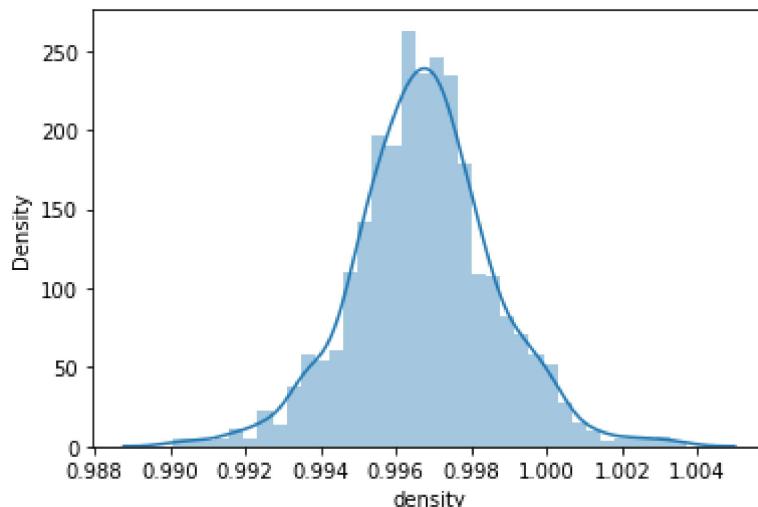


```
In [12]: sns.distplot(df['density'])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) o

```
r `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)
```

Out[12]: <AxesSubplot:xlabel='density', ylabel='Density'>



In [13]:

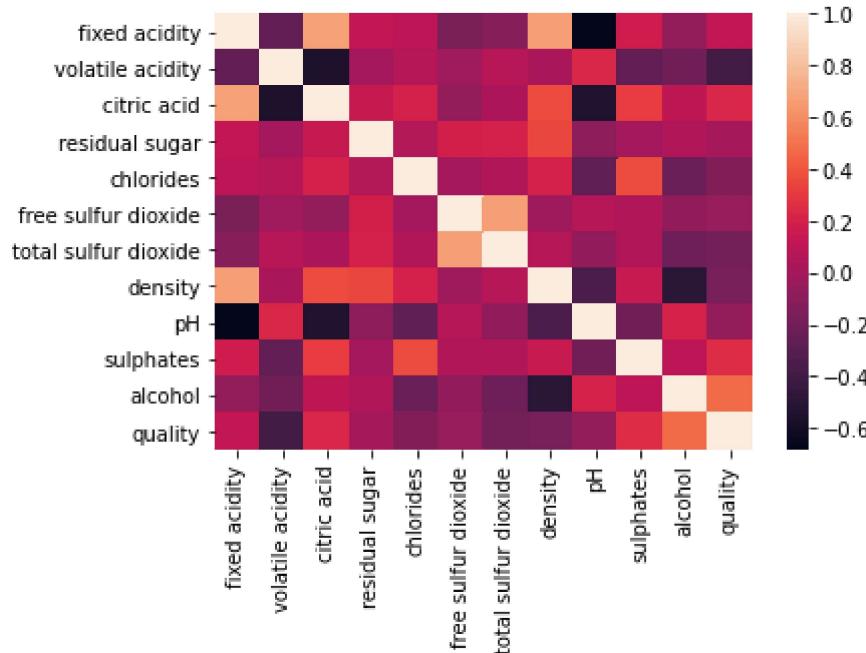
```
df1=df[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
        'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
        'pH', 'sulphates', 'alcohol', 'quality']]
```

## Plot Using Heat Map

In [14]:

```
sns.heatmap(df1.corr())
```

Out[14]: <AxesSubplot:>



## To Train The Model-Model Building

we are going to train Linear Regression Model; We need to split our data into two variables x and y where x is independent variable(input) and y is dependent on x(output) we could ignore address column as it required for our model

```
In [15]: x=df1[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
           'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
           'pH', 'sulphates', 'alcohol', ]]
y=df1['quality']
```

## To Split my dataset into training and test data

```
In [16]: from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
In [17]: from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[17]: LinearRegression()

```
In [18]: lr.intercept_
```

Out[18]: 27.43377035145128

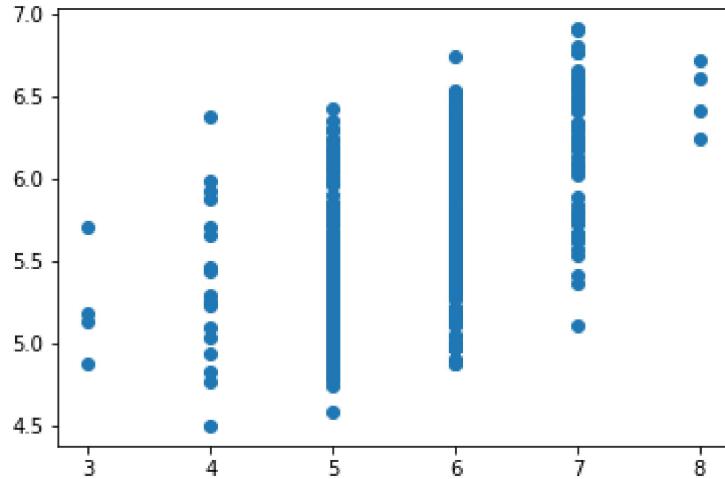
```
In [19]: coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

	Co-efficient
<b>fixed acidity</b>	0.044208
<b>volatile acidity</b>	-1.011553
<b>citric acid</b>	-0.137128
<b>residual sugar</b>	0.022786
<b>chlorides</b>	-2.100006
<b>free sulfur dioxide</b>	0.003833
<b>total sulfur dioxide</b>	-0.004075
<b>density</b>	-23.852831
<b>pH</b>	-0.275970
<b>sulphates</b>	0.917957
<b>alcohol</b>	0.263386

In [20]:

```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[20]: &lt;matplotlib.collections.PathCollection at 0x1ec63a7e340&gt;



In [21]:

```
lr.score(x_test,y_test)
```

Out[21]: 0.3223262071115506