

DATA COLLECTION

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

```
In [2]: # To Import Dataset
sd=pd.read_csv(r"c:\Users\user\Downloads\11_winequality-red.csv")
sd
```

Out[2]:

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

1599 rows × 12 columns

```
In [3]: # to display top 10 rows
sd.head(10)
```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
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
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5
9	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5



DATA CLEANING AND PRE_PROCESSING


```
In [4]: sd.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]: # to display summary of statistics
sd.describe()
```

```
Out[5]:
```

	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
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.46779
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.89532
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.00000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.00000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.00000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.00000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.00000



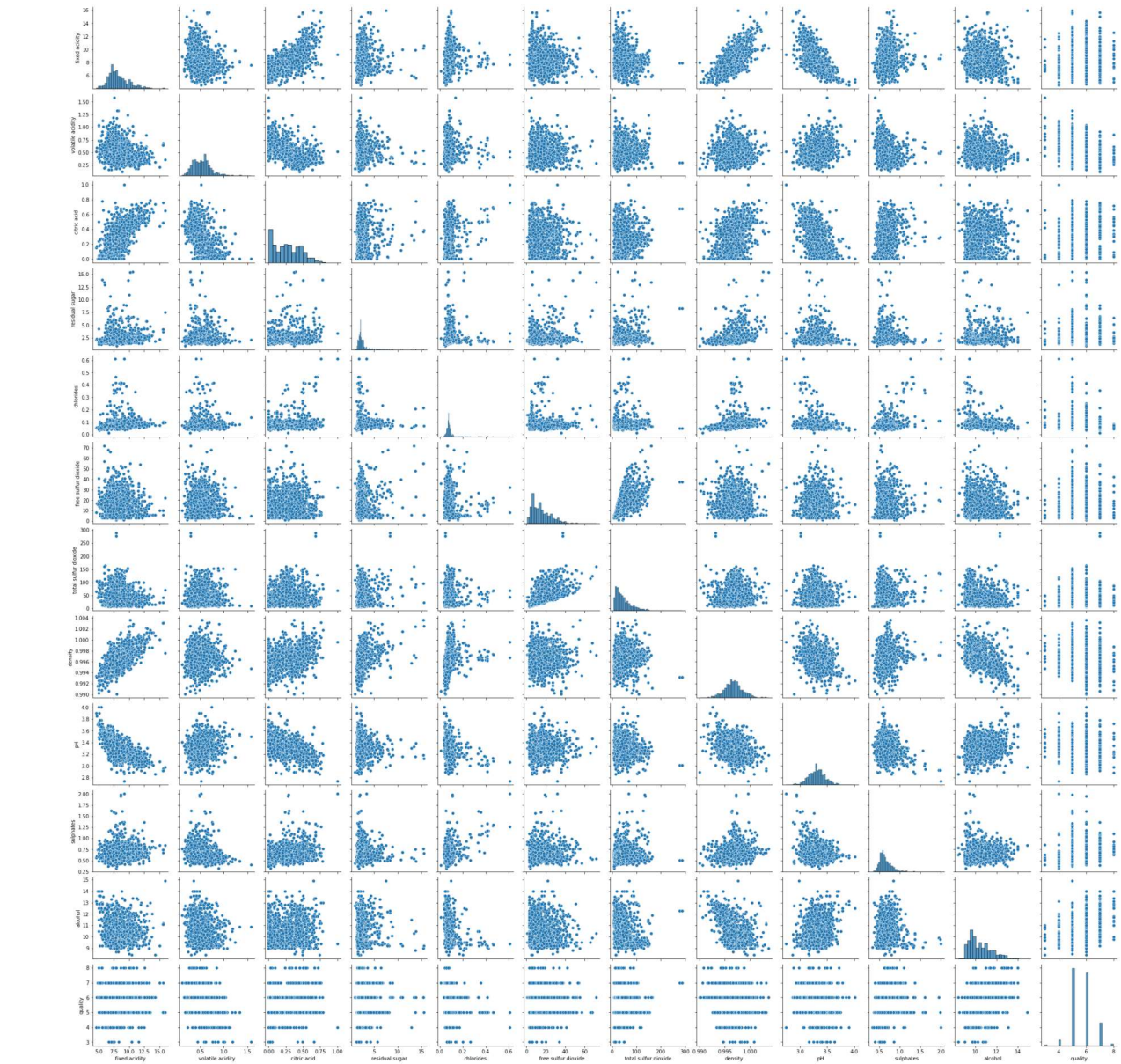
```
In [6]: #to display columns heading
sd.columns
```

```
Out[6]: 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 [7]: sns.pairplot(sd)
```

```
Out[7]: <seaborn.axisgrid.PairGrid at 0x17589b8c6a0>
```

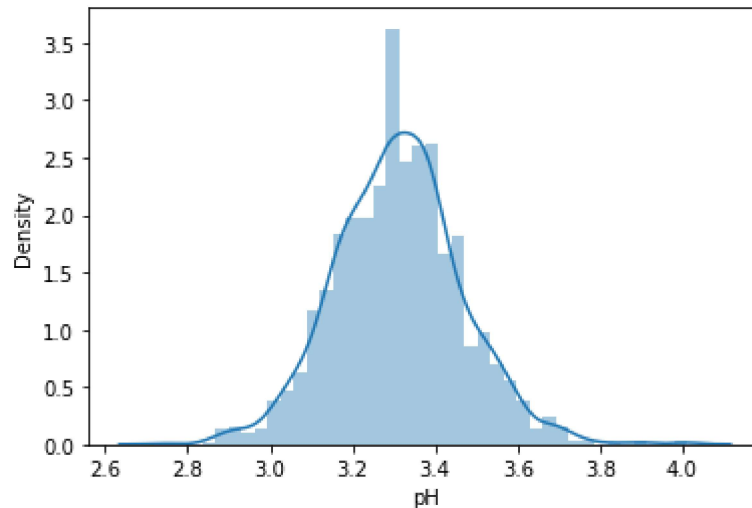


```
In [8]: sns.distplot(sd['pH'])
```

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) or `histplot` (an axes-level function for histograms).

```
warnings.warn(msg, FutureWarning)
```

```
Out[8]: <AxesSubplot:xlabel='pH', ylabel='Density'>
```



```
In [21]: sd1=sd[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',  
                'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',  
                'pH']]
```

```
In [18]: sns.heatmap(sd1.corr())
```

```
Out[18]: <AxesSubplot:>
```



TO TRAIN THE MODEL _MODEL BUILDING

we are going to train a Linear Regression model; we need to split out the data into two variables x and y where x is independent on x (output) and y is dependent on x (output) address column as it is not required in our model

```
In [19]: x= sd1[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',  
                'Avg. Area Number of Bedrooms', 'Area Population']]  
y=sd1['Price']
```

```
In [20]: # To split my dataset into training data and test data  
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 [33]: from sklearn.linear_model import LinearRegression  
  
lr=LinearRegression()  
lr.fit(x_train,y_train)
```

```
Out[33]: LinearRegression()
```

```
In [24]: from sklearn.linear_model import LinearRegression
```

```
lr=LinearRegression()  
lr.fit(x_train,y_train)
```

```
Out[24]: LinearRegression()
```

```
In [25]: print(lr.intercept_)
```

```
-2636126.984887991
```

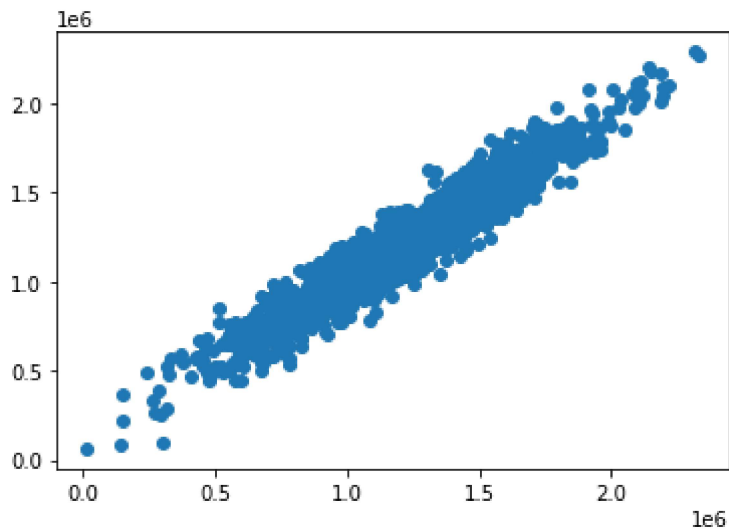
```
In [27]: coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])  
coeff
```

```
Out[27]:
```

	Co-efficient
Avg. Area Income	21.587293
Avg. Area House Age	165076.695331
Avg. Area Number of Rooms	120121.558144
Avg. Area Number of Bedrooms	3572.532356
Area Population	15.119886

```
In [30]: prediction = lr.predict(x_test)  
plt.scatter(y_test,prediction)
```

```
Out[30]: <matplotlib.collections.PathCollection at 0x2962358cf10>
```



```
In [31]: print(lr.score(x_test,y_test))
```

```
0.9147393476313253
```

```
In [ ]:
```

