LINEAR REGRESSION

Simple Linear Regression

Simple linear regression is a statistical machine learning method that allows us to summarize and study relationship between two continous (quantative) variables.
Linear regression attempts to model the relationshi between two variables by fitting a linear equation to obsered data.
One variable is considered to br an explanatory variable and the other is considered to br a dependent variable.

Simple Linear regression model

Simple Linear Equation

$E(y) = \beta_0+\beta_1 x$ ...(Pure Mathematical Notations)

Estimate of the Simple Linear Equation

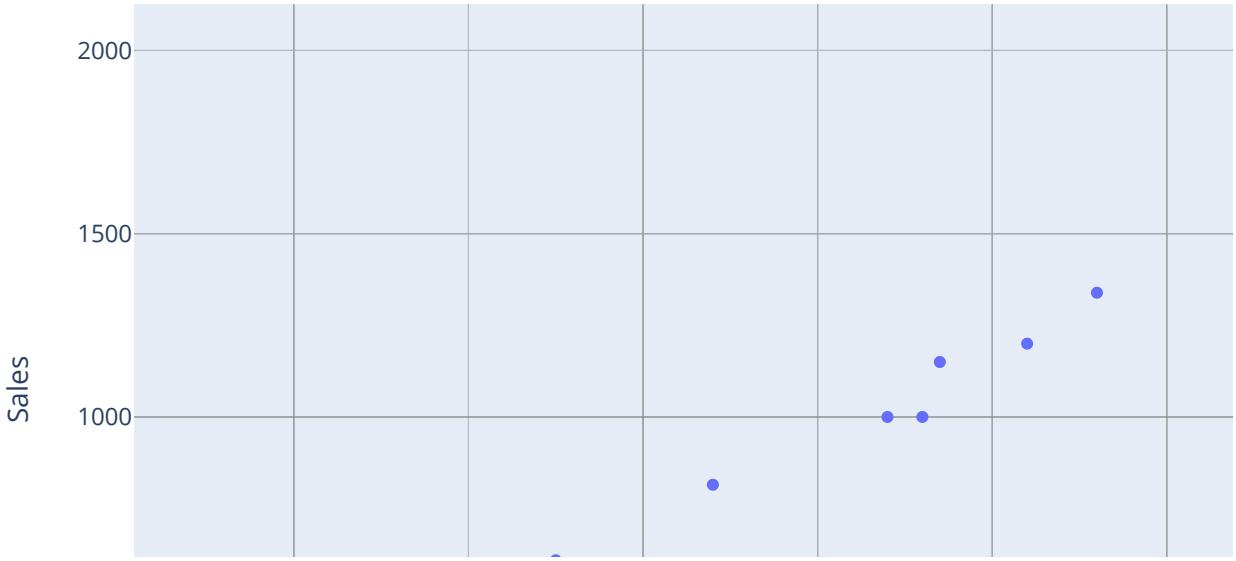
$\hat{y} = b_0+b_1 x$

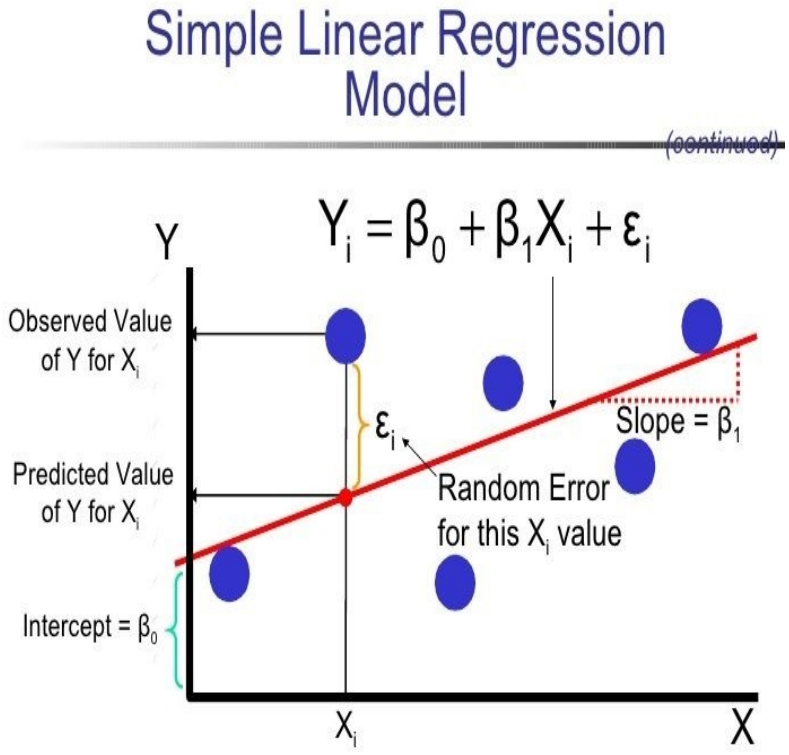
In [1]:
```
import pandas as pd
import plotly.graph_objects as go

burger = pd.read_csv('E:\\Machine Learning\\Datasets\\Burger.csv')
burger.head()
```

Out[1]:

	Promote	Sales
0	23	554
1	56	1339
2	34	815
3	25	609
4	67	1600

In [2]:
```
import plotly.express as px
fig = px.scatter(burger, x = 'Promote', y= 'Sales')
fig.show()
```





STEP 1 : PREPROCESS THE DATA

- Import the libraries
- Import the dataset
- Read the Data
- Check for null values in columns
- Split the DataSet

In [3]:
```
## Import the libraries
import pandas as pd
import numpy as np

from sklearn import linear_model
from sklearn import datasets
from sklearn.model_selection import train_test_split
```

In [4]:
```
## Import the dataset
data = datasets.load_boston()
df = pd.DataFrame(data.data, columns = data.feature_names)
df.head()
```

Out[4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

In [5]:
```
target = pd.DataFrame(data.target,columns=['MEDV'])
target
```

Out[5]:

	MEDV
0	24.0
1	21.6
2	34.7
3	33.4
4	36.2
...	...
501	22.4
502	20.6
503	23.9
504	22.0
505	11.9

506 rows × 1 columns

In [6]:
```
# Check for null values in columns
print('NUM in RM : ',df['RM'].isnull().values.any())#chek if there are NULL in the
print('NUM in MEDV : ',target['MEDV'].isnull().values.any())
```

```
NUM in RM :  False
NUM in MEDV :  False
```

In [7]:
```
X= pd.DataFrame(df['RM'])
y = pd.DataFrame(target['MEDV'])
```

In [8]:
```
#Split the data...
#split the data in 80/20 proportion
X_train ,X_test, y_train, y_test = train_test_split(X,y,test_size =0.2, random_stat
```

In [9]:
```
print('o X : ',X.shape)
print("X_train :",X_train.shape)
print("X_test : ",X_test.shape)
print('\no y : ',y.shape)
print("y_train :",y_train.shape)
print("y_test: ",y_test.shape)
```

```
o X :  (506, 1)
X_train : (404, 1)
X_test :  (102, 1)

o y :  (506, 1)
y_train : (404, 1)
y_test:  (102, 1)
```

STEP 2 : FITTING SIMPLE LINEAR REGRESSION MODEL TO THE TRAINING SET

To fit the dataset into the model we will use LinearRegression class from sklearn.linear_model library .Then we make object of lm from LinearRegression() . We will fit the regressor object into to our dataset using fit() method of LinearRegression.

In [10]:
```
lm = linear_model.LinearRegression()
model = lm.fit(X_train,y_train)
print('coeficent',lm.coef_)
print('Intercept',lm.intercept_)
print('R square',lm.score(X_train,y_train))
```

```
coeficent [[9.37638431]]
Intercept [-36.47618963]
R square 0.4970800097843844
```

STEP 3 : PREDICTING THE RESULT

Now we will predict the observations from our type set. We will save the output in vector `y_pred`. To predict the result we predict method of LinearRegression class on the regressor we trained in the previous step.

In [11]:
```
y_pred = lm.predict(X_test)
print('y_pred : ',y_pred[0:5]) ## Printing only first five y_pred elements
print(len(y_pred))
```

```
y_pred :  [[22.90445223]
 [21.80741526]
 [23.2795076 ]
 [13.67809006]
 [21.95743741]]
102
```

STEP 4 : VISUALIZATION

The final Step is to visualize the results. We will use plotly library, to make scatter plot of our training set results and test set results to see how closer our model predicted the values..

In [12]:
```
X_test
```

Out[12]:

	RM
329	6.333
371	6.216
219	6.373
403	5.349
78	6.232
...	...
56	6.383
455	6.525
60	5.741
213	6.375
108	6.474

102 rows × 1 columns

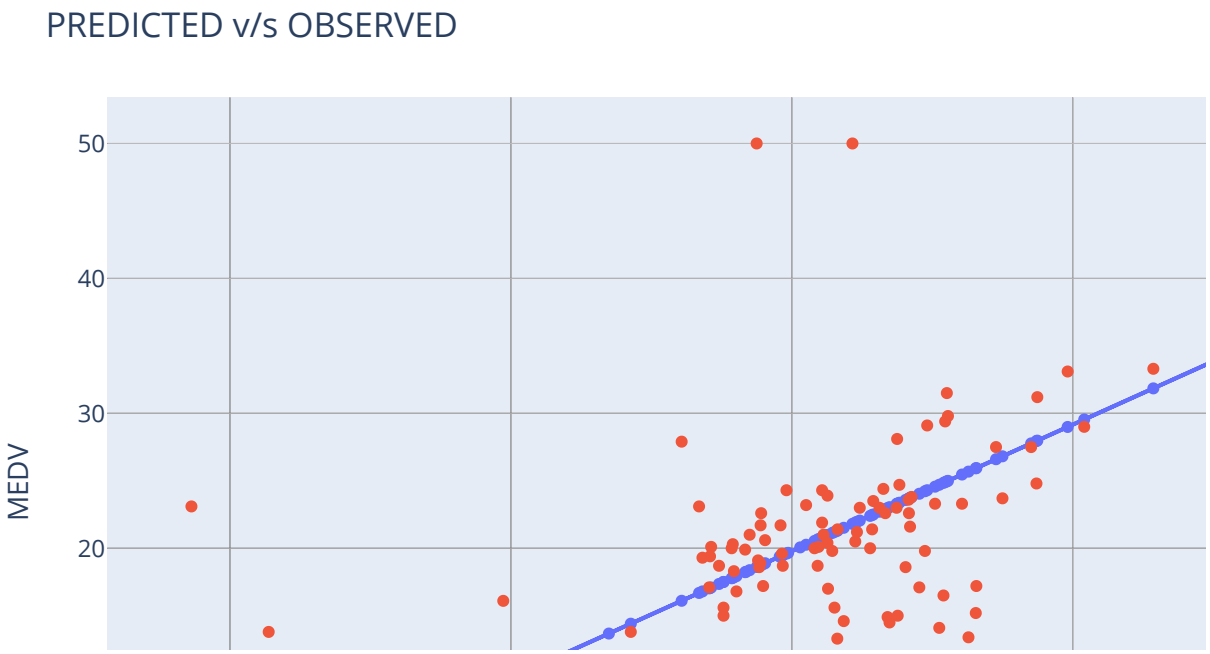
In [13]:
```
y_predict = pd.DataFrame(data = y_pred,columns = ['y_pred']) #store the data of y_p
y_predict
```

Out[13]:

	y_pred
0	22.904452
1	21.807415
2	23.279508
3	13.678090
4	21.957437
...	...
97	23.373271
98	24.704718
99	17.353633
100	23.298260
101	24.226522

102 rows × 1 columns

In [17]:
```
import plotly.express as px
fig = px.scatter()
fig.add_scatter(x = X_test['RM'], y=y_predict['y_pred'],mode = 'lines+markers',name
fig.add_scatter(x = X_test['RM'], y=y_test['MEDV'],mode = 'markers',name ='OBSERVED
fig.update_layout(title='PREDICTED v/s OBSERVED',xaxis_title='RM', yaxis_title='MED
fig.show()
```



In [ ]: