In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
import matplotlib.pyplot as plt
from sklearn.datasets import load_boston
```

In [3]:
```
boston=load_boston()
```

In [9]:
```
bos=pd.DataFrame(boston.data)
```

In [13]:
```
bos.columns=boston.feature_names
```

In [15]:
```
bos['Price']=boston.target
```

In [43]:
```
bos.head()
```

Out[43]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
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	24.0
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	21.6
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	34.7
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	33.4
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	36.2

In [21]:
```
Feature_Columns=bos.columns.drop('Price')
```

In [25]:
```
X=bos[Feature_Columns.tolist()]
```

In [26]:
```
Y=bos.Price
```

In [27]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [69]:
```
X_Train,X_Test,Y_Train,Y_Test=train_test_split(X,Y,test_size=.30,random_state=43)
```

In [70]:
```
lm=LinearRegression()
```

In [71]:
```
lm.fit(X_Train,Y_Train)
```

Out[71]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [72]:
```
lm.intercept_
```

Out[72]:
```
35.850361646380556
```

In [73]:
```
lm.coef_
```

Out[73]:
```
array([-3.23260120e-02,  4.35771384e-02,  3.72914568e-02,  2.59136498e+00,
       -1.76693690e+01,  3.77994491e+00,  2.21348220e-02, -1.24169174e+00,
        3.46294130e-01, -1.42116459e-02, -1.00581111e+00,  1.13185782e-02,
       -6.48146062e-01])
```

In [74]:
```
Sum_Of_Squared_Errors=np.sum((lm.predict(X_Train)-Y_Train)**2)
Sum_Of_Squared_Errors
```

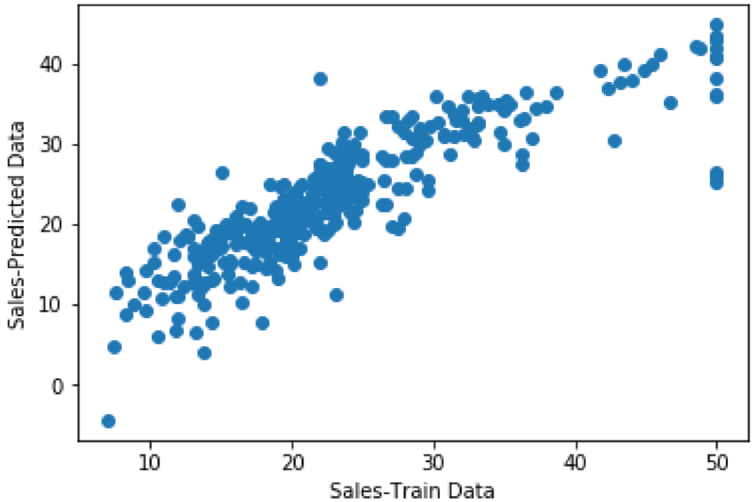
Out[74]:
```
7584.984478127653
```

In [75]:
```
Mean_Of_Squared_Errors=Sum_Of_Squared_Errors/506
Mean_Of_Squared_Errors
```

Out[75]:
```
14.99008790143805
```

In [89]:
```
plt.xlabel('Sales-Train Data')
plt.ylabel('Sales-Predicted Data')
plt.scatter(Y_Train,lm.predict(X_Train))
```

Out[89]:
```
<matplotlib.collections.PathCollection at 0x7fd096424c50>
```



We see this is not a good model for linear regression as Sum Of Squared Errors is Too high.

In [111]:
```
import statsmodels.formula.api as smf
```

In [121]:
```
lm=smf.ols(formula='Price~CRIM+ZN+INDUS+CHAS+NOX+RM+AGE+DIS+RAD+TAX+PTRATIO+B+LSTAT',data=bos).fit()
```

In [122]:
```
lm.summary()
```

Out[122]:

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.741
Model:	OLS	**Adj. R-squared:**	0.734
Method:	Least Squares	**F-statistic:**	108.1
Date:	Sun, 21 Oct 2018	**Prob (F-statistic):**	6.95e-135
Time:	20:44:26	**Log-Likelihood:**	-1498.8
No. Observations:	506	**AIC:**	3026.
Df Residuals:	492	**BIC:**	3085.
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	36.4911	5.104	7.149	0.000	26.462	46.520
CRIM	-0.1072	0.033	-3.276	0.001	-0.171	-0.043
ZN	0.0464	0.014	3.380	0.001	0.019	0.073
INDUS	0.0209	0.061	0.339	0.735	-0.100	0.142
CHAS	2.6886	0.862	3.120	0.002	0.996	4.381
NOX	-17.7958	3.821	-4.658	0.000	-25.302	-10.289
RM	3.8048	0.418	9.102	0.000	2.983	4.626
AGE	0.0008	0.013	0.057	0.955	-0.025	0.027
DIS	-1.4758	0.199	-7.398	0.000	-1.868	-1.084
RAD	0.3057	0.066	4.608	0.000	0.175	0.436
TAX	-0.0123	0.004	-3.278	0.001	-0.020	-0.005
PTRATIO	-0.9535	0.131	-7.287	0.000	-1.211	-0.696
B	0.0094	0.003	3.500	0.001	0.004	0.015
LSTAT	-0.5255	0.051	-10.366	0.000	-0.625	-0.426

Omnibus:	178.029	**Durbin-Watson:**	1.078
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	782.015
Skew:	1.521	**Prob(JB):**	1.54e-170
Kurtosis:	8.276	**Cond. No.**	1.51e+04

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.51e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

```
This is not a good model for linear regression as Adj. R-squared is low with high AIC.It also has high values
of P>[t] for AGE and INDUS features.
```