In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import linear_model
from sklearn import datasets
from sklearn.linear_model import Ridge,Lasso
from sklearn.metrics import mean_squared_error, r2_score
dataset = datasets.load_boston()
```

In [2]:
```
dataset_x = dataset.data[:, np.newaxis, 5]
target = pd.DataFrame(dataset.target, columns=["MEDV"])
dataset_y = target["MEDV"]
```

In [6]:
```
from sklearn.cross_validation import train_test_split
x_train, x_test, y_train, y_test = train_test_split(dataset_x, dataset
_y, test_size = 1/5, random_state = 0)
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)
pred = regressor.predict(x_test)
```

```
/anaconda3/lib/python3.6/site-packages/sklearn/cross_validation.py:4
1: DeprecationWarning: This module was deprecated in version 0.18 in
favor of the model_selection module into which all the refactored cl
asses and functions are moved. Also note that the interface of the n
ew CV iterators are different from that of this module. This module
will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

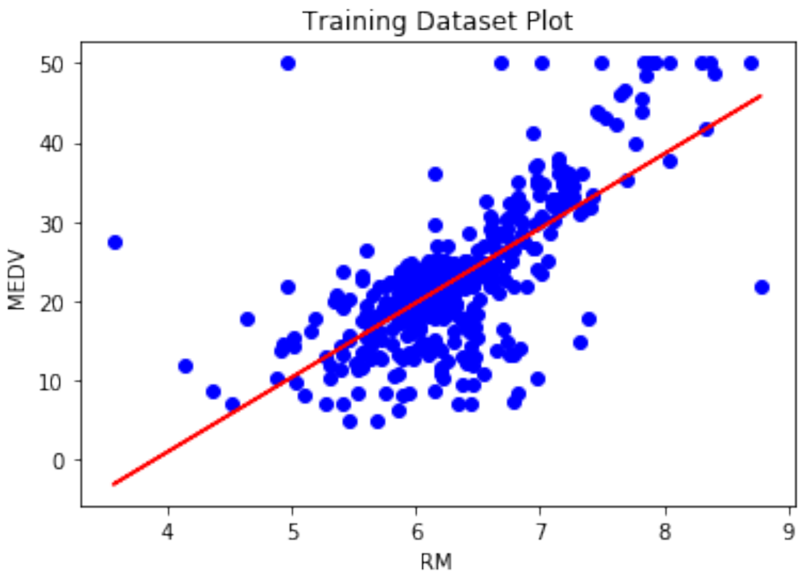
In [7]:
```
# Printing the Regressor Coefficient
print(regressor.coef_)
```

```
[ 9.37638431]
```

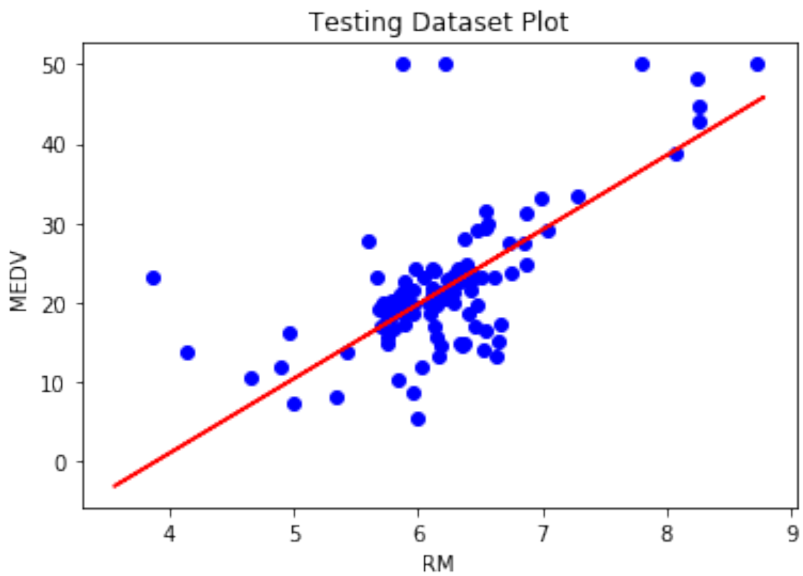
In [8]:
```
# Printing the Mean Square Error
print(mean_squared_error(y_test, pred))
```

```
46.9073516274
```

In [9]:
```
# Plotting the Training Dataset
plt.scatter(x_train, y_train, color = 'blue')
plt.plot(x_train, regressor.predict(x_train), color = 'red')
plt.title('Training Dataset Plot')
plt.xlabel('RM')
plt.ylabel('MEDV')
plt.show()
```



In [10]:
```
# Plot Testing Dataset
plt.scatter(x_test, y_test, color = 'blue')
plt.plot(x_train, regressor.predict(x_train), color = 'red')
plt.title('Testing Dataset Plot')
plt.xlabel('RM')
plt.ylabel('MEDV')
plt.show()
```



In [11]:
```
# Finding Model accuracy for Linear Regression
print(regressor.score(x_test,y_test))
```

```
0.423943868165
```

In [22]:
```
# Applying Ridge Regression
reg=Ridge(alpha= 0.25, normalize=True).fit(x_train,y_train)

print(reg.coef_)
```

```
[ 7.50110745]
```

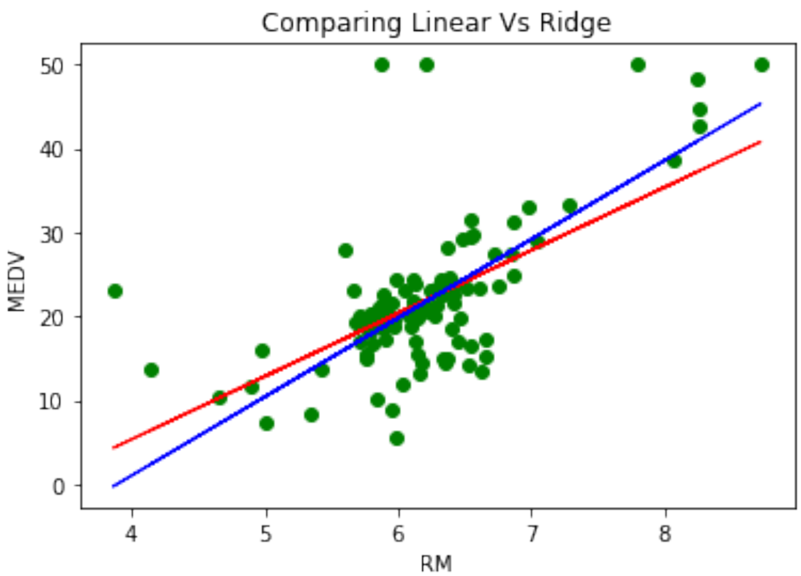
In [13]:

```
[ 7.50110745]
```

In [14]:
```
# Ridge Regression Intercept
print(reg.intercept_)
```

```
-24.6585754645
```

In [15]:
```
# Ploting Comparing Linear Regression Vs Ridge Regression
pred_reg = reg.predict(x_test)
plt.scatter(x_test, y_test, color='Green')
plt.plot(x_test, pred_reg, color='red', linewidth=1)
plt.plot(x_test, pred, color='blue', linewidth=1)
plt.title('Comparing Linear Vs Ridge')
plt.xlabel('RM')
plt.ylabel('MEDV')
plt.show()
```



In [16]:
```
# Model accuracy for Ridge Regression
print(reg.score(x_test,y_test))
```

```
0.431868879102
```

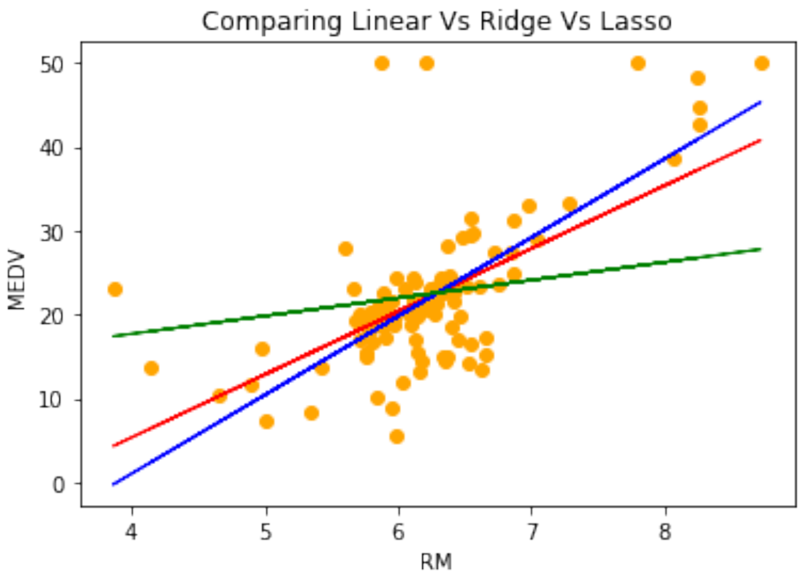
In [17]:
```
# Applying Lasso Regression
las = Lasso (alpha=0.25, normalize=True)
las.fit (x_train, y_train)
print(las.coef_)
```

```
[ 2.13408231]
```

In [18]:
```
# Lasso intercept
print(las.intercept_)
pred_las = las.predict(x_test)
```

```
9.1633276055
```

In [19]:
```
# Plot Comparing Linear Vs Ridge Vs Lasso
plt.scatter(x_test, y_test, color='orange')
plt.plot(x_test, pred_reg, color='red', linewidth=1)
plt.plot(x_test, pred, color='blue', linewidth=1)
plt.plot(x_test, pred_las, color='green', linewidth=1)
plt.title('Comparing Linear Vs Ridge Vs Lasso')
plt.xlabel('RM')
plt.ylabel('MEDV')
plt.show()
```



In [20]:
```
# Model accuracy for Lasso Regression
print(las.score(x_test,y_test))
```

```
0.197314873789
```

In [21]:
```
# Comparison between Linear Regression Vs Ridge Regression Vs Lasso Re
gression
print(regressor.score(x_test,y_test)*100)
print(reg.score(x_test,y_test)*100)
print(las.score(x_test,y_test)*100)
```

```
42.3943868165
43.1868879102
19.7314873789
```

In [ ]:
```
# The above analysis though R-squared values shows that Ridge Regressi
on performs the best followed by Linear
#and Lasso Regression since the R-squared value of Ridge regression is
the maximum.
```