Polynomial Regression

Using scikit-learn to implement Polynomial Linear Regression. Check how Engine Size is related to Co2Emissions. Create a model using train set,test using test set .

In [1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import pylab as pl
import numpy as np
%matplotlib inline
```

In [2]:
```
df = pd.read_csv("FuelConsumptionCo2.csv")
#df.head()
#df.describe()
```

Creating train and test dataset:

Train/Test Split dataset to mutually exclusive. We can use 80% of the entire data for training, and the 20% for testing. We create a mask using np.random.rand().

In [3]:
```
cdf = df[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_COMB','CO2EMISSIONS']]
msk = np.random.rand(len(df)) < 0.8
train = cdf[msk]
test = cdf[~msk]
```

Polynomial Regression

y = b + theta_1 x + theta_2 x^2 where x = Engine Size

Creating model using Training Set

In [4]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model

#Getting Training and Test Sets
train_x = np.asanyarray(train[['ENGINESIZE']])
train_y = np.asanyarray(train[['CO2EMISSIONS']])
test_x = np.asanyarray(test[['ENGINESIZE']])
test_y = np.asanyarray(test[['CO2EMISSIONS']])

#Trying to fit Polynomial of degree 2
poly = PolynomialFeatures(degree=2)
train_x_poly = poly.fit_transform(train_x)
train_x_poly
```

Out[4]:
```
array([[ 1.  ,  2.  ,  4.  ],
       [ 1.  ,  2.4 ,  5.76],
       [ 1.  ,  1.5 ,  2.25],
       ...,
       [ 1.  ,  3.  ,  9.  ],
       [ 1.  ,  3.2 , 10.24],
       [ 1.  ,  3.2 , 10.24]])
```

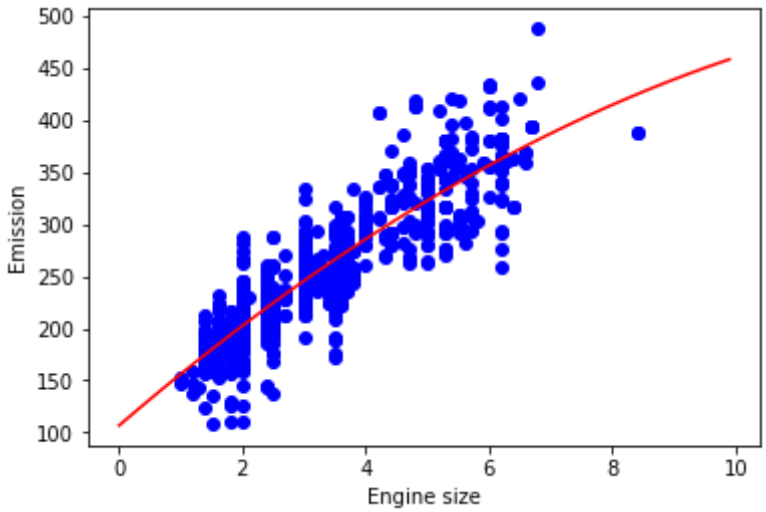
In [5]:
```
#Polynomial Function is a special case of linear regression
clf = linear_model.LinearRegression()
train_y_ = clf.fit(train_x_poly, train_y)
# The coefficients
print ('Coefficients: ', clf.coef_)
print ('Intercept: ',clf.intercept_)
```

```
Coefficients:  [[ 0.         51.04847447 -1.56633261]]
Intercept:  [106.6213904]
```

Plotting to find how Polynomial regression line fits:

In [6]:
```
plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS,  color='blue')
XX = np.arange(0.0, 10.0, 0.1)
yy = clf.intercept_[0]+ clf.coef_[0][1]*XX+ clf.coef_[0][2]*np.power(XX, 2)
plt.plot(XX, yy, '-r' )
plt.xlabel("Engine size")
plt.ylabel("Emission")
```

Out[6]:
```
Text(0, 0.5, 'Emission')
```



Evaluate using R2 score

In [7]:
```
from sklearn.metrics import r2_score

test_x_poly = poly.fit_transform(test_x)
test_y_ = clf.predict(test_x_poly)

print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) **
print("R2-score: %.2f" % r2_score(test_y,test_y_ ) )
```

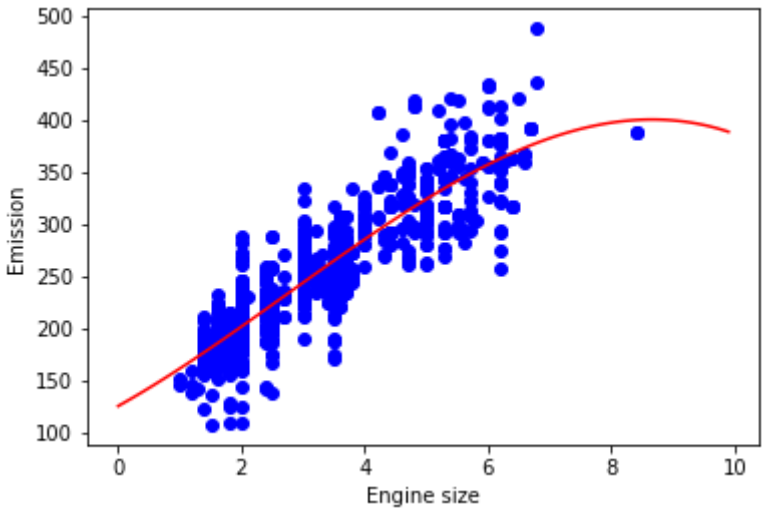
```
Mean absolute error: 23.98
Residual sum of squares (MSE): 985.77
R2-score: 0.74
```

Trying to fit Polynomial of degree 3

In [8]:
```
poly3 = PolynomialFeatures(degree=3)
train_x_poly3 = poly3.fit_transform(train_x)
clf3 = linear_model.LinearRegression()
train_y3_ = clf3.fit(train_x_poly3, train_y)

# The coefficients
print ('Coefficients: ', clf3.coef_)
print ('Intercept: ',clf3.intercept_)
plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS,  color='blue')
XX = np.arange(0.0, 10.0, 0.1)
yy = clf3.intercept_[0]+ clf3.coef_[0][1]*XX + clf3.coef_[0][2]*np.power(XX
plt.plot(XX, yy, '-r' )
plt.xlabel("Engine size")
plt.ylabel("Emission")
test_x_poly3 = poly3.fit_transform(test_x)
test_y3_ = clf3.predict(test_x_poly3)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y3_ - test_y))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y3_ - test_y) *
print("R2-score: %.2f" % r2_score(test_y,test_y3_ ) )
```

```
Coefficients:  [[ 0.         33.02808771  3.36599809 -0.40574581]]
Intercept:  [125.82115194]
Mean absolute error: 23.95
Residual sum of squares (MSE): 979.67
R2-score: 0.74
```



In [ ]:

In [ ]: