PolynomialRegression

June 16, 2020

[26]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import pylab as pl
%matplotlib inline
```

[27]:
```
!wget -O FuelConsumption.csv https://s3-api.us-geo.objectstorage.softlayer.net/
↪cf-courses-data/CognitiveClass/ML0101ENv3/labs/FuelConsumptionCo2.csv
```

```
--2020-05-28 22:08:53--  https://s3-api.us-geo.objectstorage.softlayer.net/cf-
courses-data/CognitiveClass/ML0101ENv3/labs/FuelConsumptionCo2.csv
Resolving s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-
geo.objectstorage.softlayer.net)… 67.228.254.196
Connecting to s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-
geo.objectstorage.softlayer.net)|67.228.254.196|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 72629 (71K) [text/csv]
Saving to: 'FuelConsumption.csv'

FuelConsumption.csv 100%[===================>]  70.93K  --.-KB/s    in 0.04s

2020-05-28 22:08:53 (1.62 MB/s) - 'FuelConsumption.csv' saved [72629/72629]
```

[28]:
```
df = pd.read_csv("FuelConsumption.csv")
df.head()
```

[28]:

	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION	FUELTYPE	FUELCONSUMPTION_CITY	FUELCONSUMPTION_HWY
0	2014	ACURA	ILX	COMPACT	2.0	4	AS5	Z	9.9	6.7
1	2014	ACURA	ILX	COMPACT	2.4	4	M6	Z	11.2	7.7
2	2014	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7	Z	6.0	5.8
3	2014	ACURA	MDX 4WD	SUV - SMALL	3.5	6				
4	2014	ACURA	RDX AWD	SUV - SMALL	3.5	6				

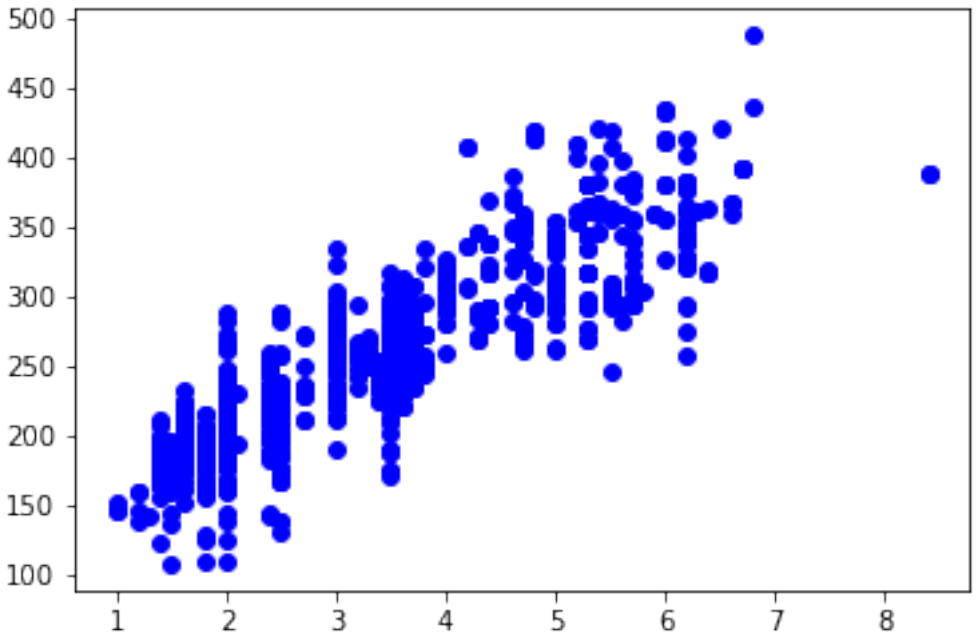
```
3          AS6        Z                    12.7                     9.1
4          AS6        Z                    12.1                     8.7

   FUELCONSUMPTION_COMB  FUELCONSUMPTION_COMB_MPG  CO2EMISSIONS
0                   8.5                        33           196
1                   9.6                        29           221
2                   5.9                        48           136
3                  11.1                        25           255
4                  10.6                        27           244
```

[29]:
```
regrdata = df[['ENGINESIZE', 'CYLINDERS', 'FUELCONSUMPTION_COMB',␣
↪'CO2EMISSIONS']]
regrdata.head(9)
```

[29]:
```
   ENGINESIZE  CYLINDERS  FUELCONSUMPTION_COMB  CO2EMISSIONS
0         2.0          4                   8.5           196
1         2.4          4                   9.6           221
2         1.5          4                   5.9           136
3         3.5          6                  11.1           255
4         3.5          6                  10.6           244
5         3.5          6                  10.0           230
6         3.5          6                  10.1           232
7         3.7          6                  11.1           255
8         3.7          6                  11.6           267
```

[30]:
```
plt.scatter(regrdata.ENGINESIZE, regrdata.CO2EMISSIONS, color = 'blue')
plt.xlabel = "Engine size"
plt.ylabel = 'Emissions'
plt.show()
```



[31]:
```
tra  = np.random.rand(len(df)) < 0.8
train = regrdata[tra]
test = regrdata[~tra]
test.head()
```

[31]:

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
3	3.5	6	11.1	255
4	3.5	6	10.6	244
8	3.7	6	11.6	267
15	4.7	8	15.4	354
25	2.0	4	10.2	235

[32]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model

train_x = np.asanyarray(train[['ENGINESIZE']])
train_y = np.asanyarray(train[['CO2EMISSIONS']])
test_x = np.asanyarray(test[['ENGINESIZE']])
test_y = np.asanyarray(test[['CO2EMISSIONS']])
polyreg = PolynomialFeatures(degree=3)
trainxpol = polyreg.fit_transform(train_x)
trainxpol
```

[32]:
```
array([[ 1.   ,  2.   ,  4.   ,  8.   ],
       [ 1.   ,  2.4  ,  5.76 , 13.824],
       [ 1.   ,  1.5  ,  2.25 ,  3.375],
```

```
       …,
       [ 1.   ,  3.2  , 10.24 , 32.768],
       [ 1.   ,  3.   ,  9.   , 27.   ],
       [ 1.   ,  3.2  , 10.24 , 32.768]])
```

[33]:
```
lin = linear_model.LinearRegression()
train_y_ = lin.fit(trainxpol, train_y)
print('Coefficients:',  lin.coef_)
print("Intercepts",  lin.intercept_)
```

```
Coefficients: [[ 0.         29.10409899  4.29365483 -0.47114945]]
Intercepts [129.53970799]
```

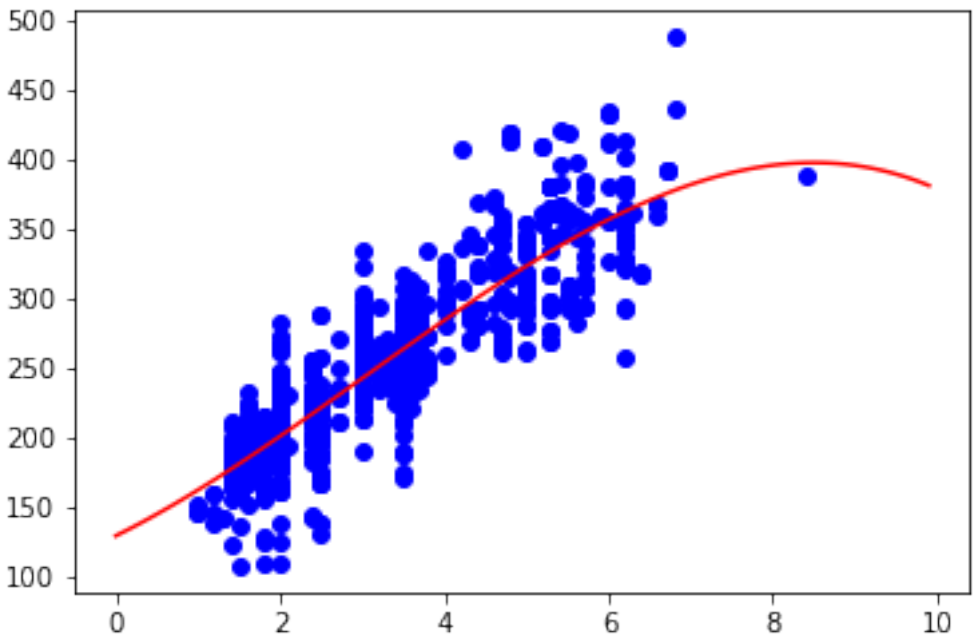
[36]:
```
plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS,  color='blue')
XX = np.arange(0.0, 10.0, 0.1)
yy = lin.intercept_[0]+ lin.coef_[0][1]*XX+ lin.coef_[0][2]*np.power(XX, 2) +␣
 ↪lin.coef_[0][3]*np.power(XX,3)
plt.plot(XX, yy, color = 'red' )
plt.xlabel("Engine size")
plt.ylabel("Emission")
```

```
      ␣
↪---------------------------------------------------------------------------

       TypeError                                 Traceback (most recent call␣
↪last)

       <ipython-input-36-d6f487fa5129> in <module>
         3 yy = lin.intercept_[0]+ lin.coef_[0][1]*XX+ lin.coef_[0][2]*np.
↪power(XX, 2) + lin.coef_[0][3]*np.power(XX,3)
         4 plt.plot(XX, yy, color = 'red' )
   ----> 5 plt.xlabel("Engine size")
         6 plt.ylabel("Emission")


       TypeError: 'str' object is not callable
```



[37]:
```
from sklearn.metrics import r2_score

test_x_poly = poly.fit_transform(test_x)
test_y_ = lin.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) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

```
␣
↪---------------------------------------------------------------------------

ValueError                                Traceback (most recent call␣
↪last)

<ipython-input-37-ba70cd6a2a1d> in <module>
  2
  3 test_x_poly = poly.fit_transform(test_x)
----> 4 test_y_ = lin.predict(test_x_poly)
  5
  6 print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ -␣
↪test_y)))
```

```
~/conda/envs/python/lib/python3.6/site-packages/sklearn/linear_model/
↪base.py in predict(self, X)
      211             Returns predicted values.
      212         """
--> 213         return self._decision_function(X)
      214
      215     _preprocess_data = staticmethod(_preprocess_data)


~/conda/envs/python/lib/python3.6/site-packages/sklearn/linear_model/
↪base.py in _decision_function(self, X)
      196         X = check_array(X, accept_sparse=['csr', 'csc', 'coo'])
      197         return safe_sparse_dot(X, self.coef_.T,
--> 198                                dense_output=True) + self.intercept_
      199
      200     def predict(self, X):


~/conda/envs/python/lib/python3.6/site-packages/sklearn/utils/extmath.py␣
↪in safe_sparse_dot(a, b, dense_output)
      171         return ret
      172     else:
--> 173         return np.dot(a, b)
      174
      175


<__array_function__ internals> in dot(*args, **kwargs)


ValueError: shapes (209,3) and (4,1) not aligned: 3 (dim 1) != 4 (dim 0)
```

[ ]: