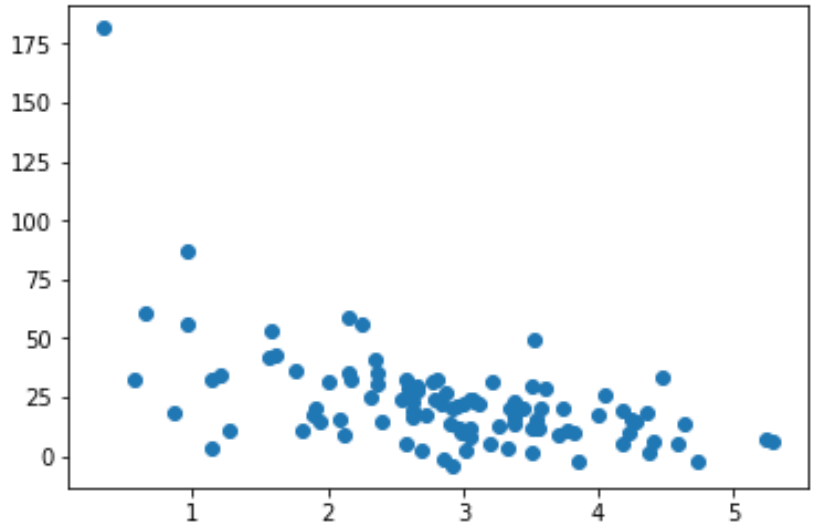
Machine Learning
Lab 2

1. A Simple Example of Supervised Learning

We'll start by creating some data set that we want to build a model for (in this case a polynomial regression):

```
%matplotlib inline
import numpy as np
from pylab import *

np.random.seed(2)

pageSpeeds = np.random.normal(3.0, 1.0, 100)
purchaseAmount = np.random.normal(50.0, 30.0, 100) / pageSpeeds


scatter(pageSpeeds, purchaseAmount)
```

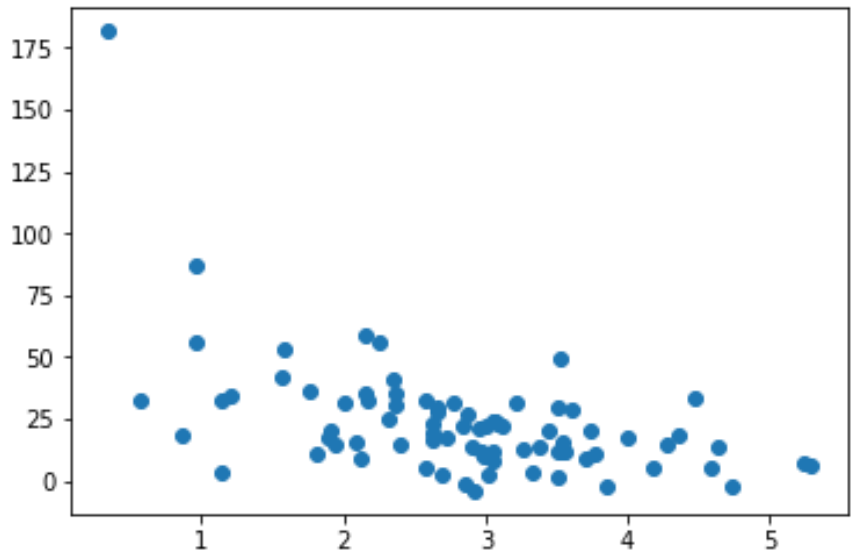


Now we'll split the data in two - 80% of it will be used for "training" our model, and the other 20% for testing it. This way we can avoid overfitting.

```
trainX = pageSpeeds[:80]
testX = pageSpeeds[80:]

trainY = purchaseAmount[:80]
testY = purchaseAmount[80:]
```

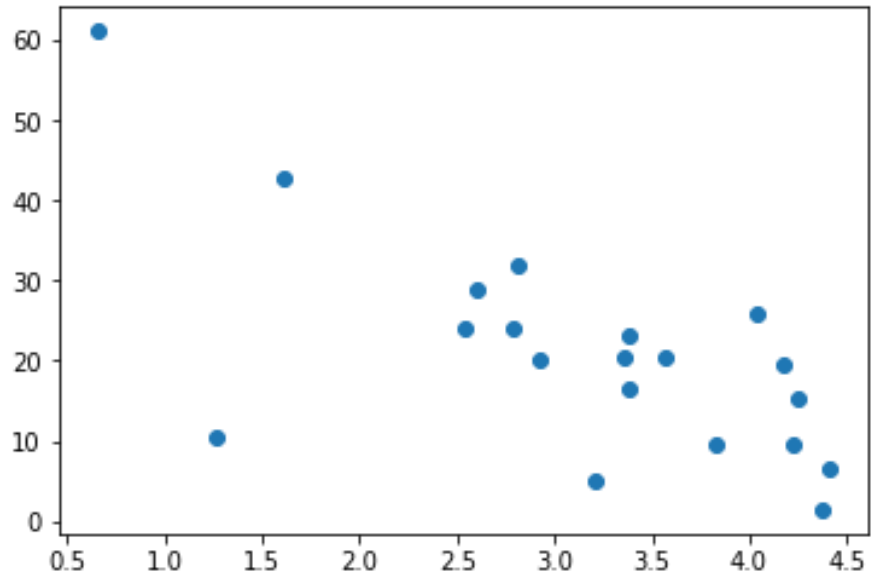
Here's our training dataset:

```
scatter(trainX, trainY)
```



And our test dataset:

```
scatter(testX, testY)
```

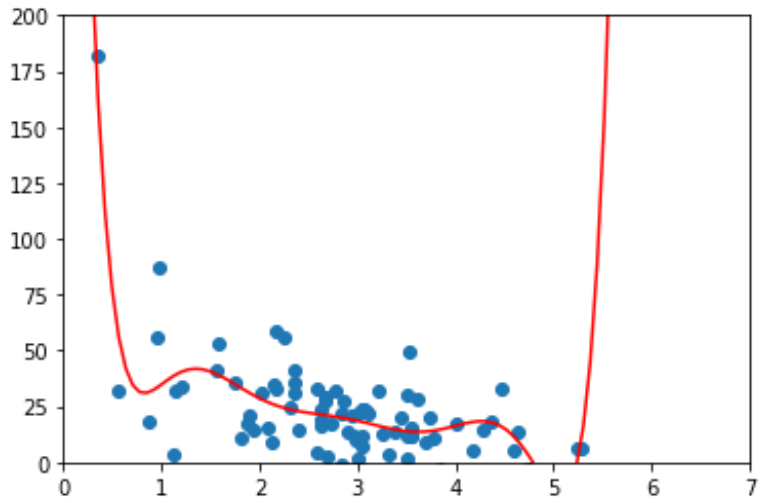


Now we'll try to fit an 8th-degree polynomial to this data (which is almost certainly overfitting, given what we know about how it was generated!)

```
x = np.array(trainX)
y = np.array(trainY)

p4 = np.poly1d(np.polyfit(x, y, 8))
```

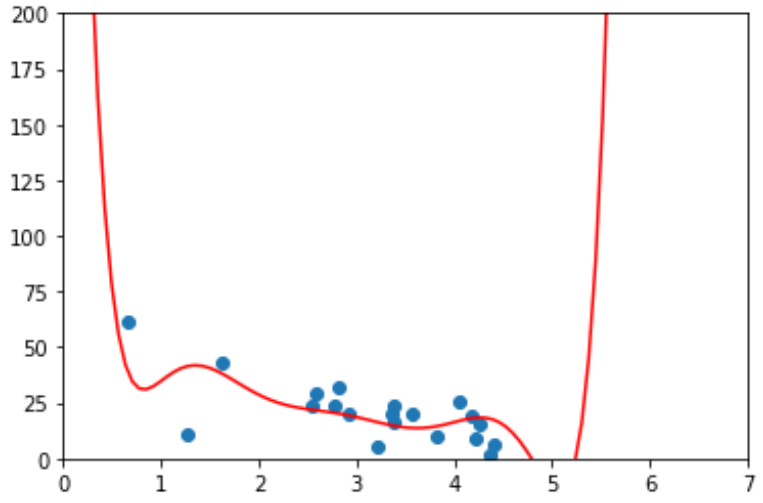
Let's plot our polynomial against the training data:

```
import matplotlib.pyplot as plt

xp = np.linspace(0, 7, 100)
axes = plt.axes()
axes.set_xlim([0,7])
axes.set_ylim([0, 200])
plt.scatter(x, y)
plt.plot(xp, p4(xp), c='r')
plt.show()
```



And against our test data:

```
testx = np.array(testX)
testy = np.array(testY)

axes = plt.axes()
axes.set_xlim([0,7])
axes.set_ylim([0, 200])
plt.scatter(testx, testy)
plt.plot(xp, p4(xp), c='r')
plt.show()
```



Doesn't look that bad when you just eyeball it, but the r-squared score (it is used to measure the model performance we will discuss details in later weeks) on the test data is kind of horrible! This tells us that our model isn't all that great...

```
from sklearn.metrics import r2_score

r2 = r2_score(testy, p4(testx))

print(r2)
```

If you haven't installed sklearn, you will get an error. Here is how to install it:

```
!pip install sklearn
```

The result is 0.30018168612... even though it fits the training data better with r2 of 0.642706951469

```
from sklearn.metrics import r2_score

r2 = r2_score(np.array(trainY), p4(np.array(trainX)))

print(r2)
```

Activity:

- If you're working with a Pandas DataFrame (using tabular, labeled data,) scikit-learn has built-in train_test_split functions to split the train/test data. Search for the function and think of how to use it. We will use it for next section.
- Try measuring the error on the test data using different degree polynomial fits. What degree works best?

2. Lasso Regression

The above example is simple. What if we are facing a more complex problem with multiple inputs?

As we all know, linear regression calculates the coefficients of every variable of the model. As the complexity of the data increases, the value of coefficients turns out to be a higher value which in turn makes the model sensitive to further inputs being fed to it. This in turn makes the model a bit unstable!

So, here we go with the solution. Lasso Regression, also known as L1 regression suffices the purpose. With Lasso regression, we tend to penalize the model against the value of the coefficients. So, it manipulates the loss function by including extra costs for the variables of the model that happens to have a large value of coefficients.

It penalizes the model against absolute coefficient values. By this, it lets the value of the coefficients (that do not contribute to the predictor variable) become zero. Further to this, it removes those input features from the model.

Now, let's look at a more complex problem and try to implement Lasso regression method. The dataset is 'DATA.csv' and you can find it on Moodle next to this tutorial.

Activity:

- Now, search for the Lasso function (available in sklearn), try to develop some codes to build the model for the bike data. The solution will be shared with you in second half of the lab session.

Read the following instructions before you start:

- First, import the data and define X and Y using the codes below. Note that here X and Y is stored as data frame. For the example in Section 1, we use numpy array instead. Remember to use the correct function to process the data.
- Then, develop some codes for splitting of the dataset into train and test data using train_test_split() function.
- Next, build a Lasso regression model.
- Finally, evaluate the model using some performance measure (any measure you prefer).

```
import os
import pandas

DATA = pandas.read_csv("Data.csv")

DATA

#Separating the depenedent and independent data variables into two dataframes.
from sklearn.model_selection import train_test_split
X = DATA.drop(['cnt'],axis=1)
Y = DATA['cnt']

type(X)

X
```