Project 1: Regression Model to Predict Bike Rentals in Washington, DC (10%)
Due before Midnight on February 13

The bike-sharing rental process is highly correlated to the environmental and seasonal settings. For instance, weather conditions, precipitation, day of week, season, hour of the day, etc. can affect the rental behaviors. The core data set for this project is related to the two-year historical log corresponding to years 2011 and 2012 from Capital Bikeshare System, Washington D.C. The data contains conditions (11 x_i's) on a daily basis along with the count of bikes that were rented each day (y).

- season : season (1:springer, 2:summer, 3:fall, 4:winter)
- yr : year (0: 2011, 1:2012)
- mnth : month (1 to 12)
- holiday : whether a day is a holiday or not (0=no, 1 = yes)
- weekday : day of the week (0-6)
- workingday : if day is neither a weekend nor holiday, = 1, otherwise = 0.
- weathersit :
    - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
    - 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
    - 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
    - 4: Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog
- temp : Normalized temperature in Celsius. The values are divided to 41 (max)
- atemp: Normalized feeling temperature in Celsius. The values are divided to 50 (max)
- hum: Normalized humidity. The values are divided to 100 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)
- cnt: count of total rental bikes including both casual and registered

Your assignment is to develop a regression model that will predict how many bikes will be rented on any particular day based on these 11 conditions. Your data set is named BikeAll.txt and is formatted as follows:

first line: integer number of lines of training data (m = 731), tab, integer number of features (n = 11)
all other lines: values of input features, tab separated, followed by the number of bikes rented that day (in order listed above)

Example:

m n
$x^{(1)}_1$ $x^{(1)}_2$ $x^{(1)}_3$ . . . $x^{(1)}_n$ $y^{(1)}$
. . .
$x^{(m)}_1$ $x^{(m)}_2$ $x^{(m)}_3$ . . . $x^{(m)}_n$ $y^{(m)}$

You should develop a model and hypothesis function that provide the best predictions of how many bikes will be rented. In order for us to evaluate your model it must follow strict input and output requirements.
Your Python program must run in Spyder. It should first prompt the use for a Training Set file name that is formatted as:

first line: integer number of lines of training data (m), tab, integer number of features (n)
all other lines: values of input features, tab separated, followed by the number of bikes rented that day (in order listed above)

Next: Prints out values for all weights and final J for the training set. All output should be clearly labelled.

Next: Prompts the user for the name of a validation set. Same file format as the training data.
Next: Prints out J for the validation set. All clearly labelled.

Next: Prompts the user for the name of a test set. Same file format as the training data.
Next: Prints out J for the test set and adjusted R^2 for the test set. All clearly labelled.

Model development.

You should randomize the data file and put 439 data sets in your training set file, 146 data sets in your cross validation set data file and 146 data sets in your test set data file.

You can choose to use the training data features as given or to create additional features. For example, if the original feature data was:

x_1, x_2, x_3

you could square each value to create a new training set consisting of

$x_1, x_1^2, x_2, x_2^2, x_3, x_3^2$

The important thing to remember is that the different model should be represented entirely by your data file! Your program should be constructed so that it will work for any training set that lists, m, n on the first line and data sets on each following line.

What to turn in (uploaded to canvas).

1. A pdf file containing:
    - A table describing your different models and giving your J values for test and validation sets for all models, and J and R^2 values for the test set for the best model.
    - A list of your final weights.

Example of table:

Model	J_{train}	$J_{validation}$	J_{test}/R^2_{test}
Linear (n=11)	440,000	460,000	
Quadratic (n=22)	290,000	295,000	296,000/0.84
Other models			

2. Your Python program named: yourlastname_yourfirstname_P1.py
3. Your Final Training set named: yourlastname_yourfirstname_Train.txt
4. Your Final Validation set named: yourlastname_yourfirstname_Valid.txt
5. Your Final Test set named: yourlastname_yourfirstname_Test.txt

Grading

Pdf file with all information included	+20	+20
All py and text files turned in with all information and correctly named	+20	+20
Validation set output	CPSC 4820	CPSC 6820
J < 450,000	+55	+45
J < 300,000	+60	+55
J < 250,000	+65	+65
Program throws an error	-65	-65
Program does not prompt for training set	-10	-10
Program does not prompt for the validation set	-10	-10
Program does not prompt for test set	-10	-10
Program does not print out weights and J for the training data.	-15	-15
Program does not print out J for the validation set	-10	-10
Program does not print out J and Adjusted R^2 for the test set	-10	-10
Data not clearly labelled.	-10	-10
Late submissions	-3^{n-1}	-3^{n-1}

I suggest implementing this program using the Normal Equation instead of Gradient Descent, but either are acceptable.