Regression Week 1: Simple Linear Regression Assignment

Predicting House Prices (One feature)

In this notebook we will use data on house sales in King County, where Seattle is located, to predict house prices using simple (one feature) linear regression. You will:

- Use SArray and SFrame functions to compute important summary statistics
- Write a function to compute the Simple Linear Regression weights using the closed form solution
- Write a function to make predictions of the output given the input feature
- Turn the regression around to predict the input/feature given the output
- Compare two different models for predicting house prices

If you are doing the assignment with IPython Notebook

An IPython Notebook has been provided below to you for this assignment. This notebook contains the instructions, quiz questions and partially-completed code for you to use as well as some cells to test your code.

What you need to download

If you are using GraphLab Create:

- Download the King County House Sales data In SFrame format: kc_house_data.gl.zip
- Download the companion IPython Notebook: week-1-simple-regression-quiz-blank.ipynb
- Save both of these files in the same directory (where you are calling IPython notebook from) and unzip the data file.

If you are not using GraphLab Create:

- Download the King County House Sales data csv file: kc_house_data.csv

- Download the King County House Sales training data csv file: kc_house_train_data.csv
- Download the King County House Sales testing data csv file: kc_house_test_data.csv

Useful resources

You may need to install the software tools or use the free Amazon EC2 machine. Instructions for both options are provided in the reading for Module 1.

If instead you are using other tools to do your homework

You are welcome, however, to write your own code and use any other libraries, like Pandas or R, to help you in the process. If you would like to take this path, follow the instructions below.

1. If you are using SFrame, import graphlab and load in the house data, otherwise you can also download the csv. (Note that we will be using the training and testing csv files provided). e.g in python with SFrames:

```
sales = graphlab.SFrame('kc_house_data.gl/')
```

2. Split data into 80% training and 20% test data. Using SFrame, use this command to set the same seed for everyone. e.g. in python with SFrames:

```
train_data,test_data = sales.random_split(.8,seed=0)
```

For those students not using graphlab please download the training and testing data csv files.

From now on we will train the models using train_data. It will be important that we use the same split here to ensure the results are the same.

3. Write a generic function that accepts a column of data (e.g, an SArray) 'input_feature' and another column 'output' and returns the Simple Linear Regression parameters 'intercept' and 'slope'. Use the closed form solution from lecture to calculate the slope and intercept. e.g. in python:

```
def simple_linear_regression(input_feature, output):
    [your code here]
return(intercept, slope)
```

4. Use your function to calculate the estimated slope and intercept on the training data to predict 'price' given 'sqft_living'. e.g. in python with SFrames using:

```
input_feature = train_data[‘sqft_living’]
output = train_data[‘price’]
```

save the value of the slope and intercept for later (you might want to call them e.g. squarfeet_slope, and squarefeet_intercept)

5. Write a function that accepts a column of data ‘input_feature’, the ‘slope’, and the ‘intercept’ you learned, and returns an a column of predictions ‘predicted_output’ for each entry in the input column. e.g. in python:

```
def get_regression_predictions(input_feature, intercept, slope)
    [your code here]
return(predicted_output)
```

6. Quiz Question: Using your Slope and Intercept from (4), What is the predicted price for a house with 2650 sqft?

7. Write a function that accepts column of data: ‘input_feature’, and ‘output’ and the regression parameters ‘slope’ and ‘intercept’ and outputs the Residual Sum of Squares (RSS). e.g. in python:

```
def get_residual_sum_of_squares(input_feature, output, intercept,slope):
    [your code here]
return(RSS)
```

Recall that the RSS is the sum of the squares of the prediction errors (difference between output and prediction).

8. Quiz Question: According to this function and the slope and intercept from (4) What is the RSS for the simple linear regression using squarefeet to predict prices on TRAINING data?

9. Note that although we estimated the regression slope and intercept in order to predict the output from the input, since this is a simple linear relationship with only two variables we can invert the linear function to estimate the input given the output!

Write a function that accept a column of data:‘output’ and the regression parameters ‘slope’ and ‘intercept’ and outputs the column of data: ‘estimated_input’. Do this by solving the linear function output = intercept + slope*input for the ‘input’ variable (i.e. ‘input’ should be on one side of the equals sign by itself). e.g. in python:

```
def inverse_regression_predictions(output, intercept, slope):
    [your code here]
```

```
return(estimated_input)
```

10. Quiz Question: According to this function and the regression slope and intercept from (3) what is the estimated square-feet for a house costing $800,000?

11. Instead of using ‘sqft_living’ to estimate prices we could use ‘bedrooms’ (a count of the number of bedrooms in the house) to estimate prices. Using your function from (3) calculate the Simple Linear Regression slope and intercept for estimating price based on bedrooms. Save this slope and intercept for later (you might want to call them e.g. bedroom_slope, bedroom_intercept).

12. Now that we have 2 different models compute the RSS from BOTH models on TEST data.

13. Quiz Question: Which model (square feet or bedrooms) has lowest RSS on TEST data? Think about why this might be the case.