Predicting house prices

In this module, we focused on using regression to predict a continuous value (house prices) from features of the house (square feet of living space, number of bedrooms,...). We also built an iPython notebook for predicting house prices, using data from King County, USA, the region where the city of Seattle is located.

In this assignment, we are going to build a more accurate regression model for predicting house prices by including more features of the house. In the process, we will also become more familiar with how the Python language can be used for data exploration, data transformations and machine learning. These techniques will be key to building intelligent applications.

Follow the rest of the instructions on this document to complete your program. When you are done, ***you will upload your notebook on GradeScope,*** to document your completion of this assignment.

Learning outcomes

- Execute programs with the iPython notebook
- Load and transform real, tabular data
- Compute summaries and statistics of the data
- Build a regression model using features of the data

Resources you will need

Download the data and starter code to use GraphLab Create

Before getting started, you will need to download the dataset and the starter iPython notebook that we used in the module.

- Download the house sales pricing dataset here, in SFrame format: **home_data.gl.zip**
- Download the house price prediction notebook from the module here: **Predicting house prices.ipynb**
- Save both of these files in the same directory (where you are calling iPython notebook from) and unzip the data file.
- Note that there is a bug in GraphLab Create 1.6.0, where the scatter plots don't show up in the notebook. Please upgrade to a newer version, if you have 1.6.0.

Now you are ready to get started!

What you will do

Now you are ready! We are going do three tasks in this assignment. There are 3 results you need to gather along the way to complete after this reading.

1. Selection and summary statistics: In the notebook we covered in the module, we discovered which neighborhood (zip code) of Seattle had the highest average house sale price. Now, take the sales data, select only the houses with this zip code, and compute the average price. ***Save this result at the end of the notebook as ANSWER 1.***

***2.* Filtering data:** One of the key features we used in our model was the number of square feet of living space ('sqft_living') in the house. For this part, we are going to use the idea of filtering (selecting) data.

- In particular, we are going to use logical filters to select rows of an SFrame. You can find more info in the Logical Filter section of this documentation.
- Using such filters, first select the houses that have 'sqft_living' higher than 2000 sqft but no larger than 4000 sqft.
- What fraction of the all houses have 'sqft_living' in this range? ***Save this result at the end of the notebook as ANSWER 2.***

3. Building a regression model with several more features: In the sample notebook, we built two regression models to predict house prices, one using just 'sqft_living' and the other one using a few more features, we called this set

```
my_features = ['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors'
,'zipcode']
```

Now, going back to the original dataset, you will build a model using the following features:

```
advanced_features = [

'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'zipcode',

'condition', # condition of house

'grade', # measure of quality of construction

'waterfront', # waterfront property

'view', # type of view

'sqft_above', # square feet above ground

'sqft_basement', # square feet in basement
```

```
'yr_built', # the year built

'yr_renovated', # the year renovated

'lat', 'long', # the lat-long of the parcel

'sqft_living15', # average sq.ft. of 15 nearest neighbors

'sqft_lot15', # average lot size of 15 nearest neighbors

]
```

Note that using copy and paste from this document to the IPython Notebook sometimes does not work perfectly in some operating systems, especially on Windows. For example, the quotes defining strings may not paste correctly. Please check carefully is you use copy & paste.

- **Compute the RMSE** (root mean squared error) on the test_data for the model using just *my_features*, and for the one using *advanced_features*.

Note 1: both models must be trained on the original sales dataset, not the filtered one.

Note 2: when doing the train-test split, make sure you use seed=0, so you get the same training and test sets, and thus results, as we do.

Note 3: in the module we discussed residual sum of squares (RSS) as an error metric for regression, but GraphLab Create uses root mean squared error (RMSE). These are two common measures of error regression, and RMSE is simply the square root of the mean RSS:

$$RMSE = \sqrt{\frac{RSS}{N}}$$

where N is the number of data points. RMSE can be more intuitive than RSS, since its units are the same as that of the target column in the data, in our case the unit is dollars ($), and doesn't grow with the number of data points, like the RSS does.

(Important note: when answering the question below using GraphLab Create, when you call the ***linear_regression.create()*** function, make sure you use the parameter ***validation_set=None***, as done in the notebook you download above. When you use regression GraphLab Create, it sets aside a small random subset of the data to validate some parameters. This process can cause fluctuations in the final RMSE, so we will avoid it to make sure everyone gets the same answer.)

- **What is the difference in RMSE between the model trained with my_features and the one trained with advanced_features? *Save this result at the end of the notebook as ANSWER 3.***