# Laboratory Exercise 1
# FRA 311 Artificial Intelligence for Robotics and Automation
# House Price Prediction

Warasinee Chaisangmongkon, PhD

March 17, 2018

## 1 Objective

This lab exercise is designed to teach a few basic skills and concepts:

a. Learn a few useful commands of Pandas

b. Implement a few techniques of data preprocessing

c. Learn a few commands of scikit-learn

d. Learn how to fit a regression model

e. Get a feel of real machine learning problem

Note that we intentionally do not include any specific instruction or a command suggestion in this exercise because half of machine learning job (or any programming job in general) is googling the right command to use. I encourage you to google extensively how to do things. It's the only way to learn things as fast as possible.

Please refer to Machine Learning in Practice video for more guidance.

Please complete this lab exercise in Jupyter Notebook and submit the ipynb, .html, and pdf file to Google Classroom. This is the only way for your notebook to be graded.

## 2 Preparation

In this exercise, we will all join millions other machine learning engineers on Kaggle and have fun with the free datasets they have available. I have selected

this particular dataset for this exercise:

https://www.kaggle.com/c/house-prices-advanced-regression-techniques

You need to sign up to download the dataset or you can download from Google Classroom.

## 3 Exploring data tables with Pandas

Pandas (http://pandas.pydata.org/) is a very popular data manipulation library. It automates data-wrangling task such as reading and writing csv files, sorting and filtering data tables, merging and grouping data, and much more. If you wish to get a quick syntax introduction, I recommend scrolling through this page:

https://bitbucket.org/hrojas/learn-pandas

a. Use Pandas to read the house prices data. How many columns and rows are there in this dataset?

b. The first step I usually do is to use commands like `pandas.head()` to print a few rows of data. Look around what kind of features are available and read `data_description.txt` for more info. Try to understand as much as you can. Pick three features you think will be good predictors of house prices and explain what they are.

c. How many unique conditions are there in `SaleCondition`? Use Pandas to find out how many samples are labeled with each condition. What do you learn from doing this?

d. Select one variable you picked in b., do you want to know something more about that variable? Use Pandas to answer your own question and describe what you did shortly here.

## 4 Learning to explore data with Seaborn

Seaborn feels like magic sometimes, because it allows you to make beautiful plots with a line of code. So let's see how Seaborn can help us explore the house prices data.

Use this reference page to get started:

http://seaborn.pydata.org/tutorial/distributions.html

a. Let us first look at the variable we want to predict `SalePrice`. Use Seaborn to plot histogram of sale prices. What do you notice in the histogram?

b. Plot the histogram of the `LotArea` variable. What do you notice in the histogram?

c. Use Seaborn to plot `LotArea` in the x-axis and `SalePrice` on the y-axis.

d. Try plotting `log(LotArea)` versus `log(SalePrice)` and see if the plot looks better.

# 5 Dealing with missing values

Suppose we want to start the first step of house price modeling by exploring the relationship between four variables: `MSSubClass`, `LotArea`, `LotFrontage` and `SalePrice`. I have done some exploring and found out that `LotFrontage` has a lot of missing values, so you need to fix it.

a. `LotFrontage` is the width of the front side of the property. Use Pandas to find out how many of the houses in our database is missing `LotFrontage` value.

b. Use Pandas to replace `NaN` values with another number. Since we are just exploring and not modeling yet, you can simply replace `NaN` with zeros for now.

# 6 Correlations between multiple variables

One incredible feature of Seaborn is the ability to create correlation grid with `pairplot` function. We want to create one single plot that show us how all variables are correlated.

a. First, you need to create a data table with four columns: `MSSubClass`, `LotArea` (with log function applied), `LotFrontage` (missing values replaced) and `SalePrice` (with log function applied).

b. Then, use `pairplot` to create a grid of correlation plots. What do you observe from this plot?

# 7 Data Preparation

The very first step would be to run some preprocessing. We will try to keep this as simple as possible.

Let us first focus on the train.csv file. Here are a few steps I recommend in order to prepare the train data to be ready for modeling.

a. Let's read `train.csv` using `pandas.read_csv()`. You can use `head()` to see some example data rows.

b. Pick columns that are numeric data and plot distributions of those data (with Seaborn). If you find a column with skewed distribution you will write a script to transform that column with a log or boxcox function. Then standardize them.

c. For categorical variables, we will simply transform categorical data into numeric data by using function `pandas.get_dummies()`.

d. For simplicity, let us replace missing values in each column with the mean of that column, using `pandas.fillna()`.

e. Split data into `x` and `y`. The variable `x` contains all the house features except the SalePrice. `y` contains only the SalePrice.

# 8 Fitting Basic Linear Regression

Let us first fit a very simple linear regression model, just to see what we get.

a. Use import LinearRegression from `sklearn.linear_model` and use function `fit()` to fit the model.

b. Use function `predict()` to get house price predictions from the model (let's call the predicted house prices `yhat`).

c. Plot `y` against `yhat` to see how good your predictions are.

If you are unsure about how sklearn works, this example should help:
`http://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html`

# 9 Assessing Your Model

According to Kaggle's official rule on this problem, they use root mean square errors (rmse) to judge the accuracy of our model. This error computes the difference between the log of actual house prices and the log of predicted house price. Find the mean and squareroot them.

We want to see how we compare to other machine learning contestants on Kaggle so let us compute our rmse. Luckily, sklearn has done most of the work for you by providing `mean_square_error` function. You can use it by importing the function from `sklearn.metrics`. Then, you can compute `mean_square_error` and take a squareroot to get rmse.

What's the rmse of your current model? Check out Kaggle Leaderboard for this problem to see how your number measures up with the other contestants.

## 10 Cross Validation

As we discussed earlier, don't brag about your model's accuracy until you have performed cross validation. Let us check cross-validated performance to avoid embarrassment.

Luckily, scikit learn has done most of the work for us once again. You can use the function `cross_val_predict()` to train the model with cross validation method and output the predictions.

What's the rmse of your cross-validated model? Discuss what you observe in your results here. You may try plotting this new `yhat` with `y` to get better insights about this question.

## 11 (Optional) Fit Better Models

There are other models you can fit that will perform better than linear regression. For example, you can fit linear regression with L2 regularization. This class of models has a street name of 'Ridge Regression' and sklearn simply called them `Ridge`. As we learned last time, this model will fight overfitting problem. Furthermore, you can try linear regression with L1 regularization (street name Lasso Regression or `Lasso` in sklearn). Try these models and see how you compare with other Kagglers now. You can write about your findings below.