Applied Machine Learning
Assignment 1

Name	YAP LI XEN
Student ID	P7414389
Lecturer	MR. PAUL

PART B: REGRESSION

(1) How is your prediction task defined? And what is the meaning of the output variable?

- The prediction task is to predict numerical data "price" (house price)
- The output variables "price" is continuous numerical data

(2) How do you represent your data as features?

- Target: price
- Features: All columns except price

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_living15	sqft_lot15
7129300520	20141013T000000	221900	3	1	1180	5650	1	0	0	3	7	1180	0	1955	0	98178	47.5112	-122.257	1340	5650
6414100192	20141209T000000	538000	3	2.25	2570	7242	2	0	0	3	7	2170	400	1951	1991	98125	47.721	-122.319	1690	7639

(3) Did you process the features in any way?

- Feature Selection

To remove features that doesn't bring impact or less important to the prediction

```
df = df.drop(['id','date','sqft_above','sqft_living15','sqft_lot15'], 1)
print(df.head())
```

```
      price  bedrooms  bathrooms  sqft_living  sqft_lot  floors  waterfront  \
0  221900.0         3       1.00         1180      5650     1.0           0
1  538000.0         3       2.25         2570      7242     2.0           0
2  180000.0         2       1.00          770     10000     1.0           0
3  604000.0         4       3.00         1960      5000     1.0           0
4  510000.0         3       2.00         1680      8080     1.0           0

   view  condition  grade  sqft_basement  yr_built  yr_renovated  zipcode  \
0     0          3      7              0      1955             0    98178
1     0          3      7            400      1951          1991    98125
2     0          3      6              0      1933             0    98028
3     0          5      7            910      1965             0    98136
4     0          3      8              0      1987             0    98074

       lat     long
0  47.5112 -122.257
1  47.7210 -122.319
2  47.7379 -122.233
3  47.5208 -122.393
4  47.6168 -122.045
```

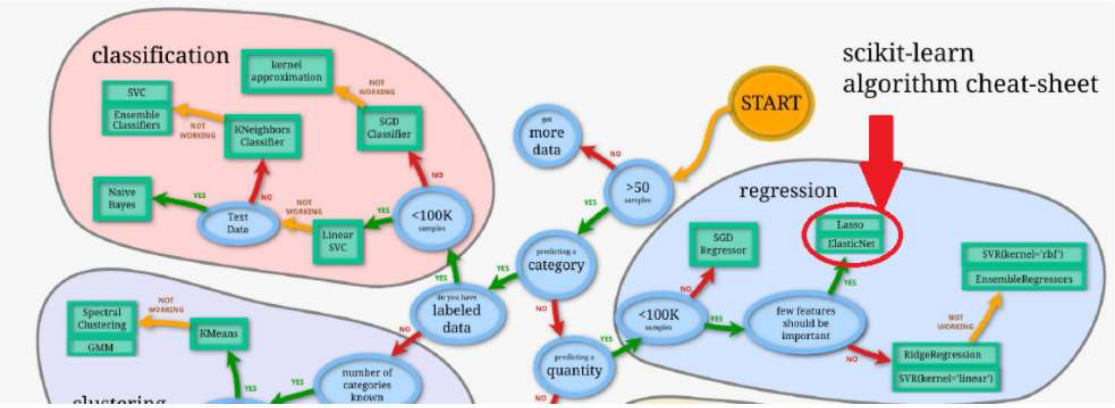
(4) Did you bring in any additional sources of data?

- Nope. Not necessary at this stage

(5) How did you select which learning algorithms to use?

- Referring to scikit-learn algorithm cheat-sheet recommendation

Lasso is selected as training model following the cheat-sheet in our use case



(6) Did you try to tune the hyperparameters of the learning algorithm, and in that case how?

- Hyperparameter Tuning: GridSearchCV
- To include a range of values in chosen parameter and assign to param grid
- To set cv to 5 (split the train-validation data into 5 folds)

To use GridSearchCV which includes Cross Validation to identify best paramter and best score.

```
from sklearn.model_selection import GridSearchCV

param_grid = {'alpha': [0.001,0.01,0.1,1.0,10.0,100.0,1000.0]}
grid_search = GridSearchCV(model, param_grid=param_grid, cv=3, verbose=3, return_train_score=True)
grid_search.fit(X_train, y_train);
```

- Then it will generate the best param and best score

```
print("Best Param: {}".format(grid_search.best_params_))
print("Best Score: {:.2f}%".format(grid_search.best_score_*100))
```

```
Best Param: {'alpha': 0.001}
Best Score: 69.63%
```

(7) How do you evaluate the quality of your system?

- Score the trained model using both train data and test data, then compare the result
- We can see how well the trained model can predict the test data
- The comparison can also help us to determine if this is under-fitting, appropriate-fitting or over-fitting

```
training_data_score = model.score(X_train, y_train)
print("Training Data Score: {:.2f}%".format(training_data_score*100))

test_data_score = model.score(X_test, y_test)
print("Test Data Score: {:.2f}%".format(test_data_score*100))
```

```
Training Data Score: 69.83%
Test Data Score: 69.99%
```

(8) How well does your system compare to a stupid baseline?

-

(9) Can you say anything about the errors that the system makes?

- Using R2 score, it computes the coefficient of determination
- Using Mean squared error

```
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

# Predict Train Data
y_predict_train = model.predict(X_train)
r2_train = r2_score(y_train, y_predict_train)
mse_train = mean_squared_error(y_train, y_predict_train)

# Predict Test Data
y_predict_test = model.predict(X_test)
r2_test = r2_score(y_test, y_predict_test)
mse_test = mean_squared_error(y_test, y_predict_test)

print("R2 Score (Train Data): {:.2f}".format(r2_train))
print("R2 Score (Test Data): {:.2f}".format(r2_test))

print("MSE Score (Train Data): {:.2f}".format(mse_train))
print("MSE Score (Test Data): {:.2f}".format(mse_test))
```

```
R2 Score (Train Data): 0.70
R2 Score (Test Data): 0.70
MSE Score (Train Data): 39414224347.30
MSE Score (Test Data): 45362639291.37
```

(10) Is it possible to say something about which features the model considers important?

- Information below shows which features are correlation to the target.

```
corr = df.corr()
corr.sort_values(["price"], ascending = False, inplace = True)
print(corr["price"])
```

```
price            1.000000
sqft_living      0.702035
grade            0.667434
sqft_above       0.605567
sqft_living15    0.585379
bathrooms        0.525138
view             0.397293
sqft_basement    0.323816
bedrooms         0.308350
lat              0.307003
waterfront       0.266369
floors           0.256794
yr_renovated     0.126434
sqft_lot         0.089661
sqft_lot15       0.082447
yr_built         0.054012
condition        0.036362
long             0.021626
id              -0.016762
zipcode         -0.053203
Name: price, dtype: float64
```