

### Question 1

What is the optimal value of alpha for ridge and lasso regression? What will be the changes in the model if you choose double the value of alpha for both ridge and lasso? What will be the most important predictor variables after the change is implemented?

Answer:

According to the model we build, the optimal value for Lasso Regression is 500(alpha) and for Ridge regression it is 100(alpha).

Following are the changes if we double the alpha values for both Lasso and Ridge regression Lasso Regression:

A) Lasso : when value of alpha = 50, random\_state = 1, train\_size = 0.70 , test\_size = 0.30 and for dataset metrics evaluation:

Train:

Accuracy : 0.9144787020519137  
MAE : 10260.993681103622  
MSE : 199915386.0244316  
RMSE : 14139.143751459336

Test:

Accuracy : 0.8853131545577257  
MAE : 11262.130614267224  
MSE : 222423064.553881  
RMSE : 14913.854785194906

Alpha double for Lasso : when value of alpha = 100, random\_state = 1, train\_size = 0.70 , test\_size = 0.30 and for dataset metrics evaluation:

Train:

Accuracy : 0.9094307738255077  
MAE : 10517.26698454338  
MSE : 208204650.19718695  
RMSE : 14429.298326571079

Test:

Accuracy : 0.8877239403313943  
MAE : 11258.975092172903  
MSE : 222178167.36273336  
RMSE : 14905.642131848375

Results:

- What we observe from here is that accuracy of the model is decreasing for train dataset and slightly decreasing for test dataset.

- And also for lasso the coefficients are tending to zero (decreases) as we increase the value of Alpha.

B) Ridge : when value of alpha = 5, random\_state = 1, train\_size = 0.70 , test\_size = 0.30 and for dataset metrics evaluation:

Train:

Accuracy : 0.9136393861126622  
 MAE : 10177.692177992576  
 MSE : 198529701.14358273  
 RMSE : 14090.05681832343

Test:

Accuracy : 0.8878416535455262  
 MAE : 11311.100583787154  
 MSE : 221945229.84900698  
 RMSE : 14897.826346450915

Alpha double for Ridge : when value of alpha =5, random\_state = 1, train\_size = 0.70 , test\_size = 0.3 and for dataset metrics evaluation:

Train:

Accuracy : 0.9117202670207247  
 MAE : 10329.578398203326  
 MSE : 202941459.2660796  
 RMSE : 14245.75232362544

Test:

Accuracy : 0.8866653756566039  
 MAE : 11400.127899150642  
 MSE : 224272914.54369143  
 RMSE : 14975.744206672716

- What we observe from here is that the test accuracy of the model is slightly decreasing for the dataset.
- As we increase the value of alpha, model complexity decreases.

Question 2:

You have determined the optimal value of lambda for ridge and lasso regression during the assignment. Now, which one will you choose to apply and why?

Answer:

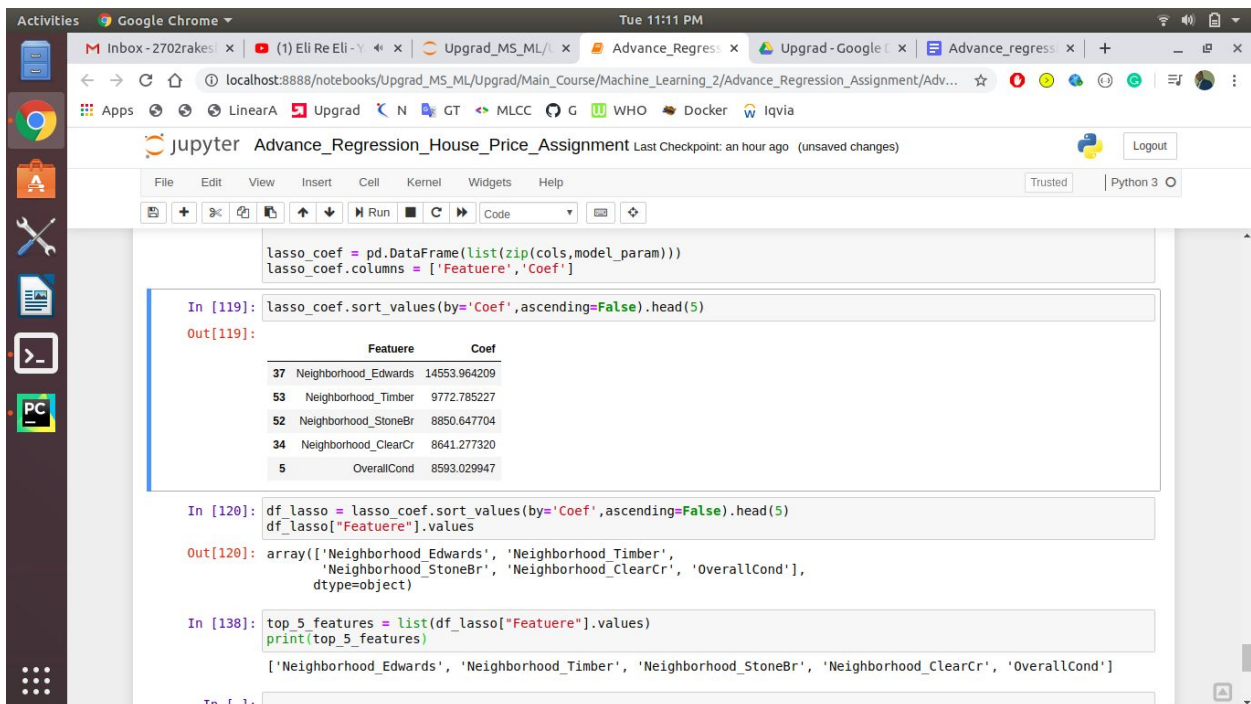
We would choose Lasso Regression as the lasso method is better than Ridge in terms of not only punishing high values of coefficients but setting them to zero equivalent.

Question3:

After building the model, you realized that the five most important predictor variables in the lasso model are not available in the incoming data. You will now have to create another model excluding the five most important predictor variables. Which are the five most important predictor variables now?

Answer:

Top five important predictor using Lasso regression from the model is as follows:



```
lasso_coef = pd.DataFrame(list(zip(cols,model_param)))
lasso_coef.columns = ['Featuree','Coef']

In [119]: lasso_coef.sort_values(by='Coef',ascending=False).head(5)
Out[119]:
```

	Featuree	Coef
37	Neighborhood_Edwards	14553.964209
53	Neighborhood_Timber	9772.785227
52	Neighborhood_StoneBr	8850.647704
34	Neighborhood_ClearCr	8641.277320
5	OverallCond	8593.029947

```
In [120]: df_lasso = lasso_coef.sort_values(by='Coef',ascending=False).head(5)
df_lasso["Featuree"].values
Out[120]: array(['Neighborhood_Edwards', 'Neighborhood_Timber',
'Neighborhood_StoneBr', 'Neighborhood_ClearCr', 'OverallCond'],
dtype=object)

In [138]: top_5_features = list(df_lasso["Featuree"].values)
print(top_5_features)
['Neighborhood_Edwards', 'Neighborhood_Timber', 'Neighborhood_StoneBr', 'Neighborhood_ClearCr', 'OverallCond']

In [ ]:
```

Top 5 :

['Neighborhood\_Edwards', 'Neighborhood\_Timber', 'Neighborhood\_StoneBr', 'Neighborhood\_ClearCr', 'OverallCond']

Removing these columns from the dataset and performing same LASSO regression steps which are present in the Jupyter notebook attached stepwise we can see that new Top 5 predictor we get are as follows :

```
In [127]: model_param_new = list(lasso_new.coef_)
model_param_new.insert(0, lasso_new.intercept_)

cols_new = X_new.columns
cols_new.insert(0, 'const')

lasso_coef_new = pd.DataFrame(list(zip(cols_new, model_param_new)))
lasso_coef_new.columns = ['Feature New', 'Coef New']

In [128]: lasso_coef_new = lasso_coef_new.sort_values(by='Coef New', ascending=False).head(5)
lasso_coef_new

Out[128]:
```

	Feature New	Coef New
35	Neighborhood_Gilbert	17180.442210
33	Neighborhood_CollgCr	12393.440361
5	YearBuilt	9825.571186
44	Neighborhood_OldTown	8542.299633
49	Neighborhood_Veenker	8330.003690

```
In [139]: top_5_features_new = list(lasso_coef_new["Feature New"].values)
print(top_5_features_new)

['Neighborhood_Gilbert', 'Neighborhood_CollgCr', 'YearBuilt', 'Neighborhood_OldTown', 'Neighborhood_Veenker']

In [ ]:
```

After Remove : top 5

['Neighborhood\_Gilbert', 'Neighborhood\_CollgCr', 'YearBuilt', 'Neighborhood\_OldTown', 'Neighborhood\_Veenker']

#### Question 4

How can you make sure that a model is robust and generalisable? What are the implications of the same for the accuracy of the model and why?

Answer:

Robustness is the property that is tested on a training sample and on a similar testing sample, the performance and Accuracy are close for both algorithms(Lasso & Ridge). By the means of regularization we can control the tradeoff between Model complexity and bias which is directly connected to the robustness of the model. Penalizing the coefficients for making the model too complex but just allowing the appropriate amount of complexity controls the robustness of the model.

Accuracy and robustness may be at the odds to each other as too much accurate model can be prey to over fitting hence it can be too much accurate on train data but fails when it faces the actual data or vice versa.