Description Questions

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

The Optimum value of alpha for Ridge and Lasso are 4 and 100 Respectively.

When we double the Alpha values for both Ridge and Lasso Model then,

- We can Observe that for both Models the R2 Score on Test and train decreased which is a sign of underfitting .
- We can also Observe that for both models the RSS and MSE value is Increased which is also a sign of underfitting .
- We can also Observe that the coefficients of the predictor's changed a lot.

- Especially In lasso Model, We can Clearly see that when we Increased the Alpha value, The no of variables which coefficients are equal to zero Increased from 142 to 163.
- So model excluded 21 variables when alpha becomes doubled (which is useful for feature selection).

Note : *Full code for this is attached at bottom.*

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

- I personally choose Lasso Regression Because the R2 Score are slightly high and also RSS and MSE are also less compared to Ridge.
- When it comes to Lambda (alpha) values I choose the First alpha values 4,100 for Ridge and lasso Respectively , Because when I choose the Doubled Alpha values the R2 Score on Test and train decreased and also RSS and MSE value is Increased.

Question 3

After building the model, you realised 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

1) Lets start with Ridge Model

- The top 5 important Features in ridge model are 'GrLivArea', '1stFlrSF', 'OverallQual', 'BsmtFinSF1', 'TotalBsmtSF'.
- Now Lets drop these features and we will build the model again .
- The new Top 5 features are 'TotRmsAbvGrd', 'Neighborhood_StoneBr', '2ndFlrSF', 'FullBath', 'GarageArea'.
- *We can Clearly Observe that the R2 score of the test data set decreased compared to old model.*

2) For Lasso Model.

- The top 5 important Features in Lasso model are 'GrLivArea', 'OverallQual', 'BsmtFinSF1', 'TotalBsmtSF', 'Neighborhood_StoneBr'.
- Now Lets drop these features and we will build the model again .
- The new Top 5 features are '1stFlrSF', '2ndFlrSF', 'YearBuilt', 'OverallCond', 'KitchenQual_TA'.
- We can Clearly Observe that the R2 score of the test data set is Slightly decreased compared to old model.

Note : *Full code for this is attached at bottom.*

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

1. The model should be generalized so that the accuracy of the test Should not be much less than the training points so that the model will be giving similar accuracy For Other data sets too.
2. We should also take care that our model should not be affected by Outliers, And should not given more importance to the column containing outliers which leads to overfit , So we should delete Outliers before training the model.
3. The Model Should have high R^2 Score which makes sure that our model covered the most of the variance from the data. so that even any variation in the data should not effect the model much.
4. To make sure the model is robust and generalizable, we must be careful not to overfit it. This is because the overfit model has very high variability and a small change in the data greatly affects the model prediction. Such a model would identify all training data patterns, but fail to select patterns in unseen test data.
5. In other words, the model should not be too complex in order to be robust and generalizable and also Should not be too simple to underfit.
6. If we look at it from the perspectives of Accuracy, the most complex model will have the highest accuracy. Therefore, in order to make our model more robust and generalizable, we will need to reduce the variance that will lead to certain biases. Increased bias means that accuracy will decrease by a little bit. So we need to sacrifice some variance and accuracy for the bias.
7. In general, we should find a balance between model accuracy and complexity. This can be achieved with strategies such as Ridge Regression and Lasso.

Description Questions

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

The Optimum value of alpha for Ridge and Lasso are 4 and 100 Respectively.

In [68]:
```
# Double the alpha for Ridge Regression from 4 to 8
alpha = 8
ridge2 = Ridge(alpha=alpha)

ridge2.fit(X_train, y_train)
```

Out[68]:
```
Ridge(alpha=8)
```

In [69]:
```
# Lets calculate some metrics such as R2 score, RSS and RMSE
y_pred_train = ridge2.predict(X_train)
y_pred_test = ridge2.predict(X_test)

metric1 = []
r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric1.append(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)
metric1.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
metric1.append(rss1_lr)

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(rss2_lr)
metric1.append(rss2_lr)

mse_train_lr = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric1.append(mse_train_lr**0.5)

mse_test_lr = mean_squared_error(y_test, y_pred_test)
print(mse_test_lr)
metric1.append(mse_test_lr**0.5)
```

```
0.9323807410178085
0.9108652746085143
239675756744.6213
159584679728.3385
315362837.82187015
628286140.66275
```

In [70]:
```
# Double the alpha for Ridge Regression from 100 to 200
alpha =200
```

```
lasso2 = Lasso(alpha=alpha)
lasso2.fit(X_train, y_train)
```

Out[70]: Lasso(alpha=200)

In [71]:

```
# Lets calculate some metrics such as R2 score, RSS and RMSE

y_pred_train = lasso2.predict(X_train)
y_pred_test = lasso2.predict(X_test)

metric2 = []
r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric2.append(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)
metric2.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
metric2.append(rss1_lr)

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(rss2_lr)
metric2.append(rss2_lr)

mse_train_lr = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric2.append(mse_train_lr**0.5)

mse_test_lr = mean_squared_error(y_test, y_pred_test)
print(mse_test_lr)
metric2.append(mse_test_lr**0.5)
```

```
0.9264445633351519
0.9139977986410903
260716476499.26947
153976283648.31073
343047995.39377564
606205841.1350816
```

In [72]:

```
lr_table = {'Metric': ['R2 Score (Train)','R2 Score (Test)','RSS (Train)','RSS (Test)',
                       'MSE (Train)','MSE (Test)'],
        }

lr_metric = pd.DataFrame(lr_table ,columns = ['Metric'] )

rg_metric = pd.Series(metric1, name = 'Ridge Regression')
ls_metric = pd.Series(metric2, name = 'Lasso Regression')

final_metrics_double = pd.concat([lr_metric, rg_metric, ls_metric], axis = 1)

final_metrics_double
```

Out[72]:

	Metric	Ridge Regression	Lasso Regression
0	R2 Score (Train)	9.323807e-01	9.264446e-01
1	R2 Score (Test)	9.108653e-01	9.139978e-01
2	RSS (Train)	2.396758e+11	2.607165e+11

	Metric	Ridge Regression	Lasso Regression
3	RSS (Test)	1.595847e+11	1.539763e+11
4	MSE (Train)	1.775846e+04	1.852155e+04
5	MSE (Test)	2.506564e+04	2.462125e+04

In [73]:
```
final_metrics
```

Out[73]:

	Metric	Linear Regression	Ridge Regression	Lasso Regression
0	R2 Score (Train)	7.678590e-01	9.393533e-01	9.354791e-01
1	R2 Score (Test)	7.644506e-01	9.206913e-01	9.246366e-01
2	RSS (Train)	8.228213e+11	2.149616e+11	2.286937e+11
3	RSS (Test)	4.217220e+11	1.419925e+11	1.349289e+11
4	MSE (Train)	3.290379e+04	1.681797e+04	1.734684e+04
5	MSE (Test)	4.074706e+04	2.364372e+04	2.304812e+04

- We can Observe that for both Models the R2 Score on Test and train decreased which is a sign of underfitting .
- We can also Observe that for both models the Rss and MSE value is Increased which is also a sign of underfitting .

In [74]:
```
#Co-efficients
betas = pd.DataFrame(index=X_train.columns)
betas.rows = X_train.columns
betas['Ridge2'] = ridge2.coef_
betas['Ridge'] = ridge.coef_
betas['Lasso'] = lasso.coef_
betas['Lasso2'] = lasso2.coef_
pd.set_option('display.max_rows', None)
betas.head()
```

Out[74]:

	Ridge2	Ridge	Lasso	Lasso2
LotFrontage	4608.853003	2167.708705	0.000000	0.000000
LotArea	17794.105988	20132.447889	20297.398604	18156.618705
OverallQual	31365.160494	36910.250129	56703.719301	60258.348561
OverallCond	15444.006039	20826.026772	26859.567311	17371.446959
YearBuilt	8197.301170	12238.876361	27486.614408	16181.928846

- We can Observe that the coefficents of the predictor's changed a lot.

In [75]:
```
betas['Lasso'].value_counts()[0] # count of Zero coeffecient variables with alpha 100
```

Out[75]:
```
142
```

In [76]:
```
betas['Lasso2'].value_counts()[0]  # count of Zero coeffecient variables with alpha 200
```

Out[76]: 163

- We can Clearly see that when we Increased the Alpha value, The no of variables which coeffecients are equal to zero Increased from 142 to 163.
- So model excluded 21 variables when alpha becomes doubled (which is useful for feature selection).

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

- I personally choose Lasso Regression Because the The R2 Score are slighly high and also RSS and MSE are also less compared to Ridge.
- When it comes to Lambda (alpha) values I choose the First alpha values 4,100 for Ridge and lasso Respectively , Beacuse when i choose the Doubled Alpha values the R2 Score on Test and train decreased and also Rss and MSE value is Increased.

Question 3

After building the model, you realised 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

1) Lets start with Ridge Model

In [77]:
```
ridge_imp = pd.DataFrame()
ridge_imp["Feature"] = pd.Series(X.columns)
ridge_imp["imp"] = pd.Series(ridge.coef_)
ridge_imp["abs_imp"] = [abs(t) for t in ridge.coef_]
ridge_imp = ridge_imp.sort_values(by = "abs_imp",ascending = False)
print(ridge_imp.head(5))
ridge_imp_fea = list(ridge_imp.head(5)["Feature"])
ridge_imp_fea
```

```
         Feature           imp       abs_imp
14     GrLivArea  46382.168251  46382.168251
11      1stFlrSF  41041.495734  41041.495734
2    OverallQual  36910.250129  36910.250129
7     BsmtFinSF1  34173.286652  34173.286652
10   TotalBsmtSF  30428.471570  30428.471570
```

Out[77]: ['GrLivArea', '1stFlrSF', 'OverallQual', 'BsmtFinSF1', 'TotalBsmtSF']

- The top 5 important Features in ridge model are 'GrLivArea', '1stFlrSF', 'OverallQual', 'BsmtFinSF1', 'TotalBsmtSF'.

In [78]:
```
# lets drop them
X_train1r = X_train.drop(columns=ridge_imp_fea)
X_test1r = X_test.drop(columns=ridge_imp_fea)
```

In [79]:
```
params = {'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1,
```

```
 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50, 100, 500, 1000,1500,2000,2500,5000 ]}

ridge = Ridge()

# cross validation
folds = 5
model_cv = GridSearchCV(estimator = ridge,
                        param_grid = params,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)
model_cv.fit(X_train1r, y_train)
```

```
Fitting 5 folds for each of 32 candidates, totalling 160 fits
```

Out[79]:

```
GridSearchCV(cv=5, estimator=Ridge(),
             param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3,
                                   0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
                                   4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50,
                                   100, 500, 1000, 1500, 2000, ...]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

In [80]:

```
# Printing the best hyperparameter alpha
print(model_cv.best_params_)
```

```
{'alpha': 3.0}
```

In [81]:

```
#Fitting Ridge model for alpha = 3 and printing coefficients which have been penalised
alpha = model_cv.best_params_['alpha']
ridge1 = Ridge(alpha=alpha)

ridge1.fit(X_train1r, y_train)
#print(ridge1.coef_)
```

Out[81]:

```
Ridge(alpha=3.0)
```

In [82]:

```
# Lets calculate some metrics such as R2 score, RSS and RMSE
y_pred_train = ridge1.predict(X_train1r)
y_pred_test = ridge1.predict(X_test1r)

metric_r1 = []
r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric_r1.append(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)
metric_r1.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
metric_r1.append(rss1_lr)

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(rss2_lr)
metric_r1.append(rss2_lr)

mse_train_lr = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric_r1.append(mse_train_lr**0.5)
```

```
mse_test_lr = mean_squared_error(y_test, y_pred_test)
print(mse_test_lr)
metric_r1.append(mse_test_lr**0.5)
```

```
0.9205224293210008
0.8806680763698564
281707418617.6762
213649020965.08936
370667656.07588977
841137877.8153124
```

In [83]:

```
lr_table = {'Metric': ['R2 Score (Train)','R2 Score (Test)','RSS (Train)','RSS (Test)',
                       'MSE (Train)','MSE (Test)'],
        }

lr_metric = pd.DataFrame(lr_table ,columns = ['Metric'] )

rg_metric = pd.Series(metric1, name = 'Ridge Regression')

rg_metric1 = pd.Series(metric_r1, name = 'Ridge Regression_New')


metrics_r = pd.concat([lr_metric, rg_metric,rg_metric1], axis = 1)

metrics_r
```

Out[83]:

	Metric	Ridge Regression	Ridge Regression_New
0	R2 Score (Train)	9.323807e-01	9.205224e-01
1	R2 Score (Test)	9.108653e-01	8.806681e-01
2	RSS (Train)	2.396758e+11	2.817074e+11
3	RSS (Test)	1.595847e+11	2.136490e+11
4	MSE (Train)	1.775846e+04	1.925273e+04
5	MSE (Test)	2.506564e+04	2.900238e+04

We can Clearly Observe that the R2 score of the test data set decreased compared to old model.

In [84]:

```
ridge_imp1 = pd.DataFrame()
ridge_imp1["Feature"] = pd.Series(X_train1r.columns)
ridge_imp1["imp"] = pd.Series(ridge1.coef_)
ridge_imp1["abs_imp"] = [abs(t) for t in ridge1.coef_]
ridge_imp1 = ridge_imp1.sort_values(by = "abs_imp",ascending = False)
print(ridge_imp.head(5))
ridge_imp_fea1 = list(ridge_imp1.head(5)["Feature"])
ridge_imp_fea1
```

```
        Feature           imp       abs_imp
14    GrLivArea  46382.168251  46382.168251
11     1stFlrSF  41041.495734  41041.495734
2   OverallQual  36910.250129  36910.250129
7    BsmtFinSF1  34173.286652  34173.286652
10  TotalBsmtSF  30428.471570  30428.471570
```

Out[84]:

```
['TotRmsAbvGrd', 'Neighborhood_StoneBr', '2ndFlrSF', 'FullBath', 'GarageArea']
```

The new Top 5 features are 'TotRmsAbvGrd', 'Neighborhood_StoneBr', '2ndFlrSF', 'FullBath', 'GarageArea'.

2) For Lasso Model.

In [85]:
```
lasso_imp = pd.DataFrame()
lasso_imp["Feature"] = pd.Series(X.columns)
lasso_imp["imp"] = pd.Series(lasso.coef_)
lasso_imp["abs_imp"] = [abs(t) for t in lasso.coef_]
lasso_imp = lasso_imp.sort_values(by = "abs_imp",ascending = False)
print(lasso_imp.head(5))
lasso_imp_fea = list(lasso_imp.head(5)["Feature"])
print(lasso_imp_fea)
```

```
                 Feature            imp        abs_imp
14             GrLivArea  145195.619477  145195.619477
2            OverallQual   56703.719301   56703.719301
7             BsmtFinSF1   34361.687464   34361.687464
10           TotalBsmtSF   32520.386468   32520.386468
87  Neighborhood_StoneBr   31971.621494   31971.621494
['GrLivArea', 'OverallQual', 'BsmtFinSF1', 'TotalBsmtSF', 'Neighborhood_StoneBr']
```

- The top 5 important Features in Lasso model are 'GrLivArea', 'OverallQual', 'BsmtFinSF1', 'TotalBsmtSF', 'Neighborhood_StoneBr'.

In [86]:
```
# lets drop them
X_train11 = X_train.drop(columns=lasso_imp_fea)
X_test11= X_test.drop(columns=lasso_imp_fea)
```

In [87]:
```
lasso = Lasso()

# cross validation
model_cv = GridSearchCV(estimator = lasso,
                        param_grid = params,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)

model_cv.fit(X_train11, y_train)
```

Out[87]:
```
Fitting 5 folds for each of 32 candidates, totalling 160 fits
GridSearchCV(cv=5, estimator=Lasso(),
             param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3,
                                   0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
                                   4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50,
                                   100, 500, 1000, 1500, 2000, ...]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

In [88]:
```
# Printing the best hyperparameter alpha
print(model_cv.best_params_)
```

```
{'alpha': 100}
```

In [89]:
```
#Fitting Ridge model for alpha = 100 and printing coefficients which have been penalised

alpha =model_cv.best_params_['alpha']

lasso1 = Lasso(alpha=alpha)

lasso1.fit(X_train11, y_train)
#print(lasso1.coef_)
```

Out[89]:
```
Lasso(alpha=100)
```

In [90]:

```
# Lets calculate some metrics such as R2 score, RSS and RMSE
y_pred_train = lasso1.predict(X_train1l)
y_pred_test = lasso1.predict(X_test1l)

metric_l1 = []
r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric_l1.append(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)
metric_l1.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
metric_l1.append(rss1_lr)

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(rss2_lr)
metric_l1.append(rss2_lr)

mse_train_lr = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric_l1.append(mse_train_lr**0.5)

mse_test_lr = mean_squared_error(y_test, y_pred_test)
print(mse_test_lr)
metric_l1.append(mse_test_lr**0.5)
```

```
0.9257559656924212
0.9085812155430923
263157203647.1002
163674004431.30988
346259478.48302656
644385844.2177554
```

In [91]:

```
lr_table = {'Metric': ['R2 Score (Train)','R2 Score (Test)','RSS (Train)','RSS (Test)',
                       'MSE (Train)','MSE (Test)'],
        }

lr_metric = pd.DataFrame(lr_table ,columns = ['Metric'] )

ls_metric = pd.Series(metric2, name = 'Lasso Regression')
ls_metric1 = pd.Series(metric_l1, name = 'Lasso Regression_New')

final_metrics_all = pd.concat([lr_metric,ls_metric, ls_metric1], axis = 1)

final_metrics_all
```

Out[91]:

	Metric	Lasso Regression	Lasso Regression_New
0	R2 Score (Train)	9.264446e-01	9.257560e-01
1	R2 Score (Test)	9.139978e-01	9.085812e-01
2	RSS (Train)	2.607165e+11	2.631572e+11
3	RSS (Test)	1.539763e+11	1.636740e+11
4	MSE (Train)	1.852155e+04	1.860805e+04
5	MSE (Test)	2.462125e+04	2.538476e+04

We can Clearly Observe that the R2 score of the test data set is Slightly decreased compared to old model.

In [92]:

```
lasso_imp1 = pd.DataFrame()
lasso_imp1["Feature"] = pd.Series(X_train1l.columns)
lasso_imp1["imp"] = pd.Series(lasso1.coef_)
lasso_imp1["abs_imp"] = [abs(t) for t in lasso1.coef_]
lasso_imp1 = lasso_imp1.sort_values(by = "abs_imp",ascending = False)
print(lasso_imp1.head(5))
lasso_imp_fea1 = list(lasso_imp1.head(5)["Feature"])
lasso_imp_fea1
```

```
             Feature            imp        abs_imp
8           1stFlrSF  177754.733646  177754.733646
9           2ndFlrSF   81652.315402   81652.315402
3          YearBuilt   31721.495953   31721.495953
2        OverallCond   27507.910087   27507.910087
189   KitchenQual_TA  -26069.597794   26069.597794
```

Out[92]:

```
['1stFlrSF', '2ndFlrSF', 'YearBuilt', 'OverallCond', 'KitchenQual_TA']
```

The new Top 5 features are '1stFlrSF', '2ndFlrSF', 'YearBuilt', 'OverallCond', 'KitchenQual_TA'.

In [ ]:

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

1. The model should be generalized so that the accuracy of the test Should not be much less than the training points so that the model will be giving similar accuracy For Other data sets too.
2. We should also take care that our model should not be affected by Outliers, And should not given more importance to the column containing outliers which leads to overfit , So we should delete Outliers before training the model.
3. The Model Should have high R^2 Score which makes sure that our model covered the most of the variance from the data. so that even any variation in the data should not effect the model much.
4. To make sure the model is robust and generalizable, we must be careful not to overfit it. This is because the overfit model has very high variability and a small change in the data greatly affects the model prediction. Such a model would identify all training data patterns, but fail to select patterns in unseen test data.
5. In other words, the model should not be too complex in order to be robust and generalizable and also Should not be too simple to underfit.
6. If we look at it from the prespective of Accuracy, the most complex model will have the highest accuracy. Therefore, in order to make our model more robust and generalizable, we will need to reduce the variance that will lead to certain biases. Increased bias means that accuracy will decrease by a little bit.So we need to sacrifice some variance and accuracy for the bias.
7. In general, we should find a balance between model accuracy and complexity. This can be achieved with strategies such as Ridge Regression and Lasso.

In [ ]: