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 –

It is important to regularize coefficients and improve the prediction accuracy with the decrease in variance, and making the model interpretably.

Ridge Alpha = 1 and Lasso Alpha = 10

When we double the value of alpha for our ridge regression number we will take the value of alpha equal to 3 the model will apply more penalty on the curve and try to make the model more generalized that is making model more simpler and not thinking to fit every data of the data set. We can see below that when alpha is 3 we get more error for both test and train. Similarly when we increase the value of alpha for lasso we try to penalize more our model and more coefficient of the variable will reduced to zero, when we increase the value of our r2 square also decreases.

Ridge Regression

```
#Taking the alpha as 3 for second part of the question
alpha=3
ridge2=Ridge(alpha=alpha)
ridge2.fit(X_train1,y_train)
```

```
Ridge(alpha=3)
```

```
[102] #Calculating R2 score,RSS,RMSE metrics
y_pred_train=ridge2.predict(X_train1)
y_pred_test=ridge2.predict(X_test1)

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_train_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)
```

0s completed at 7:53 PM

```
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)
```

```
0.8594616894298742
0.8552471804889255
710465897888.1814
360197323411.0558
795594510.5130811
818630280.4796722
```

1(ii) From above found metrics, we can that the R2 score on test data is much higher compared to training data. This is due to the value of alpha is doubled value in Ridge Regression.

Lasso

```
# Answer 1.b -Increasing alpha from 10 to 20
alpha=20
lasso20=Lasso(alpha=alpha)
lasso20.fit(X_train1,y_train)
```

```
Lasso(alpha=20)
```

[104]
```
#Calculating R2 score,RSS,RMSE metrics
y_pred_train=lasso20.predict(X_train1)
y_pred_test=lasso20.predict(X_test1)

metric3=[]
r2_train_lr=r2_score(y_train,y_pred_train)
print(r2_train_lr)
metric.append(r2_train_lr)

r2_test_lr=r2_score(y_test,y_pred_test)
print(r2_test_lr)
metric3.append(r2_train_lr)

rss1_lr=np.sum(np.square(y_train-y_pred_train))
print(rss1_lr)
metric3.append(rss1_lr)

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

0s completed at 7:53 PM

```
rss2_lr=np.sum(np.square(y_test-y_pred_test))
print(rss2_lr)
metric3.append(rss2_lr)

mse_train_lr=mean_squared_error(y_train,y_pred_train)
print(mse_train_lr)
metric3.append(mse_train_lr**0.5)

mse_test_lr=mean_squared_error(y_test,y_pred_test)
print(mse_test_lr)
metric3.append(mse_test_lr**0.5)
```

```
0.8647902710478352
0.8546588624002092
683528221546.9956
361661271412.7988
765429139.4703199
821957435.0290883
```

1(ii) From above found metrics, we can observe that R2 score on test data compared to the training data. This is due to the value of alpha getting doubled in Lasso Regression.

```
# Answer 1(iii) Finding out signifiant predictor variables
sigvar=pd.DataFrame(index=X_train1.columns)
sigvar.rows=X_train1.columns
sigvar['Ridge2']=ridge2.coef_
sigvar['Ridge']=ridge.coef_
sigvar['Lasso']=lasso.coef_
sigvar['Lasso20']=lasso20.coef_
pd.set_option('display.max_rows',None)
sigvar.head(68)
```

	Ridge2	Ridge	Lasso	Lasso20
LotArea	55101.870644	58176.069939	59790.799418	59178.521502
OverallQual	148494.769836	162226.405573	169373.620189	169859.381366
BsmtFinSF1	60848.007626	60675.788024	60511.914589	60443.571118
TotalBsmtSF	83702.959681	85405.067242	85098.731530	85040.335815
GrLivArea	153093.360785	160231.634056	163927.477472	163637.894083
Street_Pave	42149.633439	59203.836037	72582.614945	69442.165722
Exterior1st_CBlock	-19897.038279	-39602.271430	-71016.941911	-61087.647450
Exterior1st_Stone	-13195.430726	-28659.417092	-50648.721912	-41527.094945
ExterQual_Gd	-47900.978736	-53954.511672	-57846.026163	-56890.047520
ExterQual_TA	-71063.322157	-73316.686870	-75379.757020	-74376.889133

0s completed at 7:53 PM

Answer 1(iii)

- LotArea---------------Lot size in square feet
- TotalBsmtSF------- Total square feet of basement area
- GrLivArea-----------Above grade (ground) living area square feet
- RoofMatl_Metal----Roof material_Metal
- OverallQual---------Rates the overall material and finish of the house
- OverallCond--------Rates the overall condition of the house
- TotRmsAbvGrd----Total rooms above grade (does not include bathrooms)
- Street_Pave--------Pave road access to property
- YearBuilt-------------Original construction date
- BsmtFinSF1--------Type 1 finished square feet

From above results, we observed that predictors are same but the coefficient of these predictor has modified.

From above results, we observed that predictors are same but the coefficient of this predictor has modified. It is important to regularize coefficients and improve the prediction accuracy with the decrease in variance, and making the model interpretably. Ridge regression, uses a tuning parameter called lambda as the penalty is square of magnitude of coefficients, which is identified by cross validation. Residual sum or squares should be small by using the penalty. The penalty is lambda multiple of sum of squares of the coefficients, hence the coefficients that have greater values are penalized. As we increase the value of lambda the variance in model is dropped and bias remains constant. Ridge regression includes all variables in final model unlike Lasso Regression. Lasso regression, uses a tuning parameter called lambda, as the penalty is absolute value of magnitude of coefficients, which is identified by cross validation. As the lambda value increases Lasso shrinks the coefficient towards zero and it make the variables exactly equal to 0. Lasso also does variable selection. When lambda value is small it performs simple linear regression and as lambda value increases, shrinkage takes place and variables with 0 value are neglected by the model.

The most important variable after the changes has been implemented for ridge regression are as follows:-

- MSZoning_FV
- MSZoning_RL
- Neighborhood_Crawfor
- MSZoning_RH
- MSZoning_RM
- SaleCondition_Partial
- Neighborhood_StoneBr
- GrLivArea
- SaleCondition_Normal
- Exterior1st_BrkFace

The most important variable after the changes has been implemented for lasso regression are as follows:-

- GrLivArea
- OverallQual
- OverallCond

- TotalBsmtSF
- BsmtFinSF1
- GarageArea
- Fireplaces
- LotArea
- LotArea
- LotFrontage

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 -

The R2 scores are being checked for all the models. According to the below data the lasso's r2_score is a bit greater compared to the test dataset's lasso. Hence, lasso regression is being chosen to solve this problem. The Lasso will help in feature elimination and will be more robust, hence lasso is used for the model. It is important to regularize coefficients and improve the prediction accuracy with the decrease in variance, and making the model interpretably.

	Ridge Regression	Lasso Regression
R2 score(Train)	0.88	0.88
R2 score(Test)	0.87	0.86

```
[106] final_metric
```

	Metric	Linear Regression	Ridge Regression	Lasso Regression
0	R2 Score (Train)	8.649649e-01	8.638146e-01	8.649213e-01
1	R2 core (Test)	8.528033e-01	8.559949e-01	8.537763e-01
2	RSS (Train)	6.826452e+11	6.884606e+11	6.828660e+11
3	RSS (Test)	3.662785e+11	3.583368e+11	3.638574e+11
4	MSE (Train)	2.764851e+04	2.776603e+04	2.765298e+04
5	MSE (Test)	2.885223e+04	2.853773e+04	2.875671e+04

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 –

Those 5 most important predictor variables that will be excluded are:-

- RoofMatl_Metal------Roof material_Metal
- 11stFlrSF-----------First Floor square feet
- GrLivArea-----------Above grade (ground) living area square feet
- RoofStyle_Shed------Type of roof(Shed)
- Street_Pave---------Pave road access to property

```
[107] #Printing X_train1
X_train1
```

	LotArea	OverallQual	BsmtFinSF1	TotalBsmtSF	GrLivArea	Street_Pave	Exterior1st_CBlock	Exterior1st_Stone	ExterQual_Gd	ExterQual_TA
1108	0.187723	0.555556	0.000000	0.288210	0.407819	1	0	0	0	1
745	0.213431	0.777778	0.262797	0.356207	0.753286	1	0	0	0	0
1134	0.208004	0.555556	0.000000	0.285714	0.377486	1	0	0	0	1
512	0.217344	0.444444	0.238117	0.269495	0.129424	1	0	0	0	1
43	0.220201	0.444444	0.127971	0.292576	0.154365	1	0	0	0	1
33	0.258819	0.444444	0.465265	0.436057	0.411190	1	0	0	0	1
269	0.183553	0.555556	0.343236	0.356519	0.213347	1	0	0	0	1
789	0.306036	0.555556	0.259598	0.259513	0.541625	1	0	0	0	1
1038	0.001200	0.333333	0.000000	0.170306	0.291203	1	0	0	0	1
151	0.354195	0.777778	0.639854	0.533375	0.414560	1	0	0	1	0
344	0.031449	0.444444	0.058958	0.167187	0.213010	1	0	0	0	1
1218	0.135651	0.333333	0.000000	0.000000	0.145602	1	0	0	0	1
1040	0.332315	0.444444	0.076782	0.353712	0.445905	1	0	0	0	1
688	0.188466	0.777778	0.431901	0.442608	0.316481	1	0	0	1	0
1289	0.273473	0.777778	0.000000	0.338428	0.502191	1	0	0	1	0
1459	0.241252	0.444444	0.379342	0.391765	0.261544	1	0	0	1	0
1448	0.293525	0.333333	0.000000	0.174672	0.291877	1	0	0	0	1

✓ 0s completed at 7:53 PM

```
#Printing y_train
y_train
```

```
1108    181000
745     299800
1134    169000
512     129900
43      130250
33      165500
269     148000
789     187500
1038     97000
151     372402
344      85000
1218     80500
1040    155000
688     392000
1289    281000
1459    147500
1448    112000
733     131400
3       140000
123     153900
812      55993
1258    190000
929     222000
1348    215000
692     335000
1014    119200
412     222000
1425    142000
497     184000
603     151000
```

✓ 0s completed at 7:53 PM

```
# Printing X_train1 Columns
X_train1.columns
```

```
Index(['LotArea', 'OverallQual', 'BsmtFinSF1', 'TotalBsmtSF', 'GrLivArea', 'Street_Pave', 'Exterior1st
```

OverallQual,YearBuilt,LotArea,TotalBsmtSF,BsmtFinSF1 are the top 5 important predictor variables.

We are dropping these columns.

```
[110] X_train2=X_train1.drop(['LotArea','OverallQual','BsmtFinSF1','TotalBsmtSF'],axis=1)
      X_test2 = X_test1.drop(['LotArea','OverallQual','BsmtFinSF1','TotalBsmtSF'],axis=1)
```

```
[111] X_train2.head()
```

	GrLivArea	Street_Pave	Exterior1st_CBlock	Exterior1st_Stone	ExterQual_Gd	ExterQual_TA
1108	0.407819	1	0	0	0	1
745	0.753286	1	0	0	0	0
1134	0.377486	1	0	0	0	1
512	0.129424	1	0	0	0	1
43	0.154365	1	0	0	0	1

```
[112] X_test2.head()
```

	GrLivArea	Street_Pave	Exterior1st_CBlock	Exterior1st_Stone	ExterQual_Gd	ExterQual_TA
990	0.644422	1	0	0	1	0

✓ 0s completed at 7:53 PM

```
[112] X_test2.head()
```

	GrLivArea	Street_Pave	Exterior1st_CBlock	Exterior1st_Stone	ExterQual_Gd	ExterQual_TA
990	0.644422	1	0	0	1	0
1161	0.390967	1	0	0	1	0
1369	0.400404	1	0	0	1	0
329	0.239973	1	0	0	0	1
262	0.246714	1	0	0	0	1

Performing Lasso

```
[113] #Taking alpha as 10
      alpha=10
      lasso21=Lasso(alpha=alpha)
      lasso21.fit(X_train2,y_train)
```

```
Lasso(alpha=10)
```

```
[114] #Calulating R2 Score,RSS,RMSE Metrics
      y_pred_train = lasso21.predict(X_train2)
      y_pred_test = lasso21.predict(X_test2)

      metric3 = []
      r2_train_lr = r2_score(y_train, y_pred_train)
      print(r2_train_lr)
      metric3.append(r2_train_lr)
```

0s completed at 7:53 PM

The training and testing dataset's R2 Score has reduced

```
#important predictor variables
sigvar = pd.DataFrame(index=X_train2.columns)
sigvar.rows = X_train1.columns
sigvar['Lasso21'] = lasso21.coef_
pd.set_option('display.max_rows', None)
sigvar.head(68)
```

	Lasso21
GrLivArea	266134.574496
Street_Pave	119379.002208
Exterior1st_CBlock	-177866.690818
Exterior1st_Stone	-37283.103678
ExterQual_Gd	-95284.819821
ExterQual_TA	-142101.931900

0s completed at 7:53 PM

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 –

The model should be as simple and generalized so that the training score is greater than the test accuracy. The model should be accurate for datasets other than the ones, which were used during training. Do not give importance to the outliers so that the accuracy predicted by the model is high. That is why, the outliers analysis needs to be done and we need to retain only those values which are relevant to the dataset. The outliers, which are not significant, should be eliminated from the dataset. In the event that the model isn't strong, it can't be relied upon for prescient examination. The model ought to be as straightforward as could really be expected, however its exactness will diminish yet it will be more strong and generalisable. It tends to be likewise perceived in the conditions of the Bias-Variance compromise. The less difficult the model the more the predisposition yet not so much fluctuation but rather more generalizable it is. Its suggestion as far as precision is that a strong and generalisable model will perform similarly well on both preparation and test information for example the exactness doesn't change much for preparing and test information.

Bias: Bias is error in model, when the model is weak to learn from the data. High bias means model is unable to learn details in the data. Model performs poor on training and testing data.

Variance: When model tries to over learn from the data, Variance is error in model. High variance means model performs exceptionally well on training data as it has very well trained on this of data but performs very poor on testing data as it was unseen data for the model. For avoiding the overfitting and underfitting of data, balancing in Bias and variance.