SUBJECTIVE QUESTIONS

Question 1

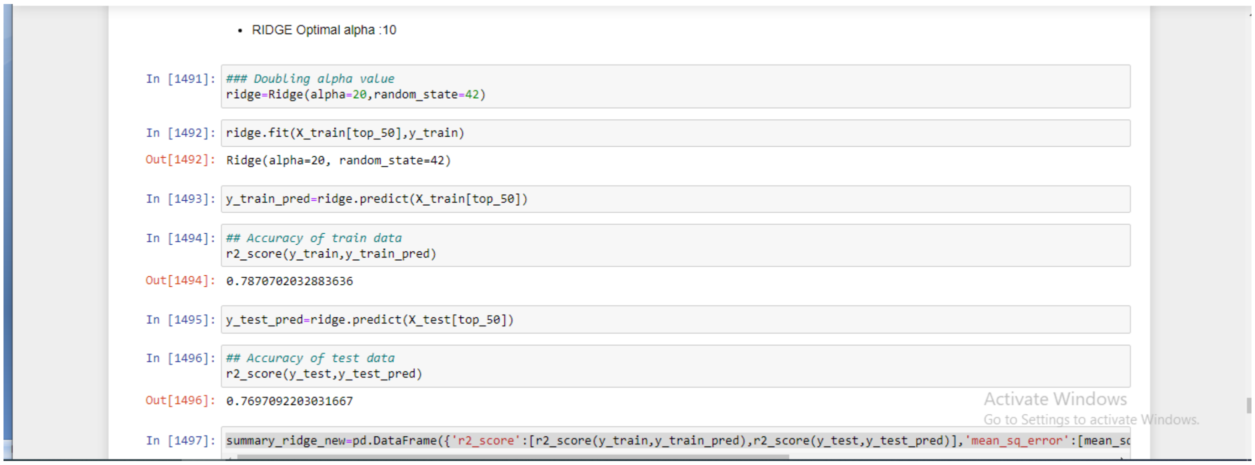
What is the optimal value of alpha for ridge and lasso regression? What will be the changes in the model if you choose double the value of alpha for both ridge and lasso? What will be the most important predictor variables after the change is implemented?

Answer:

After performing hyperparameter tuning on the various alpha values using GridSearchCV,

- The optimal value for ridge regression turns out to be alpha=10
- The optimal value for lasso regression turns out to be alpha=0.001

When doubling the value of alpha=20

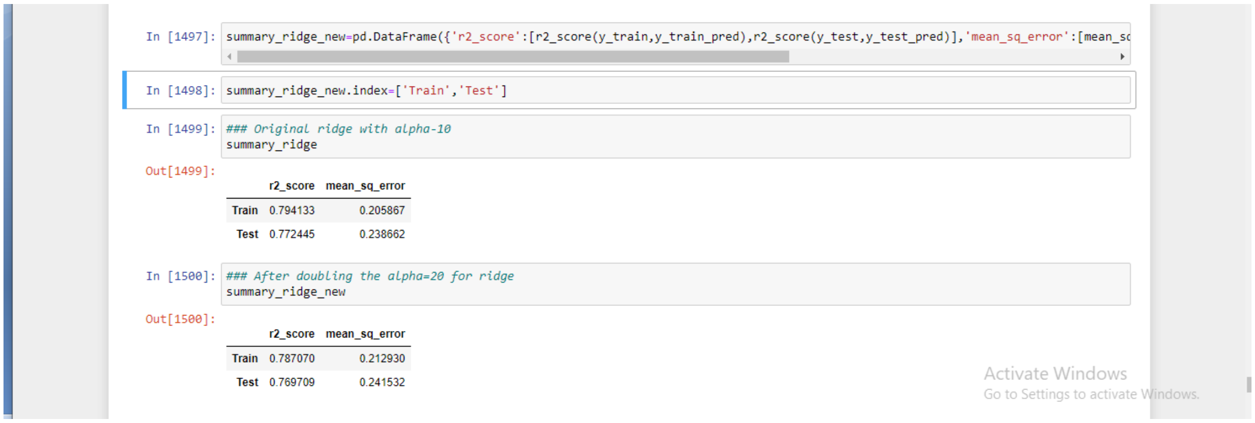


- RIDGE Optimal alpha :10

```
In [1491]: ### Doubling alpha value
           ridge=Ridge(alpha=20,random_state=42)

In [1492]: ridge.fit(X_train[top_50],y_train)

Out[1492]: Ridge(alpha=20, random_state=42)

In [1493]: y_train_pred=ridge.predict(X_train[top_50])

In [1494]: ## Accuracy of train data
           r2_score(y_train,y_train_pred)

Out[1494]: 0.7870702032883636

In [1495]: y_test_pred=ridge.predict(X_test[top_50])

In [1496]: ## Accuracy of test data
           r2_score(y_test,y_test_pred)

Out[1496]: 0.7697092203031667

In [1497]: summary_ridge_new=pd.DataFrame({'r2_score':[r2_score(y_train,y_train_pred),r2_score(y_test,y_test_pred)],'mean_sq_error':[mean_sc
```

There lies a slight variation in the R2 score of train and test dataset, which is negligible. The overall model finds no much variation on doubling the value of alpha for ridge(10).

R2 score difference for ridge before and after doubling:



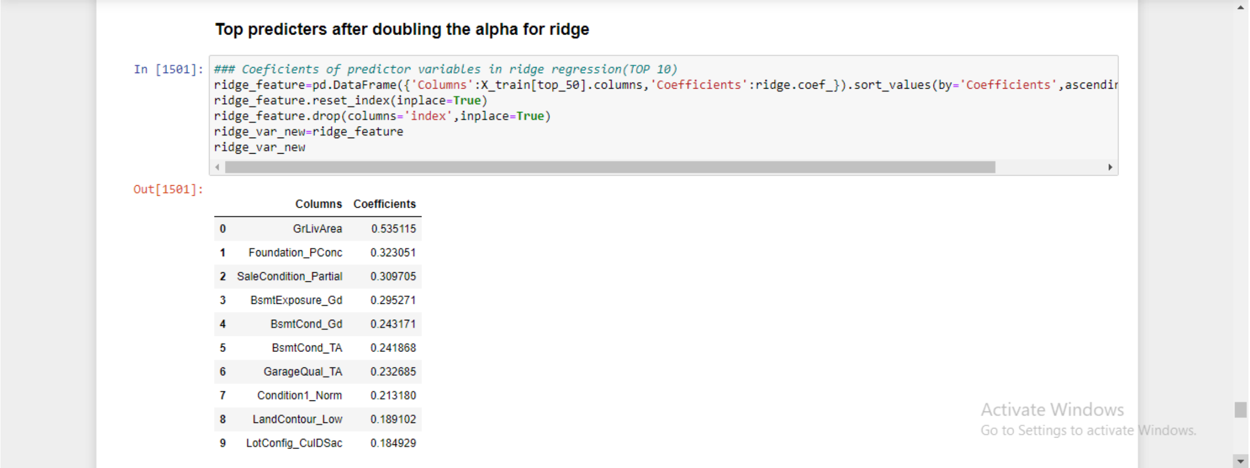
```
In [1497]: summary_ridge_new=pd.DataFrame({'r2_score':[r2_score(y_train,y_train_pred),r2_score(y_test,y_test_pred)],'mean_sq_error':[mean_sq

In [1498]: summary_ridge_new.index=['Train','Test']

In [1499]: ### Original ridge with alpha-10
           summary_ridge

Out[1499]:
```

	r2_score	mean_sq_error
Train	0.794133	0.205867
Test	0.772445	0.238662

```
In [1500]: ### After doubling the alpha=20 for ridge
           summary_ridge_new

Out[1500]:
```

	r2_score	mean_sq_error
Train	0.787070	0.212930
Test	0.769709	0.241532

Top 10 predictors of ridge after doubling:

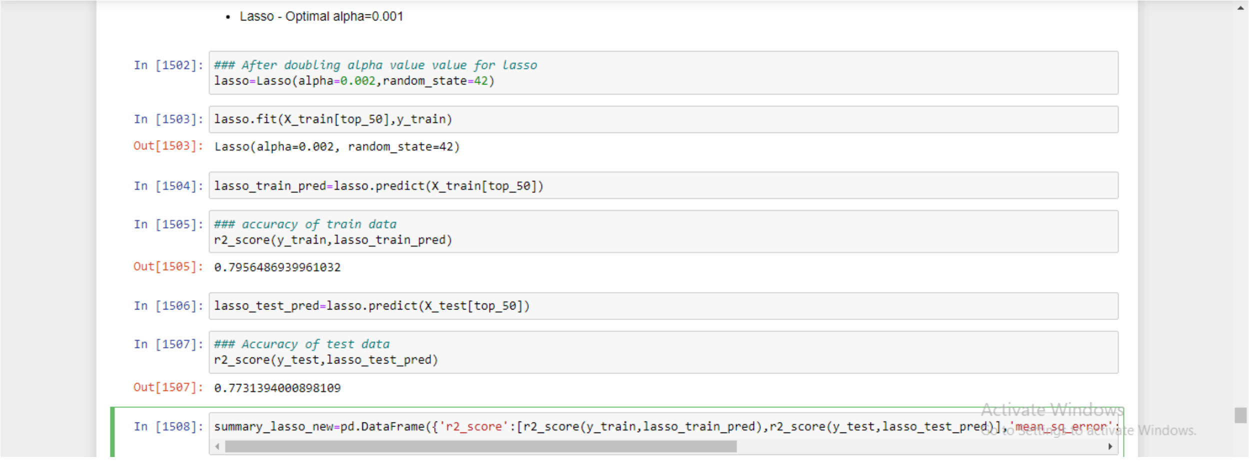


Top predicters after doubling the alpha for ridge

```
In [1501]: ### Coeficients of predictor variables in ridge regression(TOP 10)
           ridge_feature=pd.DataFrame({'Columns':X_train[top_50].columns,'Coefficients':ridge.coef_}).sort_values(by='Coefficients',ascendir
           ridge_feature.reset_index(inplace=True)
           ridge_feature.drop(columns='index',inplace=True)
           ridge_var_new=ridge_feature
           ridge_var_new

Out[1501]:
```

	Columns	Coefficients
0	GrLivArea	0.535115
1	Foundation_PConc	0.323051
2	SaleCondition_Partial	0.309705
3	BsmtExposure_Gd	0.295271
4	BsmtCond_Gd	0.243171
5	BsmtCond_TA	0.241868
6	GarageQual_TA	0.232685
7	Condition1_Norm	0.213180
8	LandContour_Low	0.189102
9	LotConfig_CulDSac	0.184929

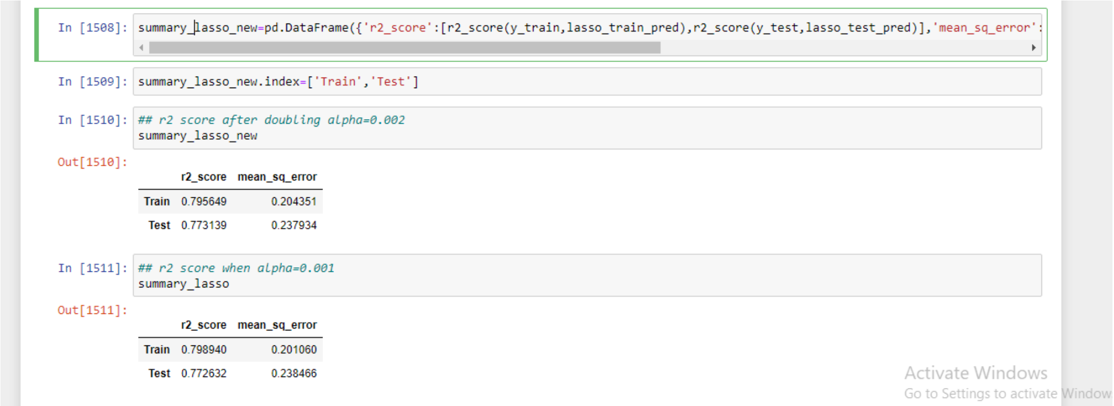
Doubling alpha =0.002 for Lasso:



- Lasso - Optimal alpha=0.001

```
In [1502]: ### After doubling alpha value value for lasso
           lasso=Lasso(alpha=0.002,random_state=42)

In [1503]: lasso.fit(X_train[top_50],y_train)

Out[1503]: Lasso(alpha=0.002, random_state=42)

In [1504]: lasso_train_pred=lasso.predict(X_train[top_50])

In [1505]: ### accuracy of train data
           r2_score(y_train,lasso_train_pred)

Out[1505]: 0.7956486939961032

In [1506]: lasso_test_pred=lasso.predict(X_test[top_50])

In [1507]: ### Accuracy of test data
           r2_score(y_test,lasso_test_pred)

Out[1507]: 0.7731394000898109

In [1508]: summary_lasso_new=pd.DataFrame({'r2_score':[r2_score(y_train,lasso_train_pred),r2_score(y_test,lasso_test_pred)],'mean_sq_error':
```

R2 score before and after doubling alpha for lasso:



```
In [1508]: summary_lasso_new=pd.DataFrame({'r2_score':[r2_score(y_train,lasso_train_pred),r2_score(y_test,lasso_test_pred)],'mean_sq_error':
```

```
In [1509]: summary_lasso_new.index=['Train','Test']
```

```
In [1510]: ## r2 score after doubling alpha=0.002
           summary_lasso_new
```

Out[1510]:

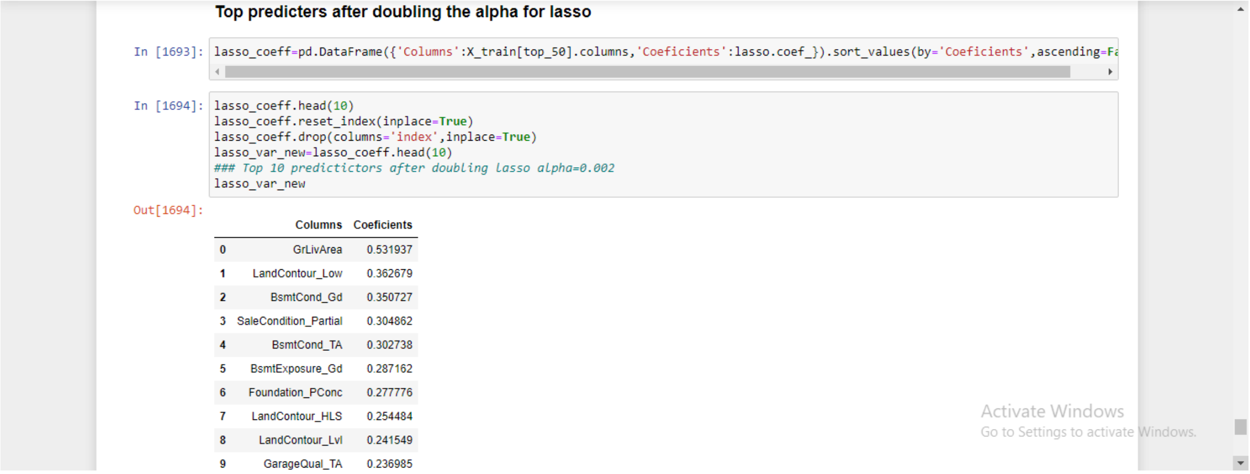
	r2_score	mean_sq_error
Train	0.795649	0.204351
Test	0.773139	0.237934

```
In [1511]: ## r2 score when alpha=0.001
           summary_lasso
```

Out[1511]:

	r2_score	mean_sq_error
Train	0.798940	0.201060
Test	0.772632	0.238466

Top 10 predictors after doubling alpha(0.002) for Lasso



Top predicters after doubling the alpha for lasso

```
In [1693]: lasso_coeff=pd.DataFrame({'Columns':X_train[top_50].columns,'Coeficients':lasso.coef_}).sort_values(by='Coeficients',ascending=Fa
```

```
In [1694]: lasso_coeff.head(10)
           lasso_coeff.reset_index(inplace=True)
           lasso_coeff.drop(columns='index',inplace=True)
           lasso_var_new=lasso_coeff.head(10)
           ### Top 10 predictictors after doubling lasso alpha=0.002
           lasso_var_new
```

Out[1694]:

	Columns	Coeficients
0	GrLivArea	0.531937
1	LandContour_Low	0.362679
2	BsmtCond_Gd	0.350727
3	SaleCondition_Partial	0.304862
4	BsmtCond_TA	0.302738
5	BsmtExposure_Gd	0.287162
6	Foundation_PConc	0.277776
7	LandContour_HLS	0.254484
8	LandContour_Lvl	0.241549
9	GarageQual_TA	0.236985

Overall, the model does not show significant difference when doubling the alpha value for both ridge and lasso and for the predictor variables.

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:

Both ridge and lasso regularization methods give similar r2_score and mean squared error values.

Ridge r2_score: Train(0.794133),Test(0.772445)

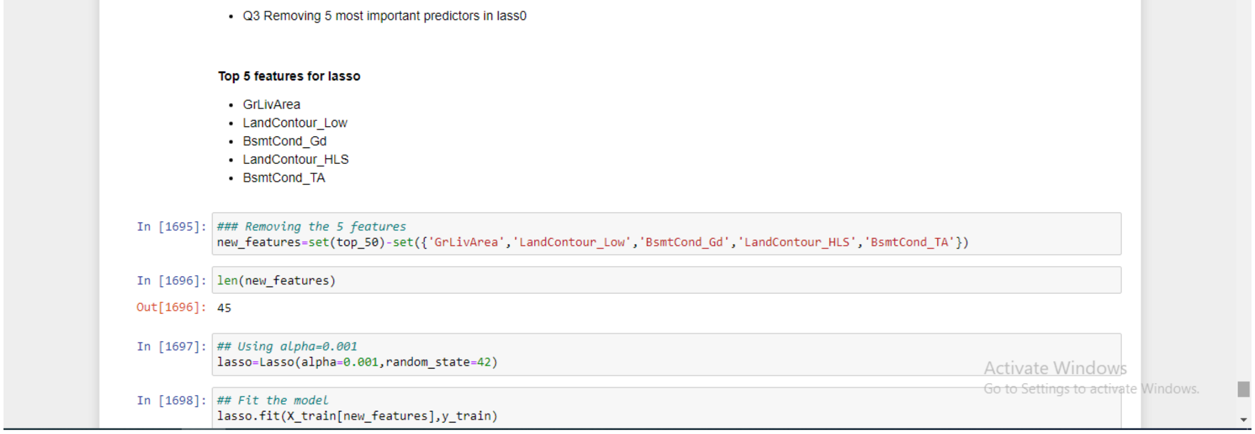
Lasso r2_score: Train(0.795649),Test(0.773139)

Considering the ‘feature elimination’ advantage of lasso regularization bringing the coefficients to zero, we tend to obtain the most important predictor variables for predicting the sales price through lasso regularization. So, it is preferred to proceed ahead with lasso as it meets the business goal of the dataset provided.

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?

The important 5 predictors of lasso regularization is listed below:



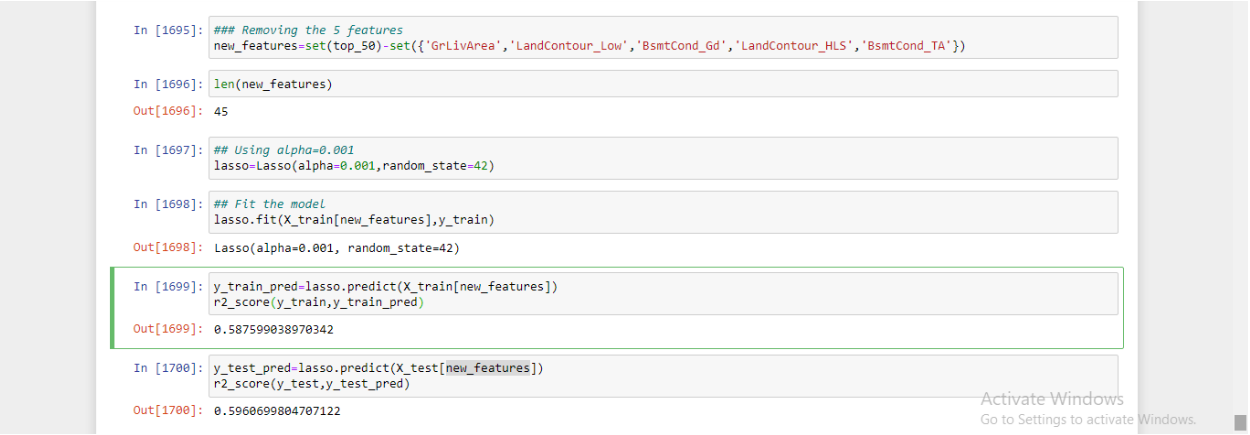
- Q3 Removing 5 most important predictors in lass0

Top 5 features for lasso

- GrLivArea
- LandContour_Low
- BsmtCond_Gd
- LandContour_HLS
- BsmtCond_TA

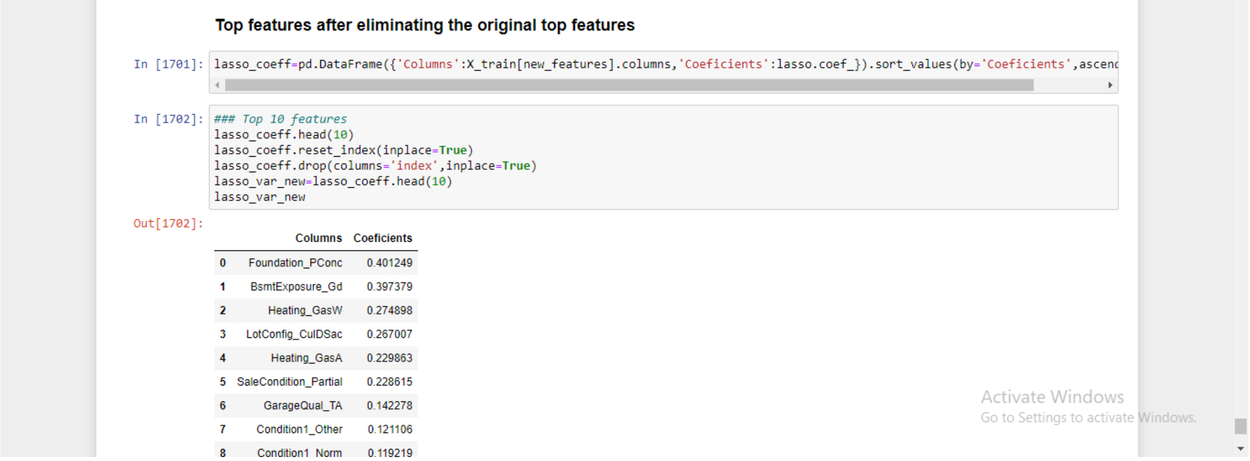
```
In [1695]: ### Removing the 5 features
           new_features=set(top_50)-set({'GrLivArea','LandContour_Low','BsmtCond_Gd','LandContour_HLS','BsmtCond_TA'})

In [1696]: len(new_features)

Out[1696]: 45

In [1697]: ## Using alpha=0.001
           lasso=Lasso(alpha=0.001,random_state=42)

In [1698]: ## Fit the model
           lasso.fit(X_train[new_features],y_train)
```

Removing those features from the input data and refitting the model to obtain the new significant predictor variables



```
In [1695]: ### Removing the 5 features
           new_features=set(top_50)-set({'GrLivArea','LandContour_Low','BsmtCond_Gd','LandContour_HLS','BsmtCond_TA'})

In [1696]: len(new_features)

Out[1696]: 45

In [1697]: ## Using alpha=0.001
           lasso=Lasso(alpha=0.001,random_state=42)

In [1698]: ## Fit the model
           lasso.fit(X_train[new_features],y_train)

Out[1698]: Lasso(alpha=0.001, random_state=42)

In [1699]: y_train_pred=lasso.predict(X_train[new_features])
           r2_score(y_train,y_train_pred)

Out[1699]: 0.587599038970342

In [1700]: y_test_pred=lasso.predict(X_test[new_features])
           r2_score(y_test,y_test_pred)

Out[1700]: 0.5960699804707122
```

New features obtained after removing the top 5 features:

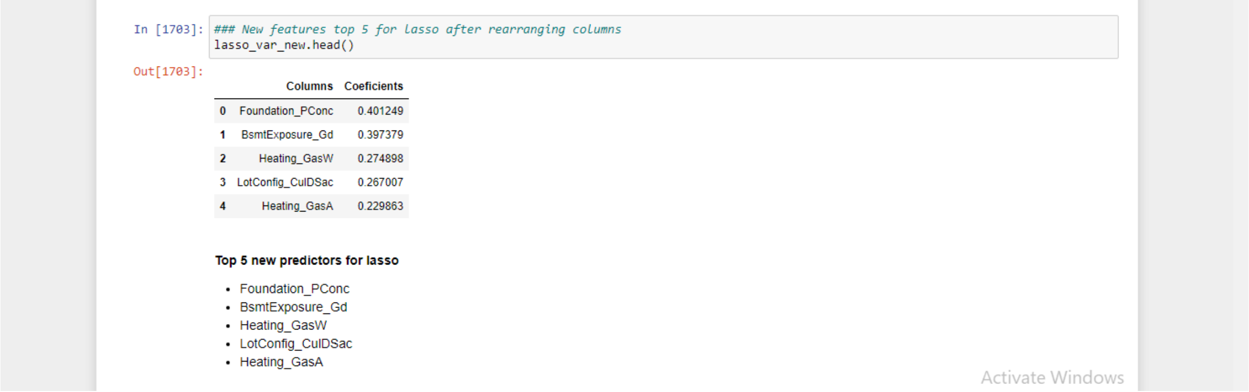


Top features after eliminating the original top features

```
In [1701]: lasso_coeff=pd.DataFrame({'Columns':X_train[new_features].columns,'Coeficients':lasso.coef_}).sort_values(by='Coeficients',ascend

In [1702]: ### Top 10 features
           lasso_coeff.head(10)
           lasso_coeff.reset_index(inplace=True)
           lasso_coeff.drop(columns='index',inplace=True)
           lasso_var_new=lasso_coeff.head(10)
           lasso_var_new

Out[1702]:
```

	Columns	Coeficients
0	Foundation_PConc	0.401249
1	BsmtExposure_Gd	0.397379
2	Heating_GasW	0.274898
3	LotConfig_CulDSac	0.267007
4	Heating_GasA	0.229863
5	SaleCondition_Partial	0.228615
6	GarageQual_TA	0.142278
7	Condition1_Other	0.121106
8	Condition1_Norm	0.119219

New Top 5 Predictors of lasso:



```
In [1703]: ### New features top 5 for lasso after rearranging columns
           lasso_var_new.head()

Out[1703]:
```

	Columns	Coeficients
0	Foundation_PConc	0.401249
1	BsmtExposure_Gd	0.397379
2	Heating_GasW	0.274898
3	LotConfig_CulDSac	0.267007
4	Heating_GasA	0.229863

Top 5 new predictors for lasso

- Foundation_PConc
- BsmtExposure_Gd
- Heating_GasW
- LotConfig_CulDSac
- Heating_GasA

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?

A model is said to be robust when the bias-variance trade-off is handled reducing the complexity of the model. More the complexity of the model, higher is the chance of overfitting to occur which results in poor performance on the unseen data. Hence an optimal value of lambda is chosen to bring down the coefficients close to zero (ridge) or zero (lasso) by compromising on the error to avoid overfitting of the variables.

The accuracy of the model for both ridge and lasso techniques (80%) was found to be quite similar on training as well as on test set, indicating that the model doesn’t overfit and provides expected accuracy on the unseen test data. This generalization was achieved by regularization making the model robust to any incoming unknown data.