Assignment-based Subjective Questions

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?

Optimal Value of alpha for ridge and lasso regression are:

- Optimal Value of lambda for Ridge: 6
- Optimal Value of lambda for Lasso: 0.0001

If we choose to double the value of alpha for both ridge and lasso: in case of Ridge that will lower the coefficients and in case of Lasso, there would be more less important features coefficients turning 0.

The most important predictor variable after the change is implemented are those which are significant.

Top 10 most significant variables in Ridge are:

In [402]:
```
## View the top 10 coefficients of Ridge regression in descending order
betas['Ridge'].sort_values(ascending=False)[:10]
```

Out[402]:
```
OverallQual_Very Excellent    0.057847
Neighborhood_NoRidge          0.057122
OverallQual_Excellent         0.051892
2ndFlrSF                      0.051019
GrLivArea                     0.050572
FullBath                      0.042332
GarageCars                    0.037787
1stFlrSF                      0.037332
TotRmsAbvGrd                  0.035781
OverallQual_Very Good         0.032714
Name: Ridge, dtype: float64
```

Top 10 most significant variables in Lasso are:

In [403]:
```
## View the top 10 coefficients of Lasso in descending order
betas['Lasso'].sort_values(ascending=False)[:10]
```

Out[403]:
```
GrLivArea                     0.257845
OverallQual_Very Excellent    0.123535
OverallQual_Excellent         0.104610
Neighborhood_NoRidge          0.068823
GarageCars                    0.061139
OverallQual_Very Good         0.049021
FullBath                      0.036673
Neighborhood_Crawfor          0.034841
Neighborhood_NridgHt          0.034288
2ndFlrSF                      0.028602
Name: Lasso, dtype: float64
```

The number of zero coefficients increased from earlier 51 to 72. Also, there is a reduction in the high value of coefficients after Ridge is applied with double the alpha value.

In [399]:
```
## View the features removed by Lasso
print(betas[betas['Lasso']==0].shape)
betas[betas['Lasso']==0]
```

(72, 3)

Out[399]:

	Linear	Ridge	Lasso
LotFrontage	-7.382836e-02	0.000691	-0.0
BsmtFinSF1	2.453099e+11	0.013941	0.0
BsmtFinSF2	6.406571e+10	0.007847	0.0
TotalBsmtSF	-2.655641e+11	0.022404	0.0
1stFlrSF	-2.790336e+09	0.037332	0.0
BsmtHalfBath	1.216412e-02	0.002055	0.0
BedroomAbvGr	-1.091003e-02	0.009331	0.0
GarageArea	-2.926636e-02	0.024246	0.0
OpenPorchSF	-1.258540e-02	0.002667	-0.0
EnclosedPorch	2.044916e-02	0.003772	0.0

Below are the snapshots from Jupyter Notebook, capturing key information when the lambda for Ridge and Lasso are changed to double their initial value.

```python
#Fitting Ridge model for optimal value of alpha and printing coefficients which have been penalised

alpha = 12
# Create a ridge regreesion instance with optimum value of alpha
ridge = Ridge(alpha=alpha)

# Fit the model on training data
ridge.fit(X_train, y_train)

# View the coefficients of ridge regression fitted model
print(ridge.coef_)
```

```
In [380]:   1 ridge.score(X_train,y_train)
Out[380]: 0.8698606374816347

In [381]:   1 ridge.score(X_test,y_test)
Out[381]: 0.8569085311544
```

```
R-Squared (Train) = 0.87
R-Squared (Test) = 0.86
RSS (Train) = 1.60
RSS (Test) = 0.78
MSE (Train) = 0.00
MSE (Test) = 0.00
```

```
[('constant', 0.18),
 ('LotFrontage', 0.001),
 ('LotArea', 0.019),
 ('MasVnrArea', 0.025),
 ('BsmtFinSF1', 0.014),
 ('BsmtFinSF2', 0.008),
 ('BsmtUnfSF', 0.02),
 ('TotalBsmtSF', 0.022),
 ('1stFlrSF', 0.037),
 ('2ndFlrSF', 0.051),
 ('LowQualFinSF', -0.007),
 ('GrLivArea', 0.051),
 ('BsmtFullBath', 0.023),
 ('BsmtHalfBath', 0.002),
 ('FullBath', 0.042),
 ('HalfBath', 0.017),
 ('BedroomAbvGr', 0.009),
 ('KitchenAbvGr', -0.012),
 ('TotRmsAbvGrd', 0.036),
 ('Fireplaces', 0.02),
 ('GarageCars', 0.038),
 ('GarageArea', 0.024),
 ('WoodDeckSF', 0.017),
 ('OpenPorchSF', 0.003),
 ('EnclosedPorch', 0.004),
 ('3SsnPorch', 0.008),
 ('ScreenPorch', 0.008),
 ('PoolArea', 0.003),
 ('MiscVal', -0.001),
 ('YearSinceRemodelAtSale', -0.017),
```

```
 ('AgeAtSale', -0.014),
 ('MSSubClass_1-1/2 STORY FINISHED ALL AGES', 0.006),
 ('MSSubClass_1-STORY 1945 & OLDER', -0.006),
 ('MSSubClass_1-STORY 1946 & NEWER ALL STYLES', 0.012),
 ('MSSubClass_1-STORY PUD (Planned Unit Development) - 1946 & NEWER', -0.0
1),
 ('MSSubClass_1-STORY W/FINISHED ATTIC ALL AGES', 0.003),
 ('MSSubClass_2 FAMILY CONVERSION - ALL STYLES AND AGES', -0.005),
 ('MSSubClass_2-1/2 STORY ALL AGES', 0.004),
 ('MSSubClass_2-STORY 1945 & OLDER', 0.005),
 ('MSSubClass_2-STORY 1946 & NEWER', 0.005),
 ('MSSubClass_2-STORY PUD - 1946 & NEWER', -0.009),
 ('MSSubClass_DUPLEX - ALL STYLES AND AGES', -0.004),
 ('MSSubClass_PUD - MULTILEVEL - INCL SPLIT LEV/FOYER', -0.005),
 ('MSSubClass_SPLIT FOYER', -0.003),
 ('MSSubClass_SPLIT OR MULTI-LEVEL', -0.001),
 ('MSZoning_Others', -0.008),
 ('MSZoning_RL', 0.003),
 ('MSZoning_RM', -0.005),
 ('LotShape_IR2', 0.005),
 ('LotShape_IR3', -0.012),
 ('LotShape_Reg', -0.0),
 ('LotConfig_CulDSac', 0.013),
 ('LotConfig_FR2', -0.015),
 ('LotConfig_FR3', -0.004),
 ('LotConfig_Inside', -0.002),
 ('Neighborhood_Blueste', -0.001),
 ('Neighborhood_BrDale', -0.001),
 ('Neighborhood_BrkSide', -0.0),
 ('Neighborhood_ClearCr', 0.006),
 ('Neighborhood_CollgCr', -0.007),
 ('Neighborhood_Crawfor', 0.026),
 ('Neighborhood_Edwards', -0.026),
 ('Neighborhood_Gilbert', -0.012),
 ('Neighborhood_IDOTRR', -0.012),
 ('Neighborhood_MeadowV', -0.01),
 ('Neighborhood_Mitchel', -0.013),
 ('Neighborhood_NAmes', -0.012),
 ('Neighborhood_NPkVill', -0.002),
 ('Neighborhood_NWAmes', -0.006),
 ('Neighborhood_NoRidge', 0.057),
 ('Neighborhood_NridgHt', 0.031),
 ('Neighborhood_OldTown', -0.015),
 ('Neighborhood_SWISU', -0.003),
 ('Neighborhood_Sawyer', -0.015),
 ('Neighborhood_SawyerW', -0.003),
 ('Neighborhood_Somerst', 0.015),
 ('Neighborhood_StoneBr', 0.017),
 ('Neighborhood_Timber', -0.004),
 ('Neighborhood_Veenker', 0.01),
 ('BldgType_2fmCon', -0.005),
 ('BldgType_Duplex', -0.004),
 ('BldgType_Twnhs', -0.012),
 ('BldgType_TwnhsE', -0.01),
 ('HouseStyle_1Story', 0.004),
 ('HouseStyle_2Story', -0.005),
```

```
('HouseStyle_Others', -0.002),
('HouseStyle_SLvl', -0.005),
('OverallQual_Average', -0.008),
('OverallQual_Below Average', -0.009),
('OverallQual_Excellent', 0.052),
('OverallQual_Fair', -0.013),
('OverallQual_Good', 0.01),
('OverallQual_Poor', -0.007),
('OverallQual_Very Excellent', 0.058),
('OverallQual_Very Good', 0.033),
('OverallQual_Very Poor', -0.005),
('OverallCond_Average', -0.009),
('OverallCond_Below Average', -0.011),
('OverallCond_Excellent', 0.016),
('OverallCond_Fair', -0.017),
('OverallCond_Good', 0.005),
('OverallCond_Poor', -0.001),
('OverallCond_Very Good', -0.0),
('OverallCond_Very Poor', -0.005),
('RoofStyle_Hip', 0.005),
('RoofStyle_Others', 0.003),
('Exterior1st_CemntBd', -0.003),
('Exterior1st_HdBoard', -0.011),
('Exterior1st_MetalSd', -0.007),
('Exterior1st_Others', -0.014),
('Exterior1st_Plywood', -0.004),
('Exterior1st_VinylSd', -0.006),
('Exterior1st_Wd Sdng', -0.003),
('Exterior2nd_CmentBd', 0.001),
('Exterior2nd_HdBoard', 0.004),
('Exterior2nd_MetalSd', 0.001),
('Exterior2nd_Others', 0.011),
('Exterior2nd_Plywood', -0.004),
('Exterior2nd_VinylSd', 0.002),
('Exterior2nd_Wd Sdng', -0.002),
('Exterior2nd_Wd Shng', -0.008),
('MasVnrType_BrkFace', 0.004),
('MasVnrType_None', 0.006),
('MasVnrType_Stone', -0.0),
('ExterQual_Fa', -0.017),
('ExterQual_Gd', -0.004),
('ExterQual_TA', -0.016),
('Foundation_CBlock', 0.003),
('Foundation_Others', 0.003),
('Foundation_PConc', 0.005),
('BsmtQual_Fa', -0.019),
('BsmtQual_Gd', -0.033),
('BsmtQual_No Basement', -0.015),
('BsmtQual_TA', -0.028),
('BsmtExposure_Gd', 0.029),
('BsmtExposure_Mn', -0.004),
('BsmtExposure_No', -0.013),
('BsmtExposure_No Basement', -0.016),
('BsmtFinType1_BLQ', 0.001),
('BsmtFinType1_GLQ', 0.013),
('BsmtFinType1_LwQ', -0.004),
```

```
('BsmtFinType1_No Basement', -0.015),
('BsmtFinType1_Rec', -0.001),
('BsmtFinType1_Unf', -0.01),
('HeatingQC_Fa', -0.005),
('HeatingQC_Gd', -0.004),
('HeatingQC_Po', -0.002),
('HeatingQC_TA', -0.004),
('KitchenQual_Fa', -0.024),
('KitchenQual_Gd', -0.027),
('KitchenQual_TA', -0.029),
('FireplaceQu_None', -0.009),
('FireplaceQu_Others', 0.005),
('FireplaceQu_TA', -0.001),
('GarageType_BuiltIn', -0.001),
('GarageType_Detchd', -0.007),
('GarageType_No Garage', -0.004),
('GarageType_Others', -0.01),
('GarageFinish_No Garage', -0.004),
('GarageFinish_RFn', -0.008),
('GarageFinish_Unf', -0.011),
('Fence_None', -0.001),
('Fence_Others', -0.007),
('SaleCondition_Normal', 0.002),
('SaleCondition_Others', 0.002),
('SaleCondition_Partial', 0.011)]
```

In [388]:

```
#optimum alpha

alpha = 0.0002
# Create a lasso regreesion instance with optimum value of alpha
lasso = Lasso(alpha=alpha)

# Fit the model on training data
lasso.fit(X_train, y_train)
```

Out[388]:

```
Lasso
Lasso(alpha=0.0002)
```

In [390]:

```
lasso.score(X_train,y_train)
```

Out[390]: 0.8767438395130009

In [391]:

```
lasso.score(X_test,y_test)
```

Out[391]: 0.8594790000453509

```
R-Squared (Train) = 0.88
R-Squared (Test) = 0.86
RSS (Train) = 1.52
RSS (Test) = 0.76
MSE (Train) = 0.00
MSE (Test) = 0.00
```

```
[('constant', 0.127),
 ('LotFrontage', -0.0),
 ('LotArea', 0.006),
 ('MasVnrArea', 0.004),
 ('BsmtFinSF1', 0.0),
 ('BsmtFinSF2', 0.0),
 ('BsmtUnfSF', 0.002),
```

```
 ('TotalBsmtSF', 0.0),
 ('1stFlrSF', 0.0),
 ('2ndFlrSF', 0.029),
 ('LowQualFinSF', -0.008),
 ('GrLivArea', 0.258),
 ('BsmtFullBath', 0.024),
 ('BsmtHalfBath', 0.0),
 ('FullBath', 0.037),
 ('HalfBath', 0.012),
 ('BedroomAbvGr', 0.0),
 ('KitchenAbvGr', -0.024),
 ('TotRmsAbvGrd', 0.001),
 ('Fireplaces', 0.016),
 ('GarageCars', 0.061),
 ('GarageArea', 0.0),
 ('WoodDeckSF', 0.007),
 ('OpenPorchSF', -0.0),
 ('EnclosedPorch', 0.0),
 ('3SsnPorch', 0.0),
 ('ScreenPorch', 0.005),
 ('PoolArea', -0.0),
 ('MiscVal', -0.0),
 ('YearSinceRemodelAtSale', -0.013),
 ('AgeAtSale', -0.023),
 ('MSSubClass_1-1/2 STORY FINISHED ALL AGES', 0.003),
 ('MSSubClass_1-STORY 1945 & OLDER', -0.0),
 ('MSSubClass_1-STORY 1946 & NEWER ALL STYLES', 0.019),
 ('MSSubClass_1-STORY PUD (Planned Unit Development) - 1946 & NEWER', -0.0
),
 ('MSSubClass_1-STORY W/FINISHED ATTIC ALL AGES', 0.0),
 ('MSSubClass_2 FAMILY CONVERSION - ALL STYLES AND AGES', -0.0),
 ('MSSubClass_2-1/2 STORY ALL AGES', -0.0),
 ('MSSubClass_2-STORY 1945 & OLDER', 0.0),
 ('MSSubClass_2-STORY 1946 & NEWER', 0.0),
 ('MSSubClass_2-STORY PUD - 1946 & NEWER', -0.007),
 ('MSSubClass_DUPLEX - ALL STYLES AND AGES', -0.0),
 ('MSSubClass_PUD - MULTILEVEL - INCL SPLIT LEV/FOYER', -0.0),
 ('MSSubClass_SPLIT FOYER', 0.0),
 ('MSSubClass_SPLIT OR MULTI-LEVEL', 0.0),
 ('MSZoning_Others', -0.002),
 ('MSZoning_RL', 0.007),
 ('MSZoning_RM', -0.0),
 ('LotShape_IR2', 0.003),
 ('LotShape_IR3', -0.005),
 ('LotShape_Reg', 0.0),
 ('LotConfig_CulDSac', 0.013),
 ('LotConfig_FR2', -0.007),
 ('LotConfig_FR3', -0.0),
 ('LotConfig_Inside', -0.0),
 ('Neighborhood_Blueste', -0.0),
 ('Neighborhood_BrDale', 0.0),
 ('Neighborhood_BrkSide', 0.0),
 ('Neighborhood_ClearCr', 0.011),
 ('Neighborhood_CollgCr', 0.0),
 ('Neighborhood_Crawfor', 0.035),
 ('Neighborhood_Edwards', -0.024),
```

```
('Neighborhood_Gilbert', -0.0),
('Neighborhood_IDOTRR', -0.004),
('Neighborhood_MeadowV', -0.0),
('Neighborhood_Mitchel', -0.004),
('Neighborhood_NAmes', -0.005),
('Neighborhood_NPkVill', -0.0),
('Neighborhood_NWAmes', -0.0),
('Neighborhood_NoRidge', 0.069),
('Neighborhood_NridgHt', 0.034),
('Neighborhood_OldTown', -0.014),
('Neighborhood_SWISU', -0.0),
('Neighborhood_Sawyer', -0.01),
('Neighborhood_SawyerW', 0.0),
('Neighborhood_Somerst', 0.023),
('Neighborhood_StoneBr', 0.008),
('Neighborhood_Timber', 0.0),
('Neighborhood_Veenker', 0.005),
('BldgType_2fmCon', -0.0),
('BldgType_Duplex', -0.0),
('BldgType_Twnhs', -0.011),
('BldgType_TwnhsE', -0.005),
('HouseStyle_1Story', 0.002),
('HouseStyle_2Story', -0.005),
('HouseStyle_Others', -0.0),
('HouseStyle_SLvl', -0.0),
('OverallQual_Average', -0.003),
('OverallQual_Below Average', -0.004),
('OverallQual_Excellent', 0.105),
('OverallQual_Fair', -0.001),
('OverallQual_Good', 0.017),
('OverallQual_Poor', -0.0),
('OverallQual_Very Excellent', 0.124),
('OverallQual_Very Good', 0.049),
('OverallQual_Very Poor', -0.0),
('OverallCond_Average', -0.011),
('OverallCond_Below Average', -0.012),
('OverallCond_Excellent', 0.009),
('OverallCond_Fair', -0.017),
('OverallCond_Good', 0.004),
('OverallCond_Poor', -0.0),
('OverallCond_Very Good', 0.001),
('OverallCond_Very Poor', -0.0),
('RoofStyle_Hip', 0.0),
('RoofStyle_Others', 0.0),
('Exterior1st_CemntBd', -0.0),
('Exterior1st_HdBoard', -0.0),
('Exterior1st_MetalSd', -0.0),
('Exterior1st_Others', -0.011),
('Exterior1st_Plywood', -0.0),
('Exterior1st_VinylSd', 0.0),
('Exterior1st_Wd Sdng', -0.0),
('Exterior2nd_CmentBd', 0.0),
('Exterior2nd_HdBoard', 0.0),
('Exterior2nd_MetalSd', -0.0),
('Exterior2nd_Others', 0.008),
('Exterior2nd_Plywood', -0.0),
```

```
('Exterior2nd_VinylSd', 0.004),
('Exterior2nd_Wd Sdng', -0.0),
('Exterior2nd_Wd Shng', -0.002),
('MasVnrType_BrkFace', 0.0),
('MasVnrType_None', 0.0),
('MasVnrType_Stone', -0.002),
('ExterQual_Fa', -0.004),
('ExterQual_Gd', 0.006),
('ExterQual_TA', -0.0),
('Foundation_CBlock', 0.0),
('Foundation_Others', -0.0),
('Foundation_PConc', 0.0),
('BsmtQual_Fa', -0.014),
('BsmtQual_Gd', -0.026),
('BsmtQual_No Basement', -0.029),
('BsmtQual_TA', -0.021),
('BsmtExposure_Gd', 0.027),
('BsmtExposure_Mn', -0.0),
('BsmtExposure_No', -0.01),
('BsmtExposure_No Basement', -0.016),
('BsmtFinType1_BLQ', 0.0),
('BsmtFinType1_GLQ', 0.009),
('BsmtFinType1_LwQ', -0.001),
('BsmtFinType1_No Basement', -0.0),
('BsmtFinType1_Rec', -0.0),
('BsmtFinType1_Unf', -0.008),
('HeatingQC_Fa', -0.0),
('HeatingQC_Gd', -0.001),
('HeatingQC_Po', -0.0),
('HeatingQC_TA', -0.001),
('KitchenQual_Fa', -0.025),
('KitchenQual_Gd', -0.021),
('KitchenQual_TA', -0.025),
('FireplaceQu_None', -0.006),
('FireplaceQu_Others', 0.0),
('FireplaceQu_TA', -0.0),
('GarageType_BuiltIn', -0.002),
('GarageType_Detchd', -0.004),
('GarageType_No Garage', -0.0),
('GarageType_Others', -0.007),
('GarageFinish_No Garage', -0.0),
('GarageFinish_RFn', -0.003),
('GarageFinish_Unf', -0.008),
('Fence_None', -0.0),
('Fence_Others', -0.003),
('SaleCondition_Normal', 0.003),
('SaleCondition_Others', 0.0),
('SaleCondition_Partial', 0.013)]
```

Out[394]:

	Metric	Linear Regression	Ridge Regression	Lasso Regression
0	R2 Score (Train)	8.935476e-01	0.869861	0.876744
1	R2 Score (Test)	-9.550277e+19	0.856909	0.859479
2	RSS (Train)	1.309900e+00	1.601369	1.516671
3	RSS (Test)	5.191360e+20	0.777820	0.763847
4	MSE (Train)	3.581840e-02	0.039603	0.038542
5	MSE (Test)	1.088688e+09	0.042141	0.041761

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?**

Optimal Value of alpha for ridge and lasso regression are:

• Optimal Value of lambda for ridge: 6

• Optimal Value of lambda for Lasso: 10

As we got good score for both the models so we can go with Lasso regression as it results in model parameters such that lesser important features coefficients become zero.

Also, the model we choose to apply will depend on the use case. If we have too many variables and one of our primary goals is feature selection, then we will use Lasso. If we don't want to get too large coefficients and reduction of coefficient magnitude is one of our prime goals, then we will use Ridge Regression.
Ridge: Train :88.0 Test :86.5 and **Lasso:** Train :86.5 Test :86.7

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?**

Below is the list of predictor variables after the first execution of Lasso model.

Top 6 parameters – Lasso	
GrLivArea	0.250554
OverallQual_Very Excellent	0.118687
OverallQual_Excellent	0.099083
Neighborhood_NoRidge	0.071331
LotArea	0.064057
GarageCars	0.059015

Assuming that completely new set of 5 predictor variables is now expected, we remove OverallQual and Neighborhood altogether rather than only removing the specific dummy variable as listed above. Also, the above list contains 6 variables instead of 5 as we consider the two categories of OverquallQual as one single variable.

On running the same notebook and removing the top 5 significant variables, we found below variables as next 5 significant.

Top 5 most significant variables in Lasso are:

In [223]:
```
## View the top 10 coefficients of Lasso in descending order
betas['Lasso'].sort_values(ascending=False)[:5]
```

Out[223]:
```
1stFlrSF      0.247630
2ndFlrSF      0.173814
MasVnrArea    0.078549
GarageArea    0.074452
FullBath      0.041634
Name: Lasso, dtype: float64
```

Below are the snapshots from Jupyter Notebook, capturing key information when the top 5 predictor variables obtained from Lasso regression are removed from the dataset and Lasso regression is again performed.

Deleted top 5 predictor variables before creation of dummy variables.

```
In [171]:
surprise_housing_dataset.drop(['LotArea', 'GrLivArea','Neighborhood', 'OverallQual','GarageCars'],axis=1,inplace=True)

surprise_housing_dataset.head()
```

Out[171]:

	Id	MSSubClass	MSZoning	LotFrontage	LotShape	LotConfig	BldgType	HouseStyle	OverallCond	RoofStyle	Exterior1st	Exterior2nd	MasVnrType	M
0	1	2-STORY 1946 & NEWER	RL	65.0	Reg	Inside	1Fam	2Story	Average	Gable	VinylSd	VinylSd	BrkFace	
1	2	1-STORY 1946 & NEWER ALL STYLES	RL	80.0	Reg	FR2	1Fam	1Story	Very Good	Gable	MetalSd	MetalSd	None	
2	3	2-STORY 1946 & NEWER	RL	68.0	IR1	Inside	1Fam	2Story	Average	Gable	VinylSd	VinylSd	BrkFace	
3	4	2-STORY 1945 & OLDER	RL	60.0	IR1	Corner	1Fam	2Story	Average	Gable	Wd Sdng	Wd Shng	None	
4	5	2-STORY 1946 & NEWER	RL	84.0	IR1	FR2	1Fam	2Story	Average	Gable	VinylSd	VinylSd	BrkFace	

```
In [172]:
surprise_housing_dataset.info(verbose=True, show_counts=True)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 52 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Id                     1460 non-null   int64
 1   MSSubClass             1460 non-null   object
```

Latest set of variables.

```
In [174]:
#Checking all categorical columns to form dummy variables

cat_cols = surprise_housing_dataset.select_dtypes(include=['object','category'])
print(cat_cols.columns.size)
cat_cols.columns

23

Out[174]: Index(['MSSubClass', 'MSZoning', 'LotShape', 'LotConfig', 'BldgType',
        'HouseStyle', 'OverallCond', 'RoofStyle', 'Exterior1st', 'Exterior2nd',
        'MasVnrType', 'ExterQual', 'Foundation', 'BsmtQual', 'BsmtExposure',
        'BsmtFinType1', 'HeatingQC', 'KitchenQual', 'FireplaceQu', 'GarageType',
        'GarageFinish', 'Fence', 'SaleCondition'],
       dtype='object')
```

```
In [175]:
#Checking all numerical variables

num_cols = surprise_housing_dataset.select_dtypes(include=['int64','float64'])
print(num_cols.columns.size)
num_cols.columns

29

Out[175]: Index(['Id', 'LotFrontage', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2',
        'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',
        'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr',
        'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageArea',
        'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
        'ScreenPorch', 'PoolArea', 'MiscVal', 'SalePrice',
        'YearSinceRemodelAtSale', 'AgeAtSale'],
       dtype='object')
```

```
In [210]:
lasso.score(X_train,y_train)

Out[210]: 0.841458627562576

In [211]:
lasso.score(X_test,y_test)

Out[211]: 0.8427746308203572
```

Optimum alpha remains same.

In [207]:
```python
# Printing the best hyperparameter alpha
print(model_cv.best_params_)
```

```
{'alpha': 0.0001}
```

In [208]:
```python
#optimum alpha

alpha = 0.0001
# Create a lasso regreesion instance with optimum value of alpha
lasso = Lasso(alpha=alpha)

# Fit the model on training data
lasso.fit(X_train, y_train)
```

Out[208]:
```
Lasso
Lasso(alpha=0.0001)
```

In [209]:
```python
# View the coefficients of lasso fitted model
lasso.coef_
```

Out[209]:
```
array([-6.68193272e-03,  7.85489070e-02,  0.00000000e+00,  7.35836887e-03,
        2.50475467e-02,  0.00000000e+00,  2.47629817e-01,  1.73814357e-01,
       -0.00000000e+00,  2.37966926e-02,  0.00000000e+00,  4.16342480e-02,
        1.04122321e-02, -0.00000000e+00, -5.50291601e-02,  7.25458338e-03,
        3.16165102e-02,  7.44516404e-02,  1.32349528e-02, -4.89352646e-03,
        0.00000000e+00,  0.00000000e+00,  3.21912156e-03, -0.00000000e+00,
       -0.00000000e+00, -7.80576027e-03, -4.45578194e-02,  0.00000000e+00,
       -0.00000000e+00,  9.23039847e-03, -0.00000000e+00,  0.00000000e+00,
       -0.00000000e+00,  0.00000000e+00,  1.66087289e-02,  0.00000000e+00,
       -0.00000000e+00, -1.68764757e-03, -0.00000000e+00, -2.35553316e-03,
```

Out[214]:

	Metric	Linear Regression	Ridge Regression	Lasso Regression
0	R2 Score (Train)	8.495093e-01	0.840442	0.841459
1	R2 Score (Test)	-2.631968e+18	0.840677	0.842775
2	RSS (Train)	1.851792e+00	1.963365	1.950856
3	RSS (Test)	1.430691e+19	0.866051	0.854649
4	MSE (Train)	4.258761e-02	0.043852	0.043712
5	MSE (Test)	1.807323e+08	0.044467	0.044173

Number of zero coefficients changed to 43.

In [219]:
```python
## View the features removed by Lasso
print(betas[betas['Lasso']==0].shape)
betas[betas['Lasso']==0]
```

```
(43, 3)
```

Out[219]:

	Linear	Ridge	Lasso
BsmtFinSF1	-4.808405e+11	0.024809	0.0
TotalBsmtSF	5.205414e+11	0.042899	0.0
LowQualFinSF	2.943180e-03	-0.008406	-0.0
BsmtHalfBath	3.595352e-04	-0.000859	0.0
BedroomAbvGr	-2.507305e-02	-0.002695	-0.0
EnclosedPorch	1.287866e-02	0.008370	0.0
3SsnPorch	2.249336e-02	0.013900	0.0
PoolArea	-2.987289e-02	-0.001747	-0.0
MiscVal	-8.257151e-03	-0.005663	-0.0
MSSubClass_1-1/2 STORY FINISHED ALL AGES	-4.489708e-02	-0.000978	0.0
MSSubClass_1-STORY 1945 & OLDER	-4.700518e-02	-0.009014	-0.0
MSSubClass_1-STORY PUD (Planned Unit Development) - 1946 & NEWER	-6.703186e-02	-0.011601	-0.0
MSSubClass_1-STORY W/FINISHED ATTIC ALL AGES	-5.088830e-02	0.003146	0.0
MSSubClass_2 FAMILY CONVERSION - ALL STYLES AND AGES	-3.782447e+09	-0.006660	-0.0
MSSubClass_2-1/2 STORY ALL AGES	-4.025841e-02	-0.001830	0.0
MSSubClass_2-STORY 1946 & NEWER	-5.986023e-02	-0.003034	0.0
MSSubClass_2-STORY PUD - 1946 & NEWER	-7.008362e-02	-0.008500	-0.0
MSSubClass_PUD - MULTILEVEL - INCL SPLIT LEV/FOYER	-7.228947e-02	-0.015231	-0.0
MSSubClass_SPLIT OR MULTILEVEL	-4.521942e-02	-0.006509	-0.0

Top 10 most significant variables in Ridge are:

```
In [221]:
## View the top 10 coefficients of Ridge regression in descending order
betas['Ridge'].sort_values(ascending=False)[:10]
```

```
Out[221]: 2ndFlrSF          0.117354
          1stFlrSF          0.097380
          GarageArea        0.072756
          MasVnrArea        0.072739
          FullBath          0.056811
          TotRmsAbvGrd      0.050789
          TotalBsmtSF       0.042899
          BsmtUnfSF         0.040652
          BsmtExposure_Gd   0.032728
          Fireplaces        0.031864
          Name: Ridge, dtype: float64
```

Top 5 most significant variables in Lasso are:

```
In [223]:
## View the top 10 coefficients of Lasso in descending order
betas['Lasso'].sort_values(ascending=False)[:5]
```

```
Out[223]: 1stFlrSF      0.247630
          2ndFlrSF      0.173814
          MasVnrArea    0.078549
          GarageArea    0.074452
          FullBath      0.041634
          Name: Lasso, dtype: float64
```

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 robust when any variation in the predictor variables data does not affect its performance considerably.
- A generalized model can adapt properly to new, previously unseen data, drawn from the same distribution as the one used to create the model. This is the test data out of the entire dataset.
- To ensure that a model is robust and generalizable, we must ensure that it does not overfit. This is because an overfitting model has very high variance due to which a slight change in data and its underlying pattern affects the model prediction heavily. Though such a model can properly identify all the patterns of a training data, but it fails to identify the patterns when made to run on unseen test data.
- To state in a different way, a model should not be too complex to be robust and generalizable. The model should be as simple as possible, though this may reduce its accuracy as a trade-off, but it will be more robust and generalisable.
- **From accuracy point of view,** a too complex model will have a very high accuracy. This can be also understood using the Bias-Variance trade-off.
- So, to make a model more robust and generalizable, we will have to decrease variance which will lead to some amount of bias. Addition of bias will consequently reduce the accuracy of the model.
- The simpler the model the more the bias it will acquire but lesser will be its variance and hence more generalizable. Its implication in terms of accuracy is that a robust and generalisable model

will perform equally well on both training and test data i.e. the accuracy does not change much for training and test data.

- In general, we must find some balance between model accuracy and complexity. It is important to have balance in Bias and Variance to avoid overfitting and under-fitting of data. This can be achieved by Regularization techniques like Ridge Regression and Lasso.
- **Bias:** Bias is the error in a model due to which the model is weak to learn from the dataset. High bias means model is unable to learn details of the underlying pattern in the data. As a result, the model will perform poorly on training and testing data.
- **Variance:** Variance is the error in a model due to which the model tries to over learn from the data. High variance means model performs exceptionally well on training data as it has very well trained on the training dataset and its underlying patterns, but it performs poorly on test data which is unknown for the model.