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 1:

- Optimal alpha value for ridge regression is 10
- Optimal alpha value for lasso regression is 100

After doubling the alpha value for ridge

jupyter Prasoon_Mahawar_Surprise_Housing_Assignment Last Checkpoint: a day ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

After doubling alpha value

In [49]:
```
ridge = Ridge(alpha=20)
ridge.fit(X_train, y_train)
y_train_pred = ridge.predict(X_train)
y_pred = ridge.predict(X_test)
```

In [50]:
```
print("R-Squared of train data =", '%.5f' % r2_score(y_train, y_train_pred))
print("R-Squared of test data =", '%.5f' % r2_score(y_test, y_pred))

rss_train = np.sum(np.square(y_train - y_train_pred))
rss_test = np.sum(np.square(y_test - y_pred))
print("Residual sum of squares of train data =", '%.5f' % rss_train)
print("Residual sum of squares of test data =", '%.5f' % rss_test)


mse_train = mean_squared_error(y_train, y_train_pred)
mse_test = mean_squared_error(y_test, y_pred)
print("Mean Squared Error of train data =", '%.5f' % mse_train)
print("Mean Squared Error of test data =", '%.5f' % mse_test)


rmse_train = mse_train**0.5
rmse_test = mse_test**0.5
print("Root Mean Squared Error for train data =", '%.5f' % rmse_train)
print("Root Mean Squared Error for test data =", '%.5f' % rmse_test)
```

```
R-Squared of train data = 0.93842
R-Squared of test data = 0.89307
Residual sum of squares of train data = 264420114628.80493
Residual sum of squares of test data = 190244645493.22272
Mean Squared Error of train data = 258981503.06445
Mean Squared Error of test data = 434348505.69229
Root Mean Squared Error for train data = 16092.90226
Root Mean Squared Error for test data = 20841.02938
```

Changes in ridge regression:

- R2 score of train set decreased from 0.94 to 0.93
- R2 score of test set remained same at 0.89

After doubling the alpha value for lasso

jupyter Prasoon_Mahawar_Surprise_Housing_Assignment Last Checkpoint: a day ago (autosaved)

In [53]:
```
lasso = Lasso(alpha=200)
lasso.fit(X_train, y_train)
y_train_pred_lasso = lasso.predict(X_train)
y_pred_lasso = lasso.predict(X_test)
```

In [54]:
```
print("R-Squared of train data =", '%.5f' % r2_score(y_train, y_train_pred_lasso))
print("R-Squared of test data =", '%.5f' % r2_score(y_test, y_pred_lasso))

rss_train_lasso = np.sum(np.square(y_train - y_train_pred_lasso))
rss_test_lasso = np.sum(np.square(y_test - y_pred_lasso))
print("Residual sum of squares of train data =", '%.5f' % rss_train_lasso)
print("Residual sum of squares of test data =", '%.5f' % rss_test_lasso)


mse_train_lasso = mean_squared_error(y_train, y_train_pred_lasso)
mse_test_lasso = mean_squared_error(y_test, y_pred_lasso)
print("Mean Squared Error of train data =", '%.5f' % mse_train_lasso)
print("Mean Squared Error of test data =", '%.5f' % mse_test_lasso)


rmse_train_lasso = mse_train**0.5
rmse_test_lasso = mse_test**0.5
print("Root Mean Squared Error for train data =", '%.5f' % rmse_train_lasso)
print("Root Mean Squared Error for test data =", '%.5f' % rmse_test_lasso)
```

```
R-Squared of train data = 0.92598
R-Squared of test data = 0.88873
Residual sum of squares of train data = 317829272756.98334
Residual sum of squares of test data = 197974070561.79901
Mean Squared Error of train data = 311292137.86188
Mean Squared Error of test data = 451995594.88995
Root Mean Squared Error for train data = 16092.90226
Root Mean Squared Error for test data = 20841.02938
```

In [ ]:

Changes in lasso regression:

- R2 score of train set decreased from 0.94 to 0.92
- R2 score of test set increased from at 0.87 to 0.88

Most important predictor variables after doubling the alpha value

jupyter Prasoon_Mahawar_Surprise_Housing_Assignment Last Checkpoint: a day ago (unsaved changes)

BsmtUnfSF	-2085.023791	-13.940892
TotalBsmtSF	10209.646332	9021.795416
1stFlrSF	1985.394145	-0.000000
2ndFlrSF	6341.977544	1966.882166
LowQualFinSF	0.000000	0.000000

In [56]:
```
# Viewing new top 5 for ridge
comparison['Ridge'].sort_values(ascending=False)[:5]
```

Out[56]:
```
OverallQual_9           18316.435308
GrLivArea               16920.527827
OverallQual_8           15285.964118
Neighborhood_Crawfor    11425.949956
Functional_Typ          10706.145029
Name: Ridge, dtype: float64
```

In [57]:
```
# Viewing new top 5 for lasso
comparison['Lasso'].sort_values(ascending=False)[:5]
```

Out[57]:
```
OverallQual_9           44742.310546
OverallQual_8           30431.672747
GrLivArea               20724.902417
Neighborhood_Crawfor    15893.976267
Functional_Typ          13989.657544
Name: Lasso, dtype: float64
```

In [ ]:

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 2:

- Lasso regression is used when there are too many variables and we want to select features from it.
- Ridge regression is used when we want to reduce the coefficient magnitude.

So as per the problem statement we need to tell variables are significant in predicting the price of a house so we will apply Lasso regression.

Question 3

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 3

After performing the analysis as shown in the image below

jupyter Prasoon_Mahawar_Surprise_Housing_Assignment Last Checkpoint: Last Monday at 23:10 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [55]:
```
# For question 3

# Removing the top 5 variables of lasso

top5 = ['OverallQual_9', 'OverallQual_8', 'OverallCond_9', 'GrLivArea','Neighborhood_Crawfor']

X_train_dropped = X_train.drop(top5, axis=1)
X_test_dropped = X_test.drop(top5, axis=1)

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]}

lasso = Lasso()

# cross validation

lassoCV = GridSearchCV(estimator = lasso,
                       param_grid = params,
                       scoring= 'neg_mean_absolute_error',
                       cv = 5,
                       return_train_score=True,
                       verbose = 1, n_jobs=-1)
lassoCV.fit(X_train_dropped, y_train)
```

```
ergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of t
he features or consider increasing regularisation. Duality gap: 9.718e+08, tolerance: 3.466e+08
  model = cd_fast.enet_coordinate_descent(
/Users/prasoonmahawar/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_coordinate_descent.py:647: Conv
ergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of t
he features or consider increasing regularisation. Duality gap: 3.944e+09, tolerance: 3.477e+08
  model = cd_fast.enet_coordinate_descent(

/Users/prasoonmahawar/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_coordinate_descent.py:647: Conv
ergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of t
he features or consider increasing regularisation. Duality gap: 6.624e+08, tolerance: 3.477e+08
  model = cd_fast.enet_coordinate_descent(
```

Out[55]:
```
GridSearchCV(cv=5, estimator=Lasso(), n_jobs=-1,
             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,
```

jupyter Prasoon_Mahawar_Surprise_Housing_Assignment Last Checkpoint: Last Monday at 23:10 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

/Users/prasoonmahawar/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_coordinate_descent.py:647: Conv
ergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of t
he features or consider increasing regularisation. Duality gap: 6.624e+08, tolerance: 3.477e+08
  model = cd_fast.enet_coordinate_descent(

Out[55]:
```
GridSearchCV(cv=5, estimator=Lasso(), n_jobs=-1,
             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]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

In [56]:
```
lassoCV.best_params_
```

Out[56]:
```
{'alpha': 50}
```

In [57]:
```
lasso = Lasso(alpha=50)
lasso.fit(X_train_dropped, y_train)

## Make predictions
y_train_pred = lasso.predict(X_train_dropped)
y_pred = lasso.predict(X_test_dropped)
```

In [58]:
```
betas = pd.DataFrame(index=X_train_dropped.columns)
betas.rows = X_train_dropped.columns
betas['Lasso'] = lasso.coef_
betas['Lasso'].sort_values(ascending=False)[:5]
```

Out[58]:
```
Condition2_PosA        33844.334462
Exterior1st_BrkFace    19375.625098
2ndFlrSF               19358.853143
GarageYrBlt_1939       15312.433288
Functional_Typ         14344.484067
Name: Lasso, dtype: float64
```

In [ ]:

The new five most important variable will be:

Condition2_PosA, Exterior1st_BrkFace, 2ndFlrSF, GarageYrBlt_1939, Functional_Typ

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 4

To make sure a model is robust and generalizable, we have to take care it doesn't overfit. This is because an overfitting model has very high variance and a smallest change in data affects the model prediction heavily. Such a model will identify all the patterns of a training data, but fail to pick up the patterns in unseen test data. In other words, the model should not be too complex in order to be robust and generalizable.

Now a complex model will have a very high accuracy. So, to make our model more robust and generalizable, we will have to decrease variance which will lead to some bias. Addition of bias means that accuracy will decrease.

In general, we have to strike some balance between model accuracy and complexity. This can be achieved by Regularization techniques like Ridge Regression and Lasso.