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?

Ridge Alpha: 10.0 and Lasso Alpha:0.0001 and corresponding code are below:

Doubling the Alpha value for both

- The Optimal Value of alpha for ridge and lasso regression:
  - Ridge Alpha : 10.0
  - Lasso Alpha : 0.0001

```
# chossing Ridge Alpha = 20.0
ridge = Ridge(alpha=20.0)
ridge.fit(X_train, y_train)
```

```
Ridge(alpha=20.0)
```

```
#Ridge Regression Model Predicitons

print("For Ridge Regression Model :")
y_pred_train_ridge = ridge.predict(X_train)
y_pred_test_ridge = ridge.predict(X_test)

#R2 score for Ridge Regression Model

r2_score_train_ridge = r2_score(y_true=y_train, y_pred=y_pred_train_ridge)
r2_score_test_ridge = r2_score(y_true=y_test, y_pred=y_pred_test_ridge)

#MSE(Mean Squared Error) for Ridge Regression Model

mse_train_ridge = mean_squared_error(y_train, y_pred_train_ridge)
mse_test_ridge = mean_squared_error(y_test, y_pred_test_ridge)

#Mean Absolute Error for train and test sets

mae_train_ridge = mean_absolute_error(y_train,y_pred_train_ridge)
mae_test_ridge = mean_absolute_error(y_test,y_pred_test_ridge)

#RMSE(Root Mean Squared Error) for train and test sets

RMSE_train_ridge = np.sqrt(mse_train_ridge)
RMSE_test_ridge = np.sqrt(mse_test_ridge)
```

```
#RMSE(Root Mean Squared Error) for train and test sets

RMSE_train_ridge = np.sqrt(mse_train_ridge)
RMSE_test_ridge = np.sqrt(mse_test_ridge)

print("\n For Train Set: \n R2 Score : ",r2_score_train_ridge,"\nMSE Score : ",mse_train_ridge,"\nMAE Score : ",mae_train_ridge,
      "\nRMSE Score : ",RMSE_train_ridge)
print("\n For Test Set: \n R2 Score : ",r2_score_test_ridge,"\nMSE Score : ",mse_test_ridge,"\nMAE Score : ",mae_test_ridge,
      "\nRMSE Score : ",RMSE_test_ridge)
```

```
For Ridge Regression Model :

 For Train Set:
 R2 Score :  0.9254900162194803
MSE Score :  0.07450998378051962
MAE Score :  0.18950020988871152
RMSE Score :  0.2729651695372866

 For Test Set:
 R2 Score :  0.8767021368319561
MSE Score :  0.10584510615003154
MAE Score :  0.21598550901351718
RMSE Score :  0.32533844861932865
```

There is a very slight variation in R2 score and RMSE score for Test set is very slightly increased.

Lasso

```
#Chossing Lasso Alpha as 0.0002
lasso = Lasso(alpha=0.0002)

lasso.fit(X_train, y_train)
```

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

```
#Lasso Regression Model Predicitons

print("For Lasso Regression Model :")
y_pred_train_lasso = lasso.predict(X_train)
y_pred_test_lasso = lasso.predict(X_test)

#R2 score for Lasso Regression Model

r2_score_train_lasso = r2_score(y_true=y_train, y_pred=y_pred_train_lasso)
r2_score_test_lasso = r2_score(y_true=y_test, y_pred=y_pred_test_lasso)

#MSE(Mean Squared Error) for Lasso Regression Model

mse_train_lasso = mean_squared_error(y_train, y_pred_train_lasso)
mse_test_lasso = mean_squared_error(y_test, y_pred_test_lasso)

#Mean Absolute Error for train and test sets

mae_train_lasso = mean_absolute_error(y_train,y_pred_train_lasso)
mae_test_lasso = mean_absolute_error(y_test,y_pred_test_lasso)
```

```
#RMSE(Root Mean Squared Error) for train and test sets

RMSE_train_lasso = np.sqrt(mse_train_lasso)
RMSE_test_lasso = np.sqrt(mse_test_lasso)

print("\n For Train Set: \n R2 Score : ",r2_score_train_lasso,"\nMSE Score : ",mse_train_lasso,"\nMAE Score : ",mae_train_lasso,
      "\nRMSE Score : ",RMSE_train_lasso)
print("\n For Test Set: \n R2 Score : ",r2_score_test_lasso,"\nMSE Score : ",mse_test_lasso,"\nMAE Score : ",mae_test_lasso,
      "\nRMSE Score : ",RMSE_test_lasso)
```

```
For Lasso Regression Model :

 For Train Set:
 R2 Score :  0.928136841490594
MSE Score :  0.0718631585094061
MAE Score :  0.18736078380717713
RMSE Score :  0.2680730469655726

 For Test Set:
 R2 Score :  0.8572791940265329
MSE Score :  0.12251874014630047
MAE Score :  0.22037256478136671
RMSE Score :  0.35002677061376386
```

There is a slight increase in R2 Score

	Features	Coefficients	Abs_Coefficient_Lasso
0	SaleType_New	0.51	0.51
1	SaleCondition_Partial	-0.43	0.43
2	GrLivArea	0.38	0.38
3	KitchenQual_TA	-0.20	0.20
4	KitchenQual_Gd	-0.18	0.18
5	MSZoning_RL	0.17	0.17
6	BsmtQual_TA	-0.16	0.16
7	BsmtQual_Gd	-0.15	0.15
8	OverallQual	0.15	0.15
9	PropertyAge	-0.12	0.12
10	MSZoning_FV	0.12	0.12
11	OverallCond	0.11	0.11
12	MSZoning_RM	0.11	0.11
13	ExterQual_TA	-0.10	0.10
14	TotalBsmtSF	0.09	0.09

Features are same but there is a change in coefficients.

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?

The R2 score for ridge is higher than the lasso, but I will choose lasso model because it assigns a zero value to insignificant features and thus it helps in predicting the model easily.

We always go for simple and robust model hence lasso is chosen.

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?

Code Below:

Dropped the first five variables and calculated the coefficients:

```
#question 3
X_train1 = X_train
y_train1  =y_train
```

```
X_test1  = X_test
y_test1 = y_test
```

```
X_train1  = X_train1.drop(['SaleType_New','SaleCondition_Partial','GrLivArea','KitchenQual_TA','KitchenQual_Gd'],axis=1)
```

```
X_test1  = X_test1.drop(['SaleType_New','SaleCondition_Partial','GrLivArea','KitchenQual_TA','KitchenQual_Gd'],axis=1)
```

```
lasso = Lasso(alpha=0.0001)

lasso.fit(X_train1, y_train1)
```

```
Lasso(alpha=0.0001)
```

```
#Lasso Regression Model Predicitons

print("For Lasso Regression Model :")
y_pred_train_lasso = lasso.predict(X_train1)
y_pred_test_lasso = lasso.predict(X_test1)

#R2 score for Lasso Regression Model

r2_score_train_lasso = r2_score(y_true=y_train1, y_pred=y_pred_train_lasso)
r2_score_test_lasso = r2_score(y_true=y_test1, y_pred=y_pred_test_lasso)

#MSE(Mean Squared Error) for Lasso Regression Model

mse_train_lasso = mean_squared_error(y_train1, y_pred_train_lasso)
mse_test_lasso = mean_squared_error(y_test1, y_pred_test_lasso)

#Mean Absolute Error for train and test sets

mae_train_lasso = mean_absolute_error(y_train1,y_pred_train_lasso)
mae_test_lasso = mean_absolute_error(y_test1,y_pred_test_lasso)
```

```
#RMSE(Root Mean Squared Error) for train and test sets

RMSE_train_lasso = np.sqrt(mse_train_lasso)
RMSE_test_lasso = np.sqrt(mse_test_lasso)

print("\n For Train Set: \n R2 Score : ",r2_score_train_lasso,"\nMSE Score : ",mse_train_lasso,"\nMAE Score : ",mae_train_lasso,
     "\nRMSE Score : ",RMSE_train_lasso)
print("\n For Test Set: \n R2 Score : ",r2_score_test_lasso,"\nMSE Score : ",mse_test_lasso,"\nMAE Score : ",mae_test_lasso,
     "\nRMSE Score : ",RMSE_test_lasso)
```

```
For Lasso Regression Model :

 For Train Set:
 R2 Score :  0.8767478499454741
MSE Score :  0.12325215005452599
MAE Score :  0.2521567725741239
RMSE Score :  0.3510728557643356

 For Test Set:
 R2 Score :  0.8194892513250096
MSE Score :  0.15495953347290387
MAE Score :  0.27721074950874797
RMSE Score :  0.39364899780502915
```

	Features	Coefficients	Abs_Coefficient_Lasso
0	OverallQual	0.31	0.31
1	BsmtQual_TA	-0.30	0.30
2	BsmtQual_Gd	-0.25	0.25
3	MSZoning_RL	0.22	0.22
4	ExterQual_TA	-0.21	0.21
5	MSZoning_RM	0.14	0.14
6	MSZoning_FV	0.14	0.14
7	GarageArea	0.13	0.13
8	ExterQual_Gd	-0.13	0.13
9	LotArea	0.11	0.11
10	BedroomAbvGr	0.11	0.11
11	TotalBsmtSF	0.11	0.11
12	BsmtQual_Fa	-0.09	0.09
13	OverallCond	0.09	0.09
14	Exterior1st_BrkFace	0.08	0.08

The next five important predictors are:

OverallQual, BsmtQual_TA, BsmtQual_Gd, MSZoning_RL and ExterQual_TA.

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?

The model should be generalized in such a way that, the model should behave well with unseen data not the training data. We should not remove entirely the outliers because this plays the major role in prediction in unseen data. If the model is not robust then it is very difficult to trust the model value in unseen data.