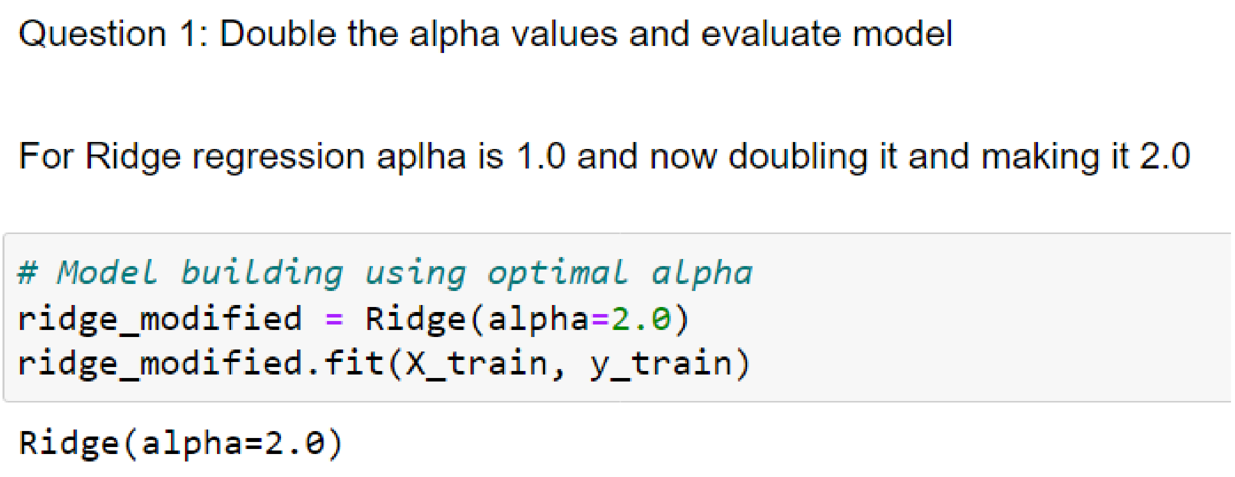
Subjective Questions

Note: Coding problem solutions are done in IPYNB file please refer to it

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:
Ridge regression - For ridge regression, the optimal value of alpha is 1.
Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity. When multicollinearity occurs, least squares estimates are unbiased, but their variances are large so they may be far from the true value.

Lasso Regression - In this case of Lasso regression, the optimal value for alpha is 0.0001.
Lasso regression is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters).

If we choose double the value of alpha for both ridge and lasso regression, model complexity will have a greater contribution to the cost. Because the minimum cost hypothesis is selected, this means that higher λ will bias the selection toward models with lower complexity.


Question 1: Double the alpha values and evaluate model

For Ridge regression aplha is 1.0 and now doubling it and making it 2.0

```
# Model building using optimal alpha
ridge_modified = Ridge(alpha=2.0)
ridge_modified.fit(X_train, y_train)
```

```
Ridge(alpha=2.0)
```

After second model is build we compare the r square value of new model with the old one.
The model which is having high r square of test and train dataset, we will select the features/variables from that model. And the variable is selected based on the high coefficient value.

```python
#creating coeffcients for the ridge regression
model_parameter = list(ridge.coef_)
model_parameter.insert(0,ridge.intercept_)
cols = house_train.columns
cols.insert(0,'const')
ridge_coef = pd.DataFrame(list(zip(cols,model_parameter,(abs(ele) for ele in model_parameter))))
ridge_coef.columns = ['Features','Coefficient','Mod']
#selecting the top 10 variables
ridge_coef.sort_values(by='Mod',ascending=False).head(10)
```

	Features	Coefficient	Mod
0	LotFrontage	10.261566	10.261566
3	OverallCond	0.527282	0.527282
14	BsmtUnfSF	0.406426	0.406426
12	BsmtFinType2	0.355562	0.355562
2	OverallQual	0.338037	0.338037
11	BsmtFinSF1	0.322512	0.322512
9	BsmtExposure	0.316265	0.316265
33	GarageFinish	0.303286	0.303286
73	LotConfig_CulDSac	-0.272453	0.272453
6	ExterCond	0.264079	0.264079

```python
y_train_pred = ridge_modified.predict(X_train)
y_test_pred = ridge_modified.predict(X_test)

print("Ridge Regression train r2:",r2_score(y_true=y_train,y_pred=y_train_pred))
print("Ridge Regression test r2:",r2_score(y_true=y_test,y_pred=y_test_pred))
```

```
Ridge Regression train r2: 0.9238505663513401
Ridge Regression test r2: 0.7526833276310241
```

For Lasso regression alpha is 0.0001 and not doubling it and making it 0.0002

```python
# Model building using optimal alpha
lasso_modified = Lasso(alpha=0.0002)
lasso_modified.fit(X_train, y_train)
```

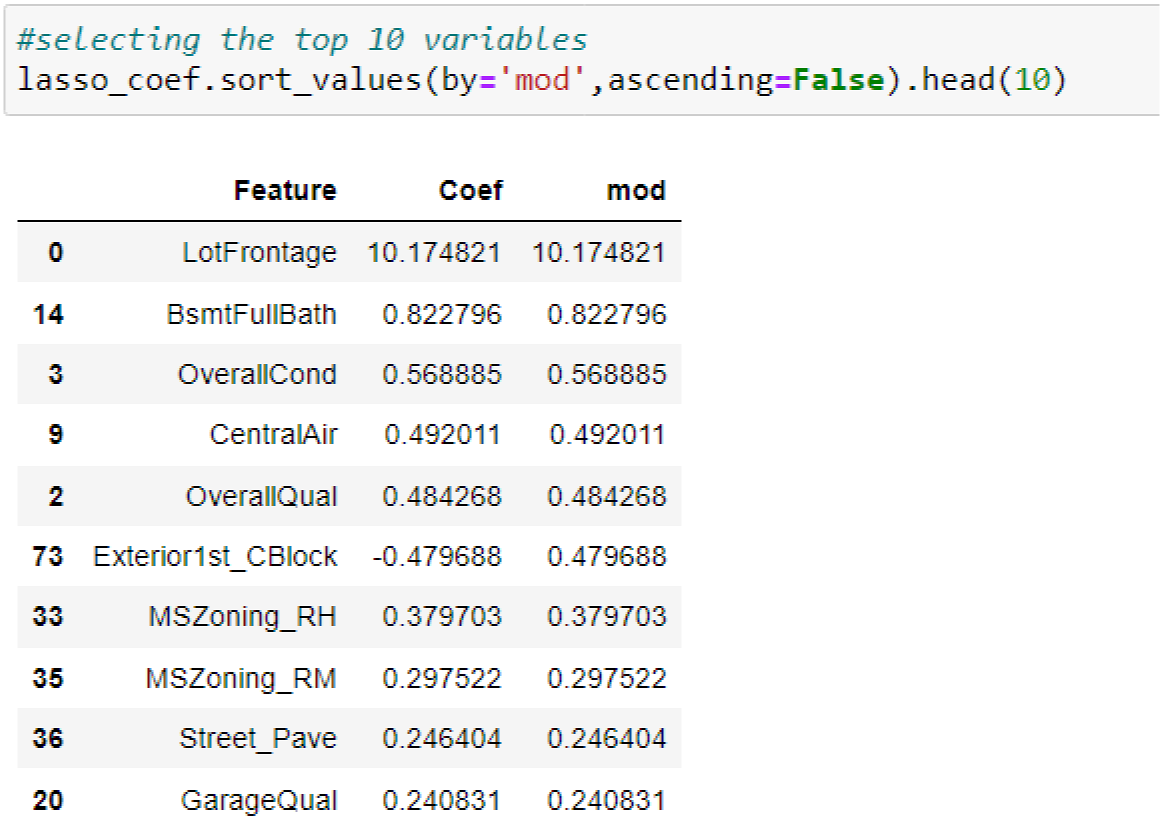
```
Lasso(alpha=0.0002)
```

```python
y_train_pred = lasso_modified.predict(X_train)
y_test_pred = lasso_modified.predict(X_test)

print("Lasso Regression train r2:",r2_score(y_true=y_train,y_pred=y_train_pred))
print("Lasso Regression test r2:",r2_score(y_true=y_test,y_pred=y_test_pred))
```

```
Lasso Regression train r2: 0.924842196868562
Lasso Regression test r2: 0.7423837845239192
```

```python
model_param = list(lasso.coef_)
model_param.insert(0,lasso.intercept_)
cols = X_train.columns
cols.insert(0,'const')
lasso_coef = pd.DataFrame(list(zip(cols,model_param,(abs(ele) for ele in model_param))))
lasso_coef.columns = ['Feature','Coef','mod']
```



```
#selecting the top 10 variables
lasso_coef.sort_values(by='mod',ascending=False).head(10)
```

	Feature	Coef	mod
0	LotFrontage	10.174821	10.174821
14	BsmtFullBath	0.822796	0.822796
3	OverallCond	0.568885	0.568885
9	CentralAir	0.492011	0.492011
2	OverallQual	0.484268	0.484268
73	Exterior1st_CBlock	-0.479688	0.479688
33	MSZoning_RH	0.379703	0.379703
35	MSZoning_RM	0.297522	0.297522
36	Street_Pave	0.246404	0.246404
20	GarageQual	0.240831	0.240831

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:
Lasso regression would be a better option it would help in feature elimination and the model will be more robust. Because

- In the ridge, the coefficients of the linear transformation are normal distributed and in the lasso they are Laplace distributed. In the lasso, this makes it easier for the coefficients to be zero and therefore easier to eliminate some of your input variable as not contributing to the output.
- Ridge regression can't zero out coefficients; thus, you either end up including all the coefficients in the model, or none of them. In contrast, the LASSO does both parameter shrinkage and variable selection automatically.
- Lasso regression can produce many solutions to the same problem.
- Ridge regression can only produce one solution to one problem.
- If I'm interested in identifying the most important predictors in the data, lasso regression I'll be a good choice. If I'm more concerned about reducing the variance of the model, then ridge regression might be a better option.
- If I have highly correlated predictors, lasso regression might be more effective at selecting a single predictor from each group of correlated predictors.

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:
Statistical measures can show the relative importance of the different predictor variables. However, these measures can't determine whether the variables are important in a practical sense. To determine practical importance, you'll need to use your subject area knowledge.

How you collect and measure your sample can bias the apparent importance of the variables in your sample compared to their true importance in the population.

If you randomly sample your observations, the variability of the predictor values in your sample likely reflects the variability in the population. In this case, the standardized coefficients and the change in R-squared values are likely to reflect their population values.

However, if you select a restricted range of predictor values for your sample, both statistics tend to underestimate the importance of that predictor. Conversely, if the sample variability for a predictor is greater than the variability in the population, the statistics tend to overestimate the importance of that predictor.

Also, consider the accuracy and precision of the measurements for your predictors because this can affect their apparent importance. For example, lower-quality measurements can cause a variable to appear less predictive than it truly is.

How you define “most important” often depends on your goals and subject area. While statistics can help you identify the most important variables in a regression model, applying subject area expertise to all aspects of statistical analysis is crucial. Real world issues are likely to influence which variable you identify as the most important in a regression model.

For example, if your goal is to change predictor values in order to change the response, use your expertise to determine which variables are the most feasible to change. There may be variables that are harder, or more expensive, to change. Some variables may be impossible to change. Sometimes a large change in one variable may be more practical than a small change in another variable.

“Most important” is a subjective, context sensitive characteristic. You can use statistics to help identify candidates for the most important variable in a regression model, but you’ll likely need to use your subject area expertise as well.

Question 3: Double the alpha values and evaluate model

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?

Droping the first five important predictors

```
X_train_new = X_train.drop(['LotFrontage','BsmtFullBath','OverallCond','CentralAir','OverallQual'],axis=1)
X_test_new = X_test.drop(['LotFrontage','BsmtFullBath','OverallCond','CentralAir','OverallQual'],axis=1)

X_test_new.head()
X_train_new.shape
```

```
(988, 85)
```

```
X_test_new.shape
```

```
(424, 85)
```

```
lasso_modified = Lasso()
param = {'alpha': [0.0001, 0.001, 0.01]}
folds = 5
# cross validation
lasso_cv_model_modified = GridSearchCV(estimator = lasso,
                        param_grid = param,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)

lasso_cv_model_modified.fit(X_train_new, y_train)
```

```
Fitting 5 folds for each of 3 candidates, totalling 15 fits
```

```
GridSearchCV(cv=5, estimator=Lasso(alpha=0.0001),
             param_grid={'alpha': [0.0001, 0.001, 0.01]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

```
#Creating the results dataframe.
lasso_cv_modified_results = pd.DataFrame(lasso_cv_model_modified.cv_results_)
#reading the results
lasso_cv_modified_results.head()
```

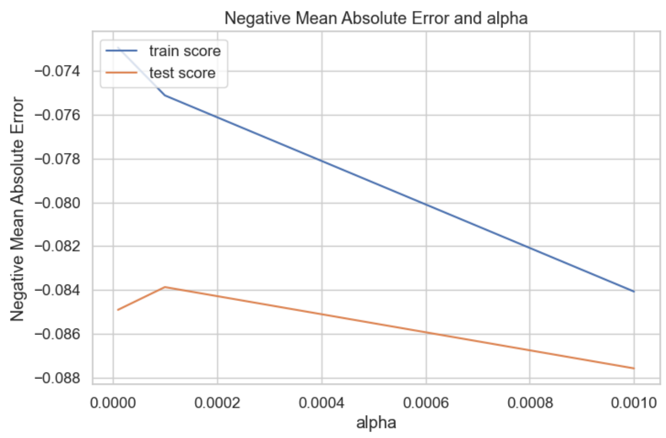
	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score
0	0.040644	0.007470	0.004079	0.004102	0.0001	{'alpha': 0.0001}	-0.100382	-0.100213	-0.091105	-0.084630
1	0.010143	0.004304	0.001067	0.002135	0.001	{'alpha': 0.001}	-0.105203	-0.108352	-0.099320	-0.084046
2	0.006657	0.003334	0.000000	0.000000	0.01	{'alpha': 0.01}	-0.145036	-0.147623	-0.138726	-0.121165

3 rows × 21 columns

```
# plotting mean test and train scores with alpha
cv_results['param_alpha'] = cv_results['param_alpha'].astype('float32')

# plotting
plt.plot(cv_results['param_alpha'], cv_results['mean_train_score'])
plt.plot(cv_results['param_alpha'], cv_results['mean_test_score'])
plt.xlabel('alpha')
plt.ylabel('Negative Mean Absolute Error')

plt.title("Negative Mean Absolute Error and alpha")
plt.legend(['train score', 'test score'], loc='upper left')
plt.show()
```



```
# Checking the best parameter(Alpha value)
model_cv.best_params_
```

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

```
# After performing grid search we found the same alpha that ue use before
lasso = Lasso(alpha=0.0001)
lasso.fit(X_train_new,y_train)

y_train_pred = lasso.predict(X_train_new)
y_test_pred = lasso.predict(X_test_new)

print("Lasso Regression train r2:",r2_score(y_true=y_train,y_pred=y_train_pred))
print("Lasso Regression test r2:",r2_score(y_true=y_test,y_pred=y_test_pred))
```

```
Lasso Regression train r2: 0.9124746336018961
Lasso Regression test r2: 0.7032947950551858
```

```
model_param = list(lasso.coef_)
model_param.insert(0,lasso.intercept_)
cols = X_train_new.columns
cols.insert(0,'const')
lasso_coef = pd.DataFrame(list(zip(cols,model_param,(abs(ele) for ele in model_param))))
lasso_coef.columns = ['Feature','Coef','mod']
```

```
#selecting the top 5 variables
lasso_coef.sort_values(by='mod',ascending=False).head(5)
```

	Feature	Coef	mod
0	LotArea	10.205140	10.205140
10	FullBath	1.029837	1.029837
6	1stFlrSF	0.606857	0.606857
1	ExterCond	0.561906	0.561906
28	MSZoning_RH	0.512930	0.512930

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:
A model needs to be made robust and generalizable so that they are not impacted by outliers in the training data. The model should also be generalisable so that the test accuracy is not lesser than the training score. The model should be accurate for datasets other than the ones which were used during training. Too much weightage should not given to the outliers so that the accuracy predicted by the model is high. To ensure that this is not the case, the outlier analysis needs to be done and only those which are relevant to the dataset need to be retained. Those outliers which it does not make sense to keep must be removed from the dataset. This would help increase the accuracy of the predictions made by the model. Confidence intervals can be used. This would help standardize the predictions made by the model. If the model is not robust, it cannot be trusted for predictive analysis.

The best accuracy is 100% indicating that all the predictions are correct. For an imbalanced dataset, accuracy is not a valid measure of model performance. For a dataset where the default rate is 5%, even if all the records are predicted as 0, the model will still have an accuracy of 95%