Assignment-based 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 to double the value of alpha for both ridge and lasso? What will be the most important predictor variables after the change is implemented?

Answer:

The optimal lambda value in the case of Ridge and Lasso is as below:

- Ridge: 3
- Lasso: 0.0002

Impact of doubling the value of alpha:

In ridge regression and Lasso regression, alpha is a hyperparameter that controls the strength of the regularization. By doubling the value of alpha in both Ridge and Lasso, the model will undergo the following changes:

- Ridge Regression: Ridge regression adds a penalty term to the ordinary least squares (OLS) cost function to avoid overfitting. By doubling the value of alpha, the penalty term will become stronger, and the model will become more regularized. This will cause the coefficients of the model to shrink more, reducing the variance of the model. The larger value of alpha will result in a simpler model that is less likely to overfit the data.
- Lasso Regression: Lasso regression also adds a penalty term to the OLS cost function to avoid overfitting. However, unlike ridge regression, Lasso uses an L1 penalty that can lead to some coefficients being reduced to zero, effectively performing feature selection. By doubling the value of alpha, the penalty term will become stronger, and the model will become more regularized, resulting in a simpler model with fewer features. The larger value of alpha will make Lasso regression even more effective at feature selection, and the coefficients will shrink more than in the case of Ridge regression.

In summary, doubling the value of alpha in both Ridge and Lasso regression will result in more regularized models that are simpler and less likely to overfit the data. Ridge regression will shrink the coefficients more, reducing the variance of the model, while Lasso regression will be more effective at feature selection, reducing the number of features used in the model.

Doubling the coefficient has reduced the number of features in the model from 74 to 63 in lasso regression

```
# Checking no. of features in Ridge and Lasso models
selected_features= len(lasso_coef2[lasso_coef2 != 0])
print('Features selected by Lasso:', selected_features)
print('Features present in Ridge:', X_train.shape[1])
```

```
Features selected by Lasso: 63
Features present in Ridge: 100
```

Top 10 features after doubling the value of alpha in ridge regression:

We will take the value of alpha as 6, which will add more penalty on the curve, and try to make the model more generalized, i.e. model will become simpler and it wouldn't fit every data point on the dataset

Top features with an alpha value of 6:

```
PoolQC_Gd              -0.198364
Condition2_PosN        -0.168450
Neighborhood_StoneBr    0.130375
Neighborhood_NridgHt    0.126800
BldgType_Twnhs         -0.113866
MSSubClass_160         -0.110881
Neighborhood_Crawfor    0.109714
CentralAir              0.106909
OverallQual             0.103366
LandSlope_Sev           0.092161
dtype: float64
```

Top 10 features after doubling the value of alpha in lasso regression:

We will take the value of alpha as 0.0004, which will add more penalty to the model, and more coefficients of the model will be reduced to zero.

Top features with an alpha value of 0.0004:

```
PoolQC_Gd              -0.783320
Condition2_PosN        -0.538341
Exterior1st_BrkComm    -0.170609
Neighborhood_StoneBr    0.147778
Neighborhood_NridgHt    0.136416
GrLivArea               0.128672
BldgType_Twnhs         -0.126294
Functional_Maj2        -0.116969
Neighborhood_Crawfor    0.115009
CentralAir              0.113077
dtype: float64
```

Most of the features are coming similar in both methods but there is a wide difference in the 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?

Answer:

Our target is to deliver a model which is simple yet robust, also Occam's Razor explains that the Model should not be unnecessarily complex by adding unnecessary assumptions.

We have identified the optimal value for alpha for both ridge and lasso as follows

- Ridge: 3
- Lasso: 0.0002

The below table represents different parameters obtained using these alphas on their respective models:

	Metric	Linear Regression	Ridge Regression	Lasso Regression
0	R2 Score (Train)	0.949954	0.891930	0.900328
1	R2 Score (Test)	0.854708	0.880796	0.872677
2	RSS (Train)	7.927972	17.119840	15.789525
3	RSS (Test)	10.796347	8.857830	9.461135
4	MSE (Train)	0.088119	0.129490	0.124357
5	MSE (Test)	0.156822	0.142047	0.146804

Also, the number of features selected by different methods after RFE are:

- Ridge: 100
- Lasso: 74

```
In [58]: # Checking no. of features in Ridge and Lasso models
lasso_coef= pd.Series(lasso.coef_, index= X_train.columns)
selected_features= len(lasso_coef[lasso_coef != 0])
print('Features selected by Lasso:', selected_features)
print('Features present in Ridge:', X_train.shape[1])

Features selected by Lasso: 74
Features present in Ridge: 100
```

Although Ridge has less deviation in the R2 score of the train and test dataset compared to Lasso, we can't neglect the fact that Lasso has also performed really well on both datasets.

Both the models have performed almost similarly in terms of mean squared error, but Lasso has reduced the features from 100 to 74.

By considering the above observations, both models have similar performances on train and test datasets. We should choose the simpler model, so we will prefer Lasso as our final Model.

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:

We have chosen the Lasso model as our final model, so starting by getting the current top 5 Features:

```
RoofMatl_Tar&Grv        0.217230
Neighborhood_StoneBr    0.158988
Neighborhood_NridgHt    0.142995
RoofMatl_WdShngl        0.141495
GrLivArea               0.129408
```

As Neighborhood_StoneBr, Neighborhood_NridgHt, RoofMatl_WdShngl, and RoofMatl_Tar&Grv are dummy variables, we'll drop entire Neighborhood and RoofMatl features.

After dropping Neighborhood, RoofMatl, and GrLivArea, we rebuilt the model again with original alpha value of 0.0002.

Now 5 most important predictor variables have been changed and they are a follows:

```
PoolQC_Gd             -0.978240
Condition2_PosN       -0.715990
Functional_Sev        -0.272704
Exterior1st_BrkComm   -0.260420
Heating_Grav          -0.172727
dtype: float64
```

- PoolQC_Gd Good Pool Quality
- Condition2_PosN Condition - Near positive off-site feature--park, greenbelt, etc
- Functional_Sev Home Functionality
- Exterior1st_BrkComm Exterior covering on house with Brick common
- Heating_Grav Gravity furnace Heating

Question 4. How can you make sure that a model is robust and generalizable? What are the implications of the same for the accuracy of the model and why?

Answer:

Ensuring that a model is robust and generalizable is crucial for its performance and practical use.

- Robustness is the model's ability to handle unexpected data inputs, noise, and perturbations without significantly affecting its performance.
- Generalizability is the model's ability to perform well on new, unseen data that was not used during training.

Here are some ways to ensure that a model is robust and generalizable:

- Train on diverse data: A model trained on a diverse range of data is more likely to be robust and generalizable. The training data should include examples from various sources, domains, and contexts to improve the model's ability to generalize.

- Use cross-validation: Cross-validation is a technique that can help determine a model's ability to generalize. Cross-validation involves partitioning the data into

multiple subsets and training the model on one subset while evaluating its performance on the other subsets. This technique helps to identify whether the model is overfitting to the training data or generalizing well to new data.

- Regularization: Regularization is a technique that helps to prevent overfitting by adding a penalty term to the loss function during training. This technique helps the model to generalize better by reducing the impact of noisy or irrelevant features.

- Testing on unseen data: The ultimate test of a model's generalizability is its performance on unseen data. It is essential to evaluate the model's performance on a separate dataset that was not used during training.

The implications of having a robust and generalizable model for its accuracy are significant. A robust and generalizable model is less likely to overfit the training data, which can lead to poor performance on new data. Such a model can perform well even when faced with new and unexpected data, leading to better accuracy and a more reliable model for practical use. Additionally, a model that is robust and generalizable can help to identify patterns and relationships in the data, leading to more accurate predictions and better insights.

Also, the model should be as simple as possible, it could cause its accuracy to drop but it will be more robust and generalizable. It can also be understood using the Bias-Variance trade-off. Its implication in terms of accuracy is that model will perform equally well on both training and test data.