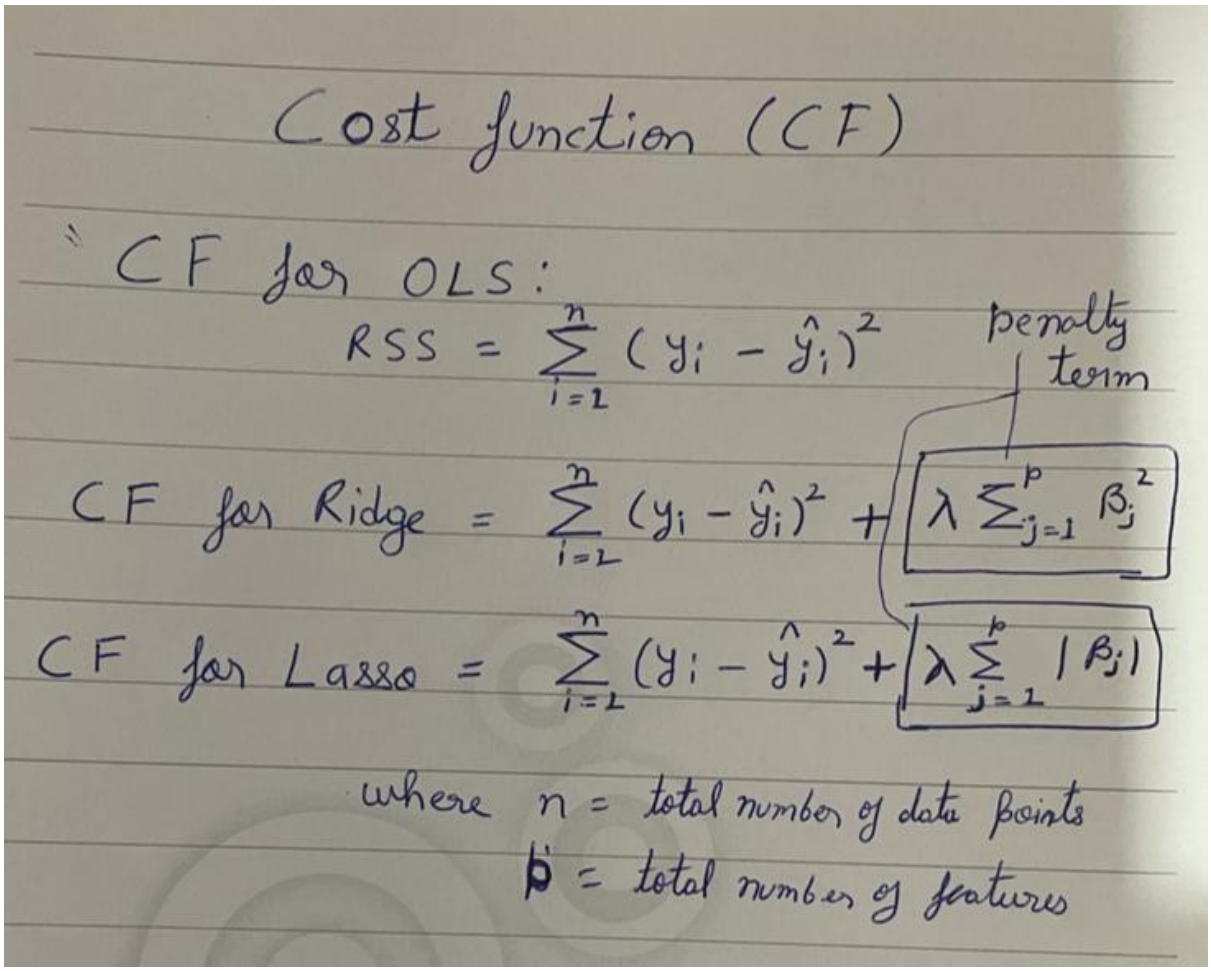
Question 1

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 – After building the model by using Ridge and Lasso Regression separately, I found that that out of the 28 values of lambda we tested with for Ridge, lambda = 5 was the optimum value and for Lasso, lambda = .0001 was found to be optimum.

Theoretically, doubling the value of lambda means higher regularization and if we keep on increasing it, we will reach a state of underfitting where our model will be so simple that it would not learn about any relationship in data.

To understand this better we need to consider the cost function for Ridge and Lasso which are very similar and the difference is only in the penalty term of cost function.



If we focus on the penalty term, it is multiplication of lambda with coefficient value square for Ridge and modulus of coefficient value for Lasso. Now if lambda is very large number, then the other term having coefficient will have to be minimum in order to keep the whole cost function minimum. As we can imagine, higher the value of lambda, it will push the value of coefficient towards zero. And when most of the coefficients will be close to 0 (or 0 in case of Lasso) our model will become very simple with a low accuracy on the training data itself.

This is called underfitting. Therefore, we need to find optimum value of lambda by hit and trial

As it was required in the question specifically, I used the double values of lambda for both Ridge and Lasso and as expected R-Square score dropped slightly. Also, Lasso eliminated more features. With lambda = .0001, Lasso selected 119 features and with lambda= 0.0002 Lasso selected 86 features only. Here are the values as calculated in code.

Before –

	Metric	Linear Regression	Ridge Regression	Lasso Regression
0	R2 Score (Train)	9.472827e-01	0.892408	0.904542
1	R2 Score (Test)	-4.134031e+20	0.863825	0.857475
2	RSS (Train)	6.488059e-01	1.324164	1.174823
3	RSS (Test)	2.252776e+21	0.742066	0.776667
4	MSE (Train)	2.520836e-02	0.036013	0.033921
5	MSE (Test)	2.265305e+09	0.041114	0.042062

After -

```
 r2 on train set after doubling lambda for Ridge
0.8803333691040196
r2 on test set after doubling lambda for Ridge
0.8575229794032266
```

```
r2 on train set after doubling lambda for Lasso
0.8911979763993433
r2 on test set after doubling lambda for Lasso
0.8608921127366799
```

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 – For selecting one out of Ridge and Lasso, Feature selection is a major criterion. If we have less features in our final model it makes interpretation of model easier and the storytelling for business is also a lot better. Since our data set had 237 features when we built the model, the inherent inclination was to choose Lasso, unless, it has a significantly lesser accuracy than Ridge.

With this thought process I went ahead and applied both Ridge and Lasso to compare the results. What I found after running Ridge and Lasso with optimum values of lambda is Lasso is indeed giving almost as good accuracy – 85.74 % as Ridge – 86.38 % on the test set.

So, I am going to choose Lasso as it is just selecting 119 out of 227 features.

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- This question is 100% practical. I had to build the model again after dropping the top 5 features. These are the top five most important predictor variables in Lasso model built the first time -

```
lasso_scores.sort_values(by="Coeff", ascending = False)
```

	Feature	Coeff
14	GrLivArea	0.338970
106	RoofMatl_WdShngl	0.123036
3	OverallQual	0.120857
60	Neighborhood_NoRidge	0.065602
21	GarageCars	0.053796
...	...	...
0	MSSubClass	-0.035662
152	BsmtQual_Gd	-0.037945
188	KitchenQual_Gd	-0.039320
189	KitchenQual_TA	-0.039331
81	Condition2_PosN	-0.318455

119 rows × 2 columns

Now I removed the top five features from the train and test data and rebuilt the Lasso model.

Here are the results after sorting the features by coefficients along with the code -

```
# recalculating the next top 5 features from Lasso

lasso_scores = pd.DataFrame(
{
    "Feature": X_train.columns,
    "Coeff": lasso.coef_
})

lasso_scores = lasso_scores[lasso_scores["Coeff"]!=0]

lasso_scores.sort_values(by="Coeff", ascending = False)
```

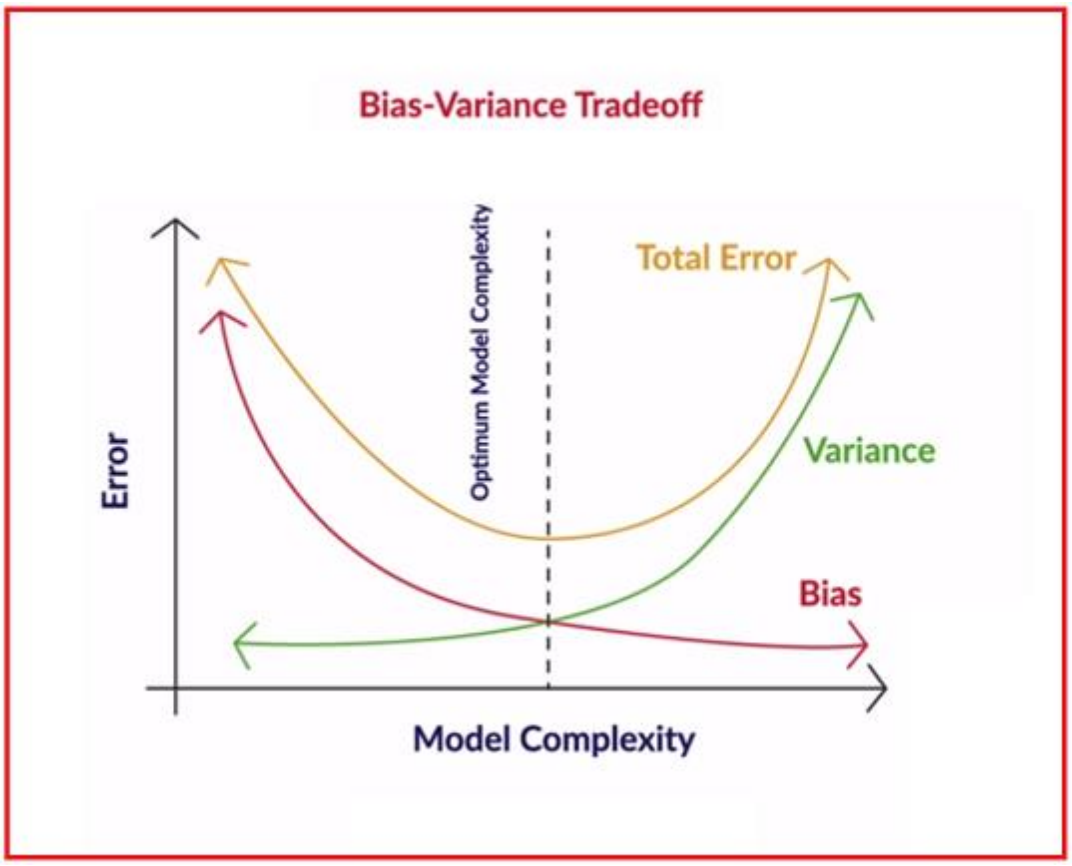
	Feature	Coeff
11	1stFlrSF	0.331983
12	2ndFlrSF	0.196507
6	MasVnrArea	0.059019
2	LotArea	0.055674
19	GarageArea	0.049545
...	...	...
147	BsmtQual_Gd	-0.041507
148	BsmtQual_TA	-0.041929
183	KitchenQual_Gd	-0.044259
184	KitchenQual_TA	-0.045322
77	Condition2_PosN	-0.345255

114 rows × 2 columns

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 – For achieving a robust and generalisable model we need to keep in mind the bias variance trade-off and Occam's razor principle. The model should be as simple as possible without losing any significant accuracy.



Along with this we have to follow one more important rule of thumb – Test your model on the data it has never seen during the training. It will give a sense of accuracy when the model will be used in real world data set.

Now if we think about Bias-Variance trade-off used in Regularization technique, we are basically trying to compromise with a little bias to gain a significant reduction in variance. It means if we move towards a simple model by tuning the beta coefficients, the model's bias (or errors on train data) is increased but the variance (how the model is prone to change in train data) is decreased. We want to find the sweet spot where the gain on error is very less but reduction in variance is significantly larger.

If we want our model to be robust and generalisable, we need to take care of the problem of overfitting i.e model has high accuracy on train data but low accuracy on test data.

The above discussion about bias-variance trade-off is the key in solving overfitting. If our coefficients are under control (using Ridge or Lasso Regression), these coefficients would have small values and the model would not change suddenly with change in input data (low variance), but at the same time our bias also would be in acceptable limit.