Subjective Questions

Question 1	
1.a	What is the optimal value of alpha for ridge and lasso regression?
	The optimal value of alpha for ridge: 100 The optimal value of alpha for lasso: 0.0001
1.b	What will be the changes in the model if you choose to double the value of alpha for both ridge and lasso?
	Please find the changes in the model with alpha doubled for both Ridge and Lasso Models (Jupyter notebook (House_Price_Rathnagiri.ipynb) in repo has the code workings):

Metric	Ridge Model		Lasso Model	
	alpha = 100	alpha = 200	alpha = 0.0001	alpha = 0.0002
R2 Score (Train)	0.853777	0.851074	0.855106	0.855074
R2 Score (Test)	0.865862	0.864005	0.866119	0.866514
RSS (Train)	3.678083	3.746074	3.644650	3.645461
RSS (Test)	1.436583	1.456478	1.433840	1.429606
MSE (Train)	0.003624	0.003691	0.003591	0.003592
MSE (Test)	0.003295	0.003341	0.003289	0.003279

We can see that in the case of Ridge model, when the alpha is doubled the R2 score for training data and test data has slightly dropped to 0.851074 and 0.864005 respectively. Similarly, we can see a very slight impact regarding RSS and MSE as well.
While in the case of Lasso Model, we can see that R2 score for Training data has dropped slightly to 0.855074. But surprisingly, we see a slight performance improvement for Test data to 0.866514. Similar impact can be noted regarding the RSS and MSE as well for training and test datasets.
We still pick lasso model, as it provides slightly better performance metrics.

1.c	What will be the most important predictor variables after the change is implemented?

Following are the most important predictor variables after the change is implemented: OverallQual, TotRmsAbvGrd, GarageArea, YearBuilt, OverallCond, FullBath, Fireplaces, TotalBsmtSF, BsmtFinSF1, HalfBath.

Note that the coefficient values have slightly changed:

Rank	Feature Variable	Lasso (alpha=0.0001)	Lasso (alpha=0.0002)
1	OverallQual	0.048354	0.048504
2	TotRmsAbvGrd	0.020651	0.020613
3	GarageArea	0.019954	0.019999
4	YearBuilt	0.020047	0.019832
5	OverallCond	0.019813	0.019657

6	FullBath	0.01593	0.015879

Note that with the previous best alpha model of 0.0001, YearBuilt was at 3^{rd} and GarageArea was $4^{th,}$ but YearBuilt dropped to 4^{th} and GarageArea moved up to 3^{rd} when alpha is doubled.

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 optimal value of alpha for ridge: 100
The optimal value of alpha for lasso: 0.0001

The R2 Score for Lasso model is 0.855106 (training) and 0.866119 (Test data). So, we get slightly better performance with the test data, Therefore I would choose lasso model.

Metric	Ridge (alpha 100)	Lasso (alpha 0.0001)
R2 Score (Train)	0.853777	0.855106
R2 Score (Test)	0.865862	0.866119
RSS (Train)	3.678083	3.644650
RSS (Test)	1.436583	1.433840
MSE (Train)	0.003624	0.003591
MSE (Test)	0.003295	0.003289

We can see that with regard to RSS and Mean Squared Error as well, Lasso performs slightly better than Ridge with respect to unseen data (Test dataset) reflecting better generalizability and robustness.

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?

After rebuilding the lasso model by dropping the top 5 columns identified earlier, we get the following columns and the corresponding beta coefficients: (p.s notbook for corresponding code)

index	Lasso (alpha=0.001)
FullBath	0.04294206657264785
TotalBsmtSF	0.036016713767477886
HalfBath	0.026299045924251436
Fireplaces	0.022684186266524378
CentralAir	0.019398210773347076
Foundation	0.01624252719789734
SaleCondition	0.014728959774826817
PavedDrive	0.013354319712113401
WoodDeckSF	0.013101027909770354

index	Lasso (alpha=0.001)
BsmtFinSF1	0.011451886985815205
LotArea	0.00935338519429179
ScreenPorch	0.00934376284521863
EnclosedPorch	0.006203733101368491
Exterior2nd	0.00367397537915161
BsmtFinSF2	0.0004701750346812123
Exterior1st	-0.0010001089136281085
PoolArea	-0.0058891932394795135
BldgType	-0.011586743189386238
HeatingQC	-0.012327813964385993
KitchenQual	-0.028152648217613815

From above we can note that the top 5 are the following:

1	**FullBath**
2	**TotalBsmtSF**
3	**HalfBath**
4	**Fireplaces**
5	**CentralAir**

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?

It is important to ensure a model is robust and generalizable as its crucial for running under real-world conditions. Regarding robustness, it means a model should perform well even with slight variations in the data it encounters. We used techniques like data cleaning and regularization (to reduce model complexity). It is also important to look at the R – Squared value. But, R-squared itself doesn't directly ensure the robustness of a model. R-Squared measures the proportion of variance in the dependent variable explained by the independent variables in the model. A high R-Squared can indicate that the model is capturing the patterns in the training data well. However, it can also be a sign of overfitting. Therefore, to ensure robustness, techniques like using a validation set, applying regularization (reducing model complexity) and handling outliers can help improve model robustness without relying on R-Squared. Our model produced the following metrics:

Best Alpha: 0.0001
Training R-squared: 0.8551
Testing R-squared: 0.8661
Training Residual Sum of Squares (RSS): 3.6446
Testing Residual Sum of Squares (RSS): 1.4338
Training Mean Squared Error: 0.0036
Testing Mean Squared Error: 0.0033

We can see that our model based on lasso performs better on testing data (Testing R-Squared of 86% vs Training R-Squared of 85.5%). We can also note that the Residual Sum of Squares and Mean Squared

Error on Testing data is performing significantly better than Training data set. Overall, the goal is to strike a balance between accuracy on the training data and generalizability to unseen data. Robustness and generalizability techniques ensure our model is reliable in real-world scenarios with variations in the data, even if it sacrifices some training accuracy.