

SUBJECTIVE ANSWER – ADVANCED REGRESSION ASSIGNMENT

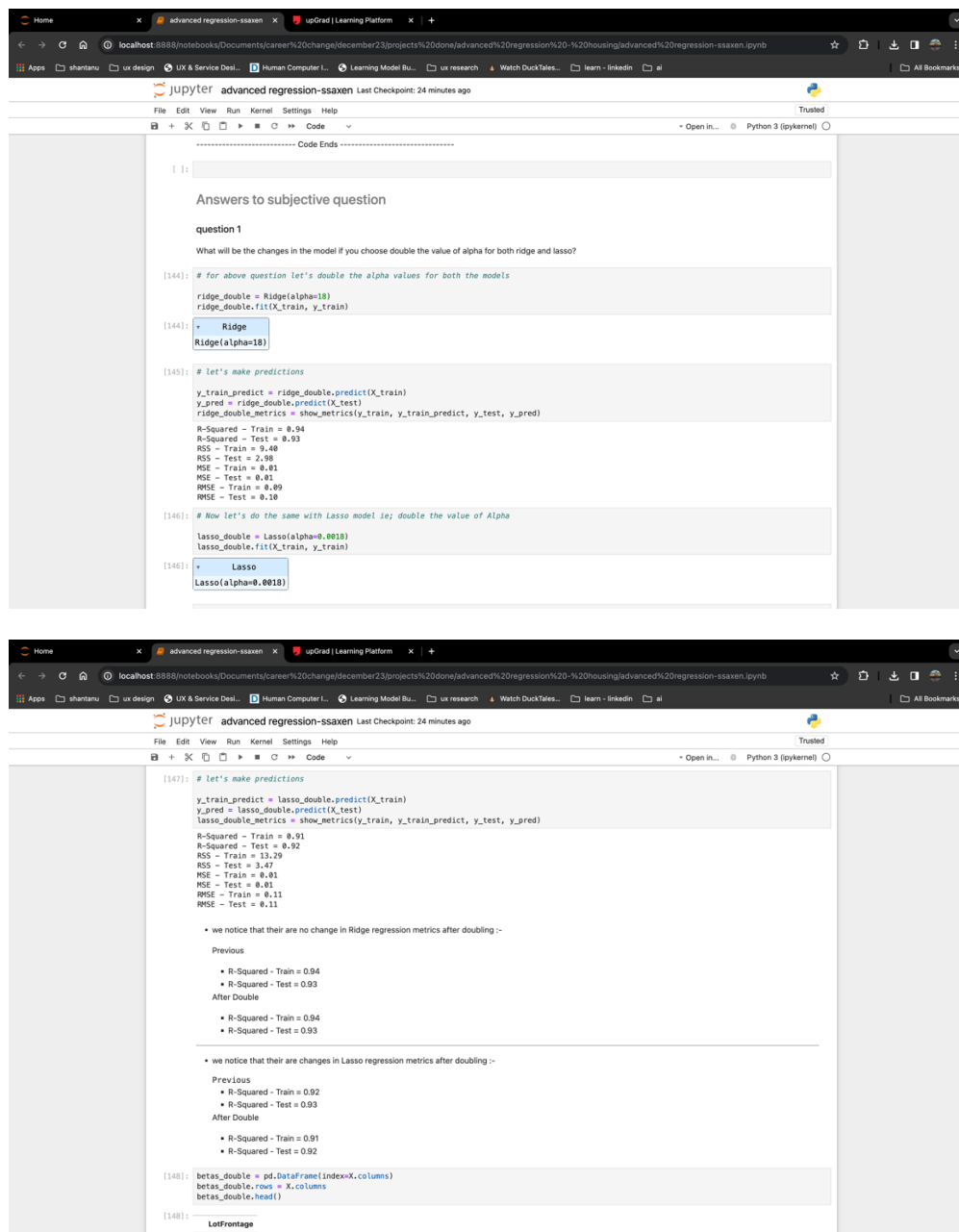
Question 1

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?

Ans :

The optimal value of lambda for Ridge Regression is 9. The optimal value of lambda for Lasso is 0.001.

To find the changes in model where we double the value of Alpha for both models we need to build is in Jupyter and see. [Attaching the screenshot below of the Jupyter file coded].



The first screenshot shows the initial setup of the Jupyter Notebook. It includes a title bar with browser tabs for 'advanced regression-ssaxen' and 'upGrad | Learning Platform'. The notebook interface shows a code cell with the following content:

```
[144]: # for above question let's double the alpha values for both the models
ridge_double = Ridge(alpha=18)
ridge_double.fit(X_train, y_train)

[144]: Ridge
Ridge(alpha=18)

[145]: # let's make predictions
y_train_predict = ridge_double.predict(X_train)
y_pred = ridge_double.predict(X_test)
ridge_double_metrics = show_metrics(y_train, y_train_predict, y_test, y_pred)

R-Squared - Train = 0.94
R-Squared - Test = 0.93
RSS - Train = 9.40
RSS - Test = 2.98
MSE - Train = 0.01
MSE - Test = 0.01
RMSE - Train = 0.09
RMSE - Test = 0.10

[146]: # Now let's do the same with Lasso model i.e; double the value of Alpha
lasso_double = Lasso(alpha=0.0018)
lasso_double.fit(X_train, y_train)

[146]: Lasso
Lasso(alpha=0.0018)
```

The second screenshot shows the continuation of the notebook. It includes a code cell with the following content:

```
[147]: # let's make predictions
y_train_predict = lasso_double.predict(X_train)
y_pred = lasso_double.predict(X_test)
lasso_double_metrics = show_metrics(y_train, y_train_predict, y_test, y_pred)

R-Squared - Train = 0.91
R-Squared - Test = 0.92
RSS - Train = 13.39
RSS - Test = 3.47
MSE - Train = 0.01
MSE - Test = 0.01
RMSE - Train = 0.11
RMSE - Test = 0.11
```

Below the code cell, there is a text block summarizing the changes in metrics:

```
• we notice that their are no change in Ridge regression metrics after doubling :-

Previous
• R-Squared - Train = 0.94
• R-Squared - Test = 0.93

After Double
• R-Squared - Train = 0.94
• R-Squared - Test = 0.93

• we notice that their are changes in Lasso regression metrics after doubling :-

Previous
• R-Squared - Train = 0.92
• R-Squared - Test = 0.93

After Double
• R-Squared - Train = 0.91
• R-Squared - Test = 0.92
```

The notebook ends with a code cell for data manipulation:

```
[148]: betas_double = pd.DataFrame(index=X.columns)
betas_double.rows = X.columns
betas_double.head()

[148]:
```

SUBJECTIVE ANSWER – ADVANCED REGRESSION ASSIGNMENT

```
[148]: LotFrontage
       LotArea
       YearRemodAdd
       MasVnrArea
       BsmFinSF1

[149]: betas_double.shape
[149]: (384, 8)

[150]: betas_double['Ridge Model'] = ridge.coef_
       betas_double['Lasso Model'] = ridge.coef_

[151]: betas_double['Ridge Model'].sort_values(ascending=False)[:15]

[151]: GrLivArea      0.898282
       OverallQual_9 0.886177
       OverallCond_9 0.879326
       OverallQual_8 0.877876
       Neighborhood_Crawfor 0.875995
       SaleCondition_Alloca 0.874258
       Functional_Typ 0.869484
       Exterior1st_BrkFace 0.867818
       TotalBsmtSF 0.857943
       CentralAir_Y 0.854588
       Neighborhood_StoneBr 0.853163
       BsmCond_Gd 0.848494
       OverallCond_7 0.847235
       Neighborhood_Nridgitt 0.844293
       OverallCond_8 0.843978
       Name: Ridge Model, dtype: float64

[152]: betas_double['Lasso Model'].sort_values(ascending=False)[:15]

[152]: GrLivArea      0.898282
       OverallQual_9 0.886177
       OverallCond_9 0.879326
       OverallQual_8 0.877876
       Neighborhood_Crawfor 0.875995
       SaleCondition_Alloca 0.874258
       Functional_Typ 0.869484
       Exterior1st_BrkFace 0.867818
       TotalBsmtSF 0.857943
       CentralAir_Y 0.854588
       Neighborhood_StoneBr 0.853163
       BsmCond_Gd 0.848494
       OverallCond_7 0.847235
       Neighborhood_Nridgitt 0.844293
       OverallCond_8 0.843978
       Name: Lasso Model, dtype: float64

So variables are doubling the values of Alpha are :-
• GrLivArea
• OverallQual_9
• OverallCond_9
• OverallQual_8
• Neighborhood_Crawfor
• SaleCondition_Alloca
• Functional_Typ
• Exterior1st_BrkFace
• TotalBsmtSF
• CentralAir_Y

[ ]:

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?

[153]: # let's drop the top 5 variables. We can pick top 5 from the above steps and drop them from train and test
       variable_topfive = ['OverallQual_9', 'GrLivArea', 'OverallQual_8', 'Neighborhood_Crawfor', 'Exterior1st_BrkFace']
       X_train_dropped = X_train.drop(variable_topfive, axis=1)
```

The most important predictor variables after the change is implemented are (refer the attached jupyter file in git for this):-

- GrLivArea
- OverallQual_9
- OverallCond_9
- OverallQual_8
- Neighborhood_Crawfor
- SaleCondition_Alloca
- Functional_Typ
- Exterior1st_BrkFace
- TotalBsmtSF
- CentralAir_Y

SUBJECTIVE ANSWER – ADVANCED REGRESSION ASSIGNMENT

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?

Ans :

Considering the value of R-Squared - Test = 0.93 for Lasso and Regression both we can select based on the use case (refer screenshot in the above question or jupyter file). So let's say if our primary goal is to select features than we can opt for Lasso Regression model.

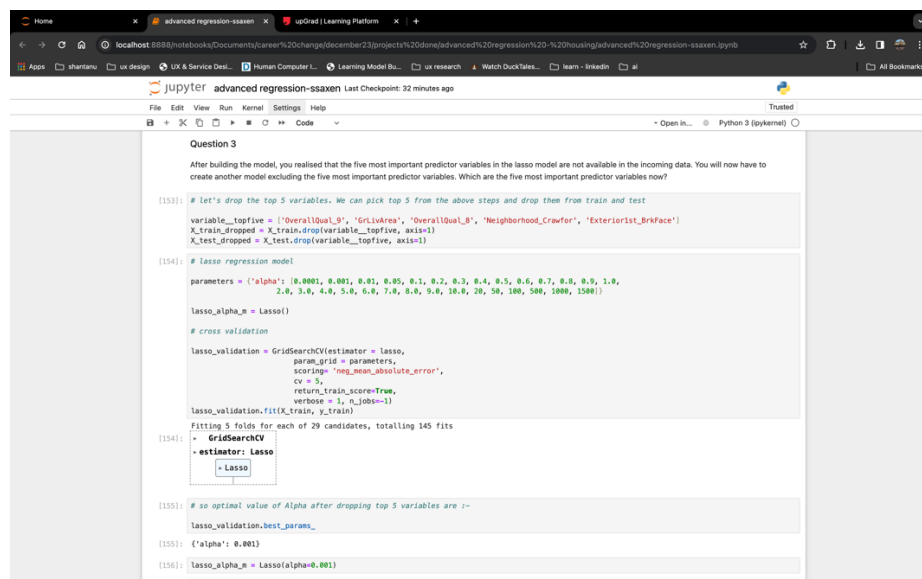
Question 3

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?

Ans:

After excluding top 5 we get these as models :- (refer below screenshots or the jupyter linked with assignment)

- 2ndFlrSF
- 1stFlrSF
- Functional_Typ
- Neighborhood_Somerst
- TotalBsmtSF



```
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?

[153]: # Let's drop the top 5 variables. We can pick top 5 from the above steps and drop them from train and test
variable_topfive = ['OverallQual_S', 'GrLivArea', 'OverallQual_B', 'Neighborhood_Crawfor', 'Exterior1st_BrkFace']
X_train_dropped = X_train.drop(variable_topfive, axis=1)
X_test_dropped = X_test.drop(variable_topfive, axis=1)

[154]: # Lasso regression model
parameters = {'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50, 100, 500, 1000, 1500]}

lasso_alpha_m = Lasso()

# cross validation
lasso_validation = GridSearchCV(estimator = lasso,
                                param_grid = parameters,
                                scoring = "neg_mean_absolute_error",
                                cv = 5,
                                return_train_score=True,
                                verbose = 1, n_jobs=-1)
lasso_validation.fit(X_train, y_train)

[154]: # GridSearchCV
- estimator: Lasso
  - Lasso

[155]: # so optimal value of Alpha after dropping top 5 variables are :-
lasso_validation.best_params_

[155]: {'alpha': 0.001}

[156]: lasso_alpha_m = Lasso(alpha=0.001)
```

SUBJECTIVE ANSWER – ADVANCED REGRESSION ASSIGNMENT

```
File Edit View Run Kernel Settings Help
JupyterLab advanced regression-ssaxen Last Checkpoint: 32 minutes ago
Python 3 (pykernel)

[156]: lasso_alpha_m = Lasso(alpha=0.001)
[157]: lasso_alpha_m.fit(X_train_dropped, y_train)
[158]: y_train_pred = lasso_alpha_m.predict(X_train_dropped)
[159]: y_pred = lasso_alpha_m.predict(X_test_dropped)
[160]: lasso_metrics = show_metrics(y_train, y_train_pred, y_test, y_pred)
[161]: lasso_metrics_table = {'Metrics': ['R2 Score (Train)', 'R2 Score (Test)', 'RSS (Train)', 'RSS (Test)',
'MSE (Train)', 'MSE (Test)', 'RMSE (Train)', 'RMSE (Test)'],
'Lasso Regression': lasso_metrics}
final_metric = pd.DataFrame(lasso_metrics_table, columns = ['Metrics', 'Lasso Regression'])
final_metric.set_index('Metrics')
```

Metrics	
R2 Score (Train)	0.913648
R2 Score (Test)	0.921713
RSS (Train)	12.826628
RSS (Test)	3.198525
MSE (Train)	0.010982
MSE (Test)	0.010982

```
File Edit View Run Kernel Settings Help
JupyterLab advanced regression-ssaxen Last Checkpoint: 32 minutes ago
Python 3 (pykernel)

[162]: # let's find the variable
empty_df = pd.DataFrame(index=X_train_dropped.columns)
empty_df['rws'] = X_train_dropped.columns
empty_df

[163]:
```

LotFrontage	
LotArea	
YearRemodAdd	
MasVnrArea	
BsmtFinSF1	
BsmtFinSF2	
BsmtUnfSF	
TotalBsmtSF	
1stFlrSF	
2ndFlrSF	

```
[164]: empty_df['Lasso Regression'] = lasso_alpha_m.coef_
[165]: empty_df.head()
```

Lasso Regression	
LotFrontage	0.002602
LotArea	0.019658
YearRemodAdd	0.026679
MasVnrArea	-0.000000
BsmtFinSF1	0.023302

```
File Edit View Run Kernel Settings Help
JupyterLab advanced regression-ssaxen Last Checkpoint: 32 minutes ago
Python 3 (pykernel)

[166]: empty_df['Lasso Regression'] = lasso_alpha_m.coef_
[167]: empty_df.head()
```

Lasso Regression	
LotFrontage	0.002602
LotArea	0.019658
YearRemodAdd	0.026679
MasVnrArea	-0.000000
BsmtFinSF1	0.023302

```
[168]: empty_df['Lasso Regression'].sort_values(ascending=False)[:5]
```

2ndFlrSF	0.098781
1stFlrSF	0.079284
Functional_Type	0.078766
Neighborhood_Somerst	0.068289
TotalBsmtSF	0.053112

```
[169]:
```

Subjective question answer ends -----

SHANTANU SAXENA

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?

Ans:

To make sure the model is robust and generalisable we can keep following in mind while making model :-

- By training the data on diverse dataset that covers wide range of scenarios and variations that can be faced later on.
- Cross verify the model performance on various data sets and check for consistency.
- If a model looks too perfect than it has some issue and please cross verify the model.
- Manage the balance between the accuracy and complexity of model because a complex model tends to fail.
- Check for model's performance on unseen dataset for accuracy and robustness

Implications of above strategies are significant but if a model is having good accuracy, robustness and less complex than it over powers the implications.