

## Assignment Part-II

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?

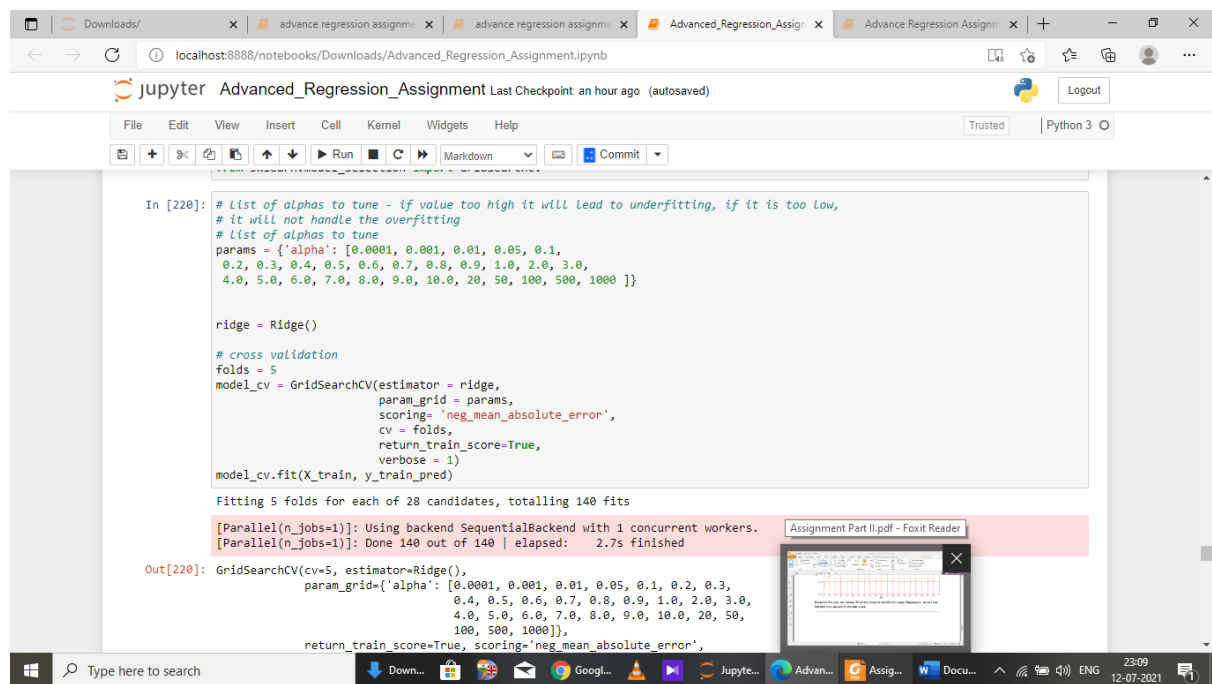
Answer 1: As per the calculations done in the iPython Notebook:

●Optimal value of Alpha for Ridge Regression is: 15

●Optimal value of Alpha for Ridge Regression is: 0.001

If we double the Alpha values for Ridge Regression, then:

●For Alpha = 15



```
In [220]: # List of alphas to tune - if value too high it will lead to underfitting, if it is too low,
# it will not handle the overfitting
# List of alphas to tune
params = {'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 ]}

ridge = Ridge()

# cross validation
folds = 5
model_cv = GridSearchCV(estimator = ridge,
                        param_grid = params,
                        scoring = 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)
model_cv.fit(X_train, y_train_pred)

Fitting 5 folds for each of 28 candidates, totalling 140 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 140 out of 140 | elapsed: 2.7s finished

Out[220]: GridSearchCV(cv=5, estimator=Ridge(),
                    param_grid={'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]},
                    return_train_score=True, scoring='neg_mean_absolute_error',
```

●For Alpha = 3

R2 value on the train dataset is: 0.8981689078710946

R2 value on the test dataset is: -222163.32913928787

RSS value on the train dataset is: 14.287332291574335

RSS value on the test dataset is: 47063693.660552606

MSE value on the train dataset is: 0.01523169753899183

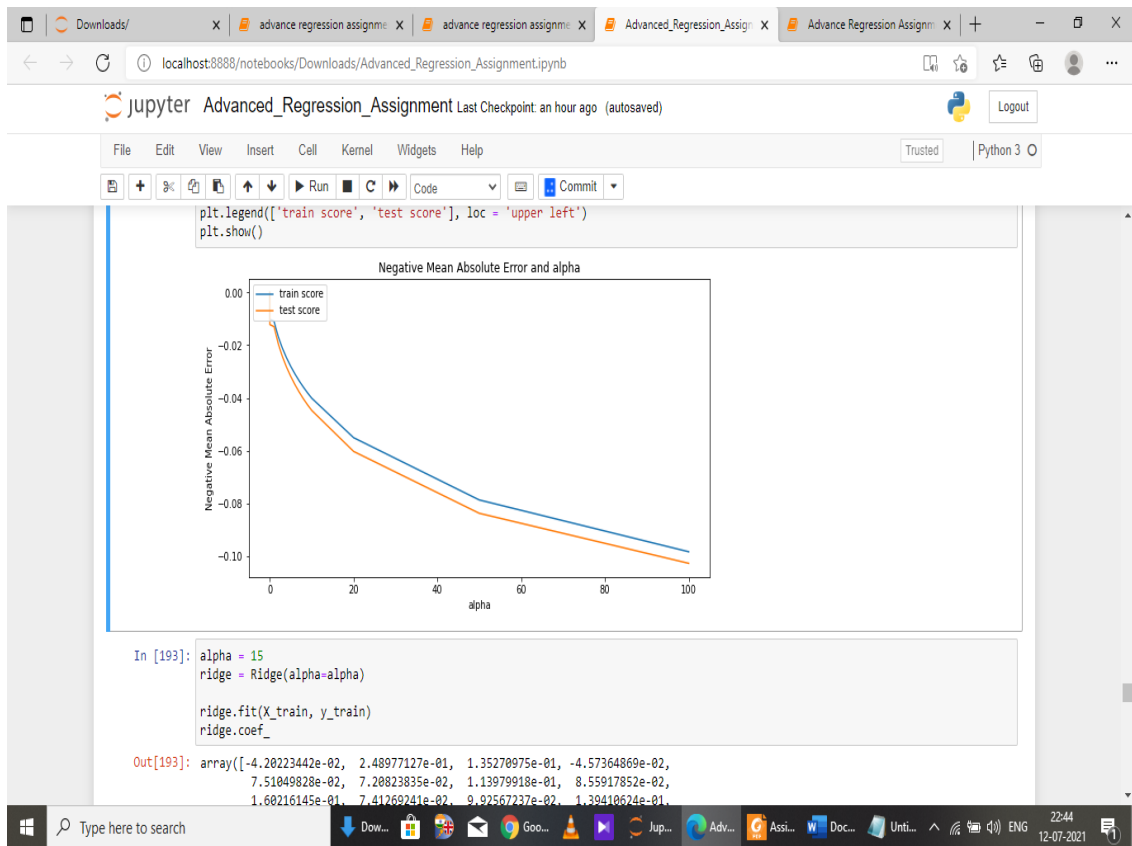
MSE value on the test dataset is: 35095.968426959436

## Question 2

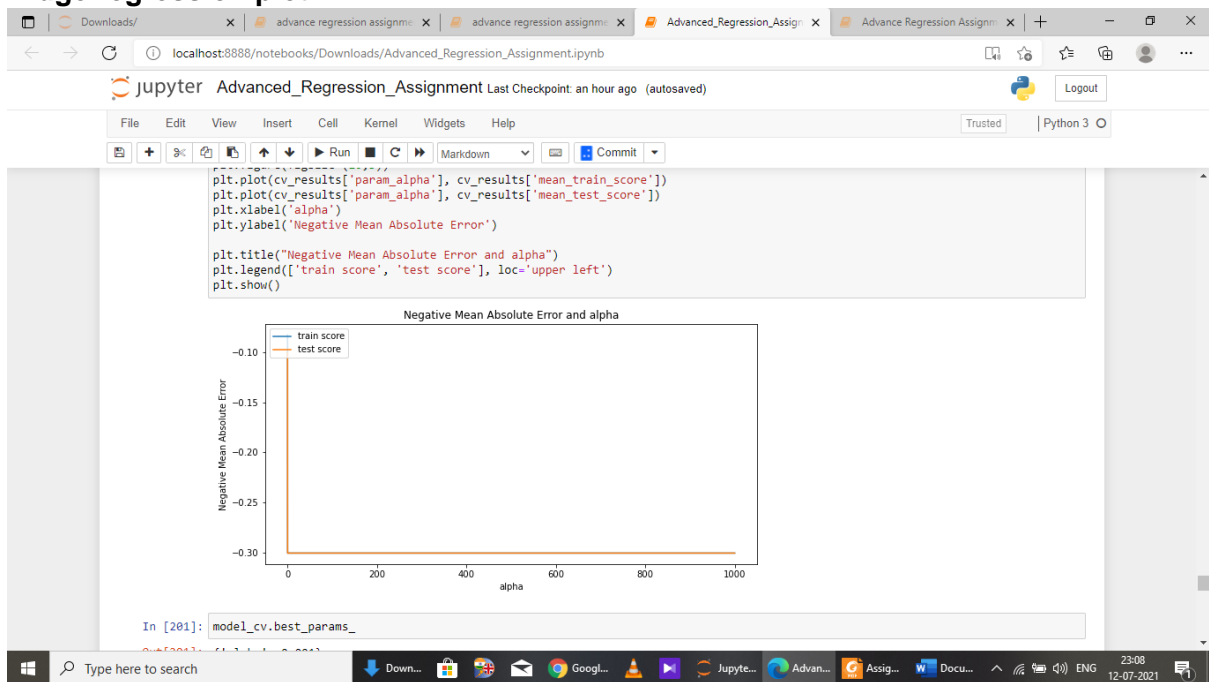
You have determined the optimal value of  $\lambda$  for ridge and lasso regression during the assignment. Now, which one will you choose to apply and why?

**Answer:**

We would decide that on the basis of plots and chose a value of  $\alpha$  where we have good training as well as the test score.



## Ridge regression plot:



## 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:

The screenshot shows a Jupyter Notebook titled 'Advanced\_Regression\_Assignment'. The code in the notebook performs a train-test split on the 'house\_price' variable and displays the first few rows of both the training and testing datasets.

```
In [122]: # Performing the test-train split:
df_train, df_test = train_test_split(house_price, train_size = 0.7, test_size = 0.3, random_state = 100)

In [123]: df_train.head()
Out[123]:
```

|      | LotShape | OverallQual | OverallCond | YearBuilt | ExterQual | HeatingQC | 2ndFlrSF | BsmtFullBath | FullBath | HalfBath | ... | GarageFinish_No Garage | Fence_2.0 | Fen |
|------|----------|-------------|-------------|-----------|-----------|-----------|----------|--------------|----------|----------|-----|------------------------|-----------|-----|
| 564  | 2        | 7           | 5           | 29        | 4         | 5         | 1129     | 1            | 2        | 1        | ... | 0                      | 0         |     |
| 162  | 3        | 7           | 5           | 16        | 4         | 5         | 0        | 0            | 2        | 0        | ... | 0                      | 0         |     |
| 1199 | 3        | 4           | 5           | 58        | 3         | 4         | 0        | 0            | 1        | 1        | ... | 0                      | 0         |     |
| 819  | 2        | 7           | 5           | 12        | 4         | 5         | 0        | 1            | 2        | 0        | ... | 0                      | 0         |     |
| 60   | 3        | 6           | 5           | 17        | 3         | 5         | 0        | 1            | 1        | 1        | ... | 0                      | 0         |     |

5 rows x 140 columns

```
In [124]: df_train.shape
Out[124]: (938, 140)

In [125]: df_test.head()
Out[125]:
```

|    | LotShape | OverallQual | OverallCond | YearBuilt | ExterQual | HeatingQC | 2ndFlrSF | BsmtFullBath | FullBath | HalfBath | ... | GarageFinish_No Garage | Fence_2.0 | Fen |
|----|----------|-------------|-------------|-----------|-----------|-----------|----------|--------------|----------|----------|-----|------------------------|-----------|-----|
| 13 | 2        | 7           | 5           | 15        | 4         | 5         | 0        | 0            | 2        | 0        | ... | 0                      | 0         |     |

The Windows taskbar at the bottom shows the time as 23:00 on 12-07-2021.

Downloads/ x advance regression assignm... x advance regression assignm... x Advanced\_Regression\_Assign... x Advance Regression Assignm... x + -

localhost:8888/notebooks/Downloads/Advanced\_Regression\_Assignment.ipynb

jupyter Advanced\_Regression\_Assignment Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code Commit

```
In [124]: df_train.shape
Out[124]: (938, 140)

In [125]: df_test.head()
Out[125]:
```

|     | LotShape | OverallQual | OverallCond | YearBuilt | ExterQual | HeatingQC | 2ndFlrSF | BsmtFullBath | FullBath | HalfBath | ... | GarageFinish_No<br>Garage | Fence_2.0 | Fenc |
|-----|----------|-------------|-------------|-----------|-----------|-----------|----------|--------------|----------|----------|-----|---------------------------|-----------|------|
| 13  | 2        | 7           | 5           | 15        | 4         | 5         | 0        | 0            | 2        | 0        | ... | 0                         | 0         |      |
| 330 | 2        | 5           | 4           | 57        | 3         | 3         | 0        | 0            | 2        | 0        | ... | 0                         | 0         |      |
| 342 | 3        | 3           | 4           | 72        | 3         | 2         | 0        | 0            | 2        | 0        | ... | 0                         | 0         |      |
| 886 | 3        | 5           | 5           | 62        | 3         | 5         | 0        | 0            | 2        | 0        | ... | 0                         | 0         |      |
| 164 | 3        | 6           | 7           | 95        | 3         | 4         | 467      | 0            | 2        | 0        | ... | 0                         | 0         |      |

5 rows x 140 columns

```
In [126]: df_test.shape
Out[126]: (403, 140)
```

### Step 6: Scaling the Features

```
In [127]: # Importing the relevant library:
```

Windows Type here to search

Downloads/ x advance regression assignm... x advance regression assignm... x Advanced\_Regression\_Assign... x Advance Regression Assignm... x + -

localhost:8888/notebooks/Downloads/Advanced\_Regression\_Assignment.ipynb

jupyter Advanced\_Regression\_Assignment Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Markdown Commit

```
1 1.0 6 8 45 5 0 2 0 3 3 ... 0 (

2 rows x 51 columns

In [184]: X_test_m1.shape
Out[184]: (1341, 51)

In [185]: # Calculating the predicted values on the test data:
y_test_pred = lm2.predict(X_test_m1)

In [186]: y_test_pred.head(2)
Out[186]: 0    214.145456
          1    176.645492
          dtype: float64

In [187]: y_test_pred.shape
Out[187]: (1341,)
```

```
In [ ]:
```

```
In [188]: # Importing necessary libraries:
from sklearn.metrics import mean_squared_error, r2_score

# Making predictions:
y_train_pred = lm2.predict(X_train_rfe_model)
```

```
Downloads/ x advance regression assignm... x advance regression assignm... x Advanced_Regression_Assign... x Advance Regression Assignm... x + -
```

localhost:8888/notebooks/Downloads/Advanced\_Regression\_Assignment.ipynb

jupyter Advanced\_Regression\_Assignment Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [201]: model\_cv.best\_params\_  
Out[201]: {'alpha': 0.001}

In [202]: *#Fitting Ridge model for alpha = 0.0001 and printing coefficients which have been penalised*  
lasso = Lasso(alpha = 0.0001)  
lasso.fit(X\_train\_rfe\_model, y\_train\_pred)  
Out[202]: Lasso(alpha=0.0001)

In [203]: lasso.coef\_  
Out[203]: array([[ 0. , 0.5125001 , 0.23052323, -0.18972871, 0.05517336,  
 0.1447164 , 0.20394189, 0.06915727, 0.05311005, 0.1353028 ,  
 0.44667522, 0.21555458, 0.35386425, 0.34302887, 0.31025173,  
 0.3280664 , 0.27022599, 0.05580037, 0.05932666, 0.08441529,  
 0.18012653, -0.08610648, 0.18630319, 0.11425085, 0.05519805,  
 0.20679717, -0.0716158 , -0.15812442, -0.05570659, -0.10128505,  
 0.05525268, 0.12203326, -0. , -0. , 0.16541034,  
 0.01051585, 0.05055252, -0.05196583, 0.05065807, 0.05418828,  
 0.13917498, -0.00649661, 0.04237915, 0.07721012, 0.14190447,  
 -0.04431516, 0.08114573, -0.03245788, -0.07916657, 0.05200369,  
 0.07460471])

In [204]: *# Lets calculate some metrics such as R2 score, RSS and MSE:*  
y\_train\_pred = lasso.predict(X\_train\_rfe\_model)  
y\_test\_pred = lasso.predict(X\_test\_m1)

```
Downloads/ x advance regression assignm... x advance regression assignm... x Advanced_Regression_Assign... x Advance Regression Assignm... x + -
```

localhost:8888/notebooks/Downloads/Advanced\_Regression\_Assignment.ipynb

jupyter Advanced\_Regression\_Assignment Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

-0.04431516, 0.08114573, -0.03245788, -0.07916657, 0.05200369,  
 0.07460471])

In [204]: *# Lets calculate some metrics such as R2 score, RSS and MSE:*  
y\_train\_pred = lasso.predict(X\_train\_rfe\_model)  
y\_test\_pred = lasso.predict(X\_test\_m1)  
  
metric\_lasso = []  
r2\_train\_lasso = r2\_score(y\_train, y\_train\_pred)  
print("R2 value on the train dataset is: ", r2\_train\_lasso)  
metric\_lasso.append(r2\_train\_lasso)  
  
r2\_test\_lasso = r2\_score(y\_test, y\_test\_pred)  
print("R2 value on the test dataset is: ", r2\_test\_lasso)  
metric\_lasso.append(r2\_test\_lasso)  
  
rss1\_lasso = np.sum(np.square(y\_train - y\_train\_pred))  
print("RSS value on the train dataset is: ", rss1\_lasso)  
metric\_lasso.append(rss1\_lasso)  
  
rss2\_lasso = np.sum(np.square(y\_test - y\_test\_pred))  
print("RSS value on the test dataset is: ", rss2\_lasso)  
metric\_lasso.append(rss2\_lasso)  
  
mse\_train\_lasso = mean\_squared\_error(y\_train, y\_train\_pred)  
print("MSE value on the train dataset is: ", mse\_train\_lasso)  
metric\_lasso.append(mse\_train\_lasso\*0.5)  
  
mse\_test\_lasso = mean\_squared\_error(y\_test, y\_test\_pred)  
print("MSE value on the test dataset is: ", mse\_test\_lasso)  
metric\_lasso.append(mse\_test\_lasso\*0.5)

The screenshot shows a Jupyter Notebook titled 'Advanced\_Regression\_Assignment' with a Python 3 kernel. The output of a cell displays a table of coefficients for Ridge and Lasso regression models across 21 features. The features include 'const', 'OverallQual', 'OverallCond', 'YearBuilt', 'HeatingQC', 'BsmtFullBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenQual', 'TotRmsAbvGrd', 'Fireplaces', 'GarageArea', 'MSZoning\_FV', 'MSZoning\_RH', 'MSZoning\_RL', 'MSZoning\_RM', 'LotConfig\_CulDSac', 'Neighborhood\_BrkSide', and 'Neighborhood\_ClearCr'. The 'const' row shows zero coefficients for both models. Other features have varying coefficients, with Ridge generally showing smaller values than Lasso for non-constant features.

|                      | Ridge     | Lasso     |
|----------------------|-----------|-----------|
| const                | 0.000000  | 0.000000  |
| OverallQual          | 0.492985  | 0.512500  |
| OverallCond          | 0.229835  | 0.230523  |
| YearBuilt            | -0.191333 | -0.189729 |
| HeatingQC            | 0.055626  | 0.055173  |
| BsmtFullBath         | 0.145608  | 0.144716  |
| FullBath             | 0.208516  | 0.203942  |
| HalfBath             | 0.072948  | 0.069157  |
| BedroomAbvGr         | 0.057642  | 0.053110  |
| KitchenQual          | 0.131233  | 0.135303  |
| TotRmsAbvGrd         | 0.441794  | 0.446675  |
| Fireplaces           | 0.213826  | 0.215555  |
| GarageArea           | 0.358962  | 0.353864  |
| MSZoning_FV          | 0.401204  | 0.343029  |
| MSZoning_RH          | 0.378263  | 0.310252  |
| MSZoning_RL          | 0.383974  | 0.328066  |
| MSZoning_RM          | 0.330787  | 0.270226  |
| LotConfig_CulDSac    | 0.057509  | 0.055800  |
| Neighborhood_BrkSide | 0.064477  | 0.059327  |
| Neighborhood_ClearCr | 0.092469  | 0.084415  |

## 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:

The model should be generalized 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 importance should not be given to the outliers so that the accuracy predicted by the model is high. To ensure that this is not the case, the outliers 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. If the model is not robust, it cannot be trusted for predictive analysis.