## 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?

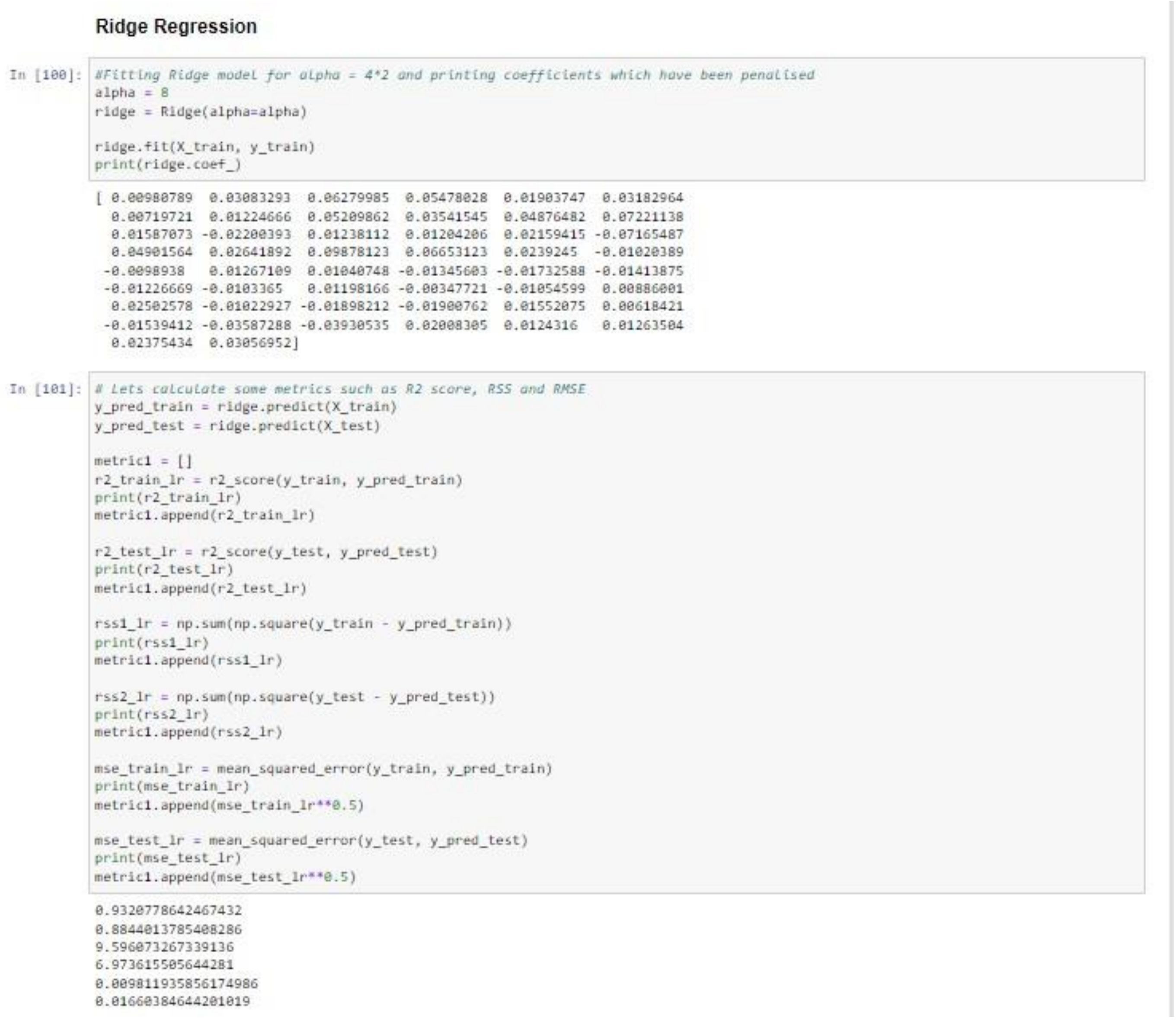
**Solution:**

Optimal value of alpha obtained for ridge regression : 4.0

Optimal value of alpha obtained for lasso regression : 0.001

If we choose to double the value of optimal alpha for ridge and lasso regression , then r2 score for train as well as test data slightly reduces.

i.e *Alpha for Ridge regression = 4 * 2 = 8*



**Ridge Regression**

```
In [100]: #Fitting Ridge model for alpha = 4*2 and printing coefficients which have been penalised
alpha = 8
ridge = Ridge(alpha=alpha)

ridge.fit(X_train, y_train)
print(ridge.coef_)
```

```
[ 0.00980789  0.03083293  0.06279985  0.05478028  0.01903747  0.03182964
  0.00719721  0.01224666  0.05209862  0.03541545  0.04876482  0.07221138
  0.01587073 -0.02200393  0.01238112  0.01204206  0.02159415 -0.07165487
  0.04901564  0.02641892  0.09878123  0.06653123  0.0239245  -0.01020389
 -0.0098938   0.01267109  0.01040748 -0.01345603 -0.01732588 -0.01413875
 -0.01226669 -0.0103365   0.01198166 -0.00347721 -0.01054599  0.00886001
  0.02502578 -0.01022927 -0.01898212 -0.01900762  0.01552075  0.00618421
 -0.01539412 -0.03587288 -0.03930535  0.02008305  0.0124316   0.01263504
  0.02375434  0.03056952]
```

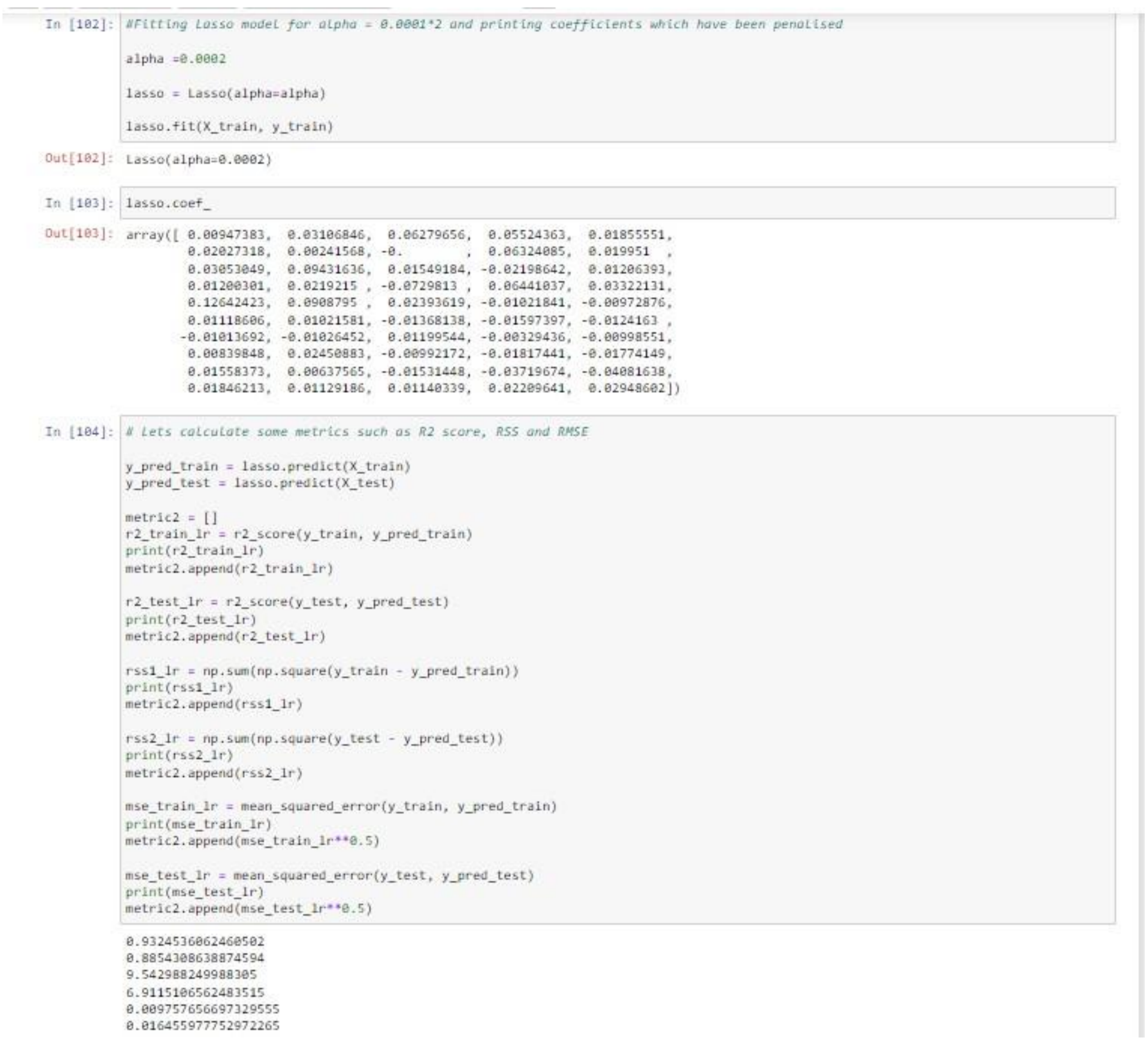
```
In [101]: # Lets calculate some metrics such as R2 score, RSS and RMSE
y_pred_train = ridge.predict(X_train)
y_pred_test = ridge.predict(X_test)

metric1 = []
r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric1.append(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)
metric1.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
metric1.append(rss1_lr)

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(rss2_lr)
metric1.append(rss2_lr)

mse_train_lr = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric1.append(mse_train_lr**0.5)

mse_test_lr = mean_squared_error(y_test, y_pred_test)
print(mse_test_lr)
metric1.append(mse_test_lr**0.5)
```

```
0.9320778642467432
0.8844013785408286
9.596073267339136
6.973615505644281
0.009811935856174986
0.01660384644201019
```

The r2 score for train set and test set got slightly decreased from what we got from the optimal alpha values in Ridge regression

And *Alpha for Lasso regression = 0.001 * 2 = 0.0002*

**Lasso Regression**

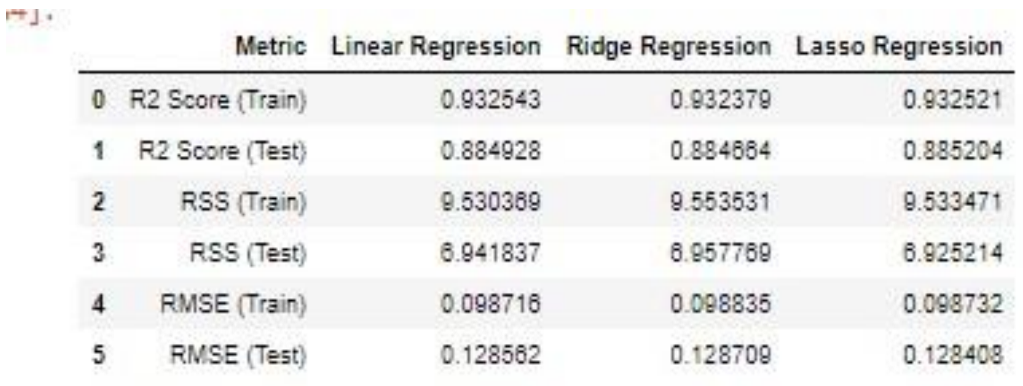


```
In [102]: #Fitting Lasso model for alpha = 0.0001*2 and printing coefficients which have been penalised

          alpha =0.0002

          lasso = Lasso(alpha=alpha)

          lasso.fit(X_train, y_train)

Out[102]: Lasso(alpha=0.0002)

In [103]: lasso.coef_

Out[103]: array([ 0.00947383,  0.03106846,  0.06279656,  0.05524363,  0.01855551,
                  0.02027318,  0.00241568, -0.        ,  0.06324085,  0.019951  ,
                  0.03053049,  0.09431636,  0.01549184, -0.02198642,  0.01206393,
                  0.01200301,  0.0219215 , -0.0729813 ,  0.06441037,  0.03322131,
                  0.12642423,  0.0908795 ,  0.02393619, -0.01021841, -0.00972876,
                  0.01118606,  0.01021581, -0.01368138, -0.01597397, -0.0124163 ,
                 -0.01013692, -0.01026452,  0.01199544, -0.00329436, -0.00998551,
                  0.00839848,  0.02450883, -0.00992172, -0.01817441, -0.01774149,
                  0.01558373,  0.00637565, -0.01531448, -0.03719674, -0.04081638,
                  0.01846213,  0.01129186,  0.01140339,  0.02209641,  0.02948602])

In [104]: # Lets calculate some metrics such as R2 score, RSS and RMSE

          y_pred_train = lasso.predict(X_train)
          y_pred_test = lasso.predict(X_test)

          metric2 = []
          r2_train_lr = r2_score(y_train, y_pred_train)
          print(r2_train_lr)
          metric2.append(r2_train_lr)

          r2_test_lr = r2_score(y_test, y_pred_test)
          print(r2_test_lr)
          metric2.append(r2_test_lr)

          rss1_lr = np.sum(np.square(y_train - y_pred_train))
          print(rss1_lr)
          metric2.append(rss1_lr)

          rss2_lr = np.sum(np.square(y_test - y_pred_test))
          print(rss2_lr)
          metric2.append(rss2_lr)

          mse_train_lr = mean_squared_error(y_train, y_pred_train)
          print(mse_train_lr)
          metric2.append(mse_train_lr**0.5)

          mse_test_lr = mean_squared_error(y_test, y_pred_test)
          print(mse_test_lr)
          metric2.append(mse_test_lr**0.5)

          0.9324536062460502
          0.8854308638874594
          9.542988249988305
          6.9115106562483515
          0.009757656697329555
          0.016455977752972265
```

The r2 score for train set got slightly decreased, whereas the r2 score for test set got increased slightly in comparison with the values we got from  the optimal alpha values in Lasso regression.

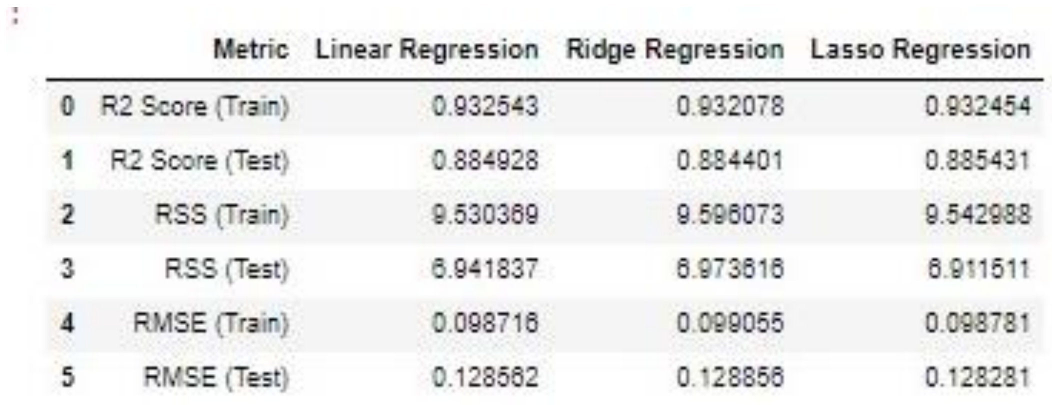
Now, let's see the metrics,

For optimal values we got:-



| | Metric | Linear Regression | Ridge Regression | Lasso Regression |
|---|---|---|---|---|
| 0 | R2 Score (Train) | 0.932543 | 0.932379 | 0.932521 |
| 1 | R2 Score (Test) | 0.884928 | 0.884664 | 0.885204 |
| 2 | RSS (Train) | 9.530369 | 9.553531 | 9.533471 |
| 3 | RSS (Test) | 6.941837 | 6.957769 | 6.925214 |
| 4 | RMSE (Train) | 0.098716 | 0.098835 | 0.098732 |
| 5 | RMSE (Test) | 0.128562 | 0.128709 | 0.128408 |

After Doubling the optimal alpha values for Ridge and Lasso regression, we got the metrics as :



| | Metric | Linear Regression | Ridge Regression | Lasso Regression |
|---|---|---|---|---|
| 0 | R2 Score (Train) | 0.932543 | 0.932078 | 0.932454 |
| 1 | R2 Score (Test) | 0.884928 | 0.884401 | 0.885431 |
| 2 | RSS (Train) | 9.530369 | 9.596073 | 9.542988 |
| 3 | RSS (Test) | 6.941837 | 6.973616 | 6.911511 |
| 4 | RMSE (Train) | 0.098716 | 0.099055 | 0.098781 |
| 5 | RMSE (Test) | 0.128562 | 0.128856 | 0.128281 |

So , when we double the optimal values of alpha for Ridge and Lasso regression we can conclude :

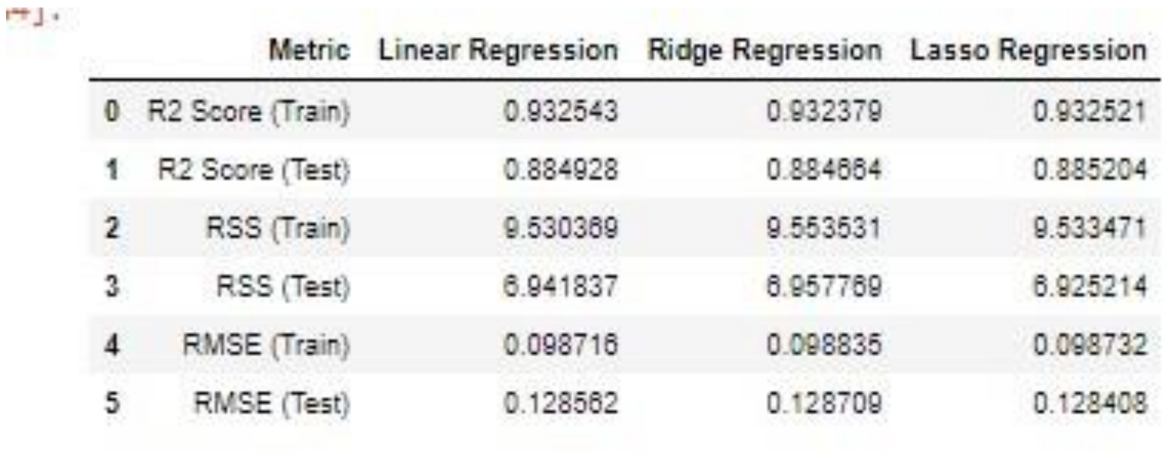
- The r2 score for train set (0.9323 to 0.9320) and test set (0.8846 to 0.8844) got slightly decreased from what we got from the optimal alpha values in Ridge regression.
- The r2 score for train set (0.9325 to 0.9324) got slightly decreased, whereas the r2 score for test set (0.8852 to 0.8854) got increased slightly in comparison with the values we got from the optimal alpha values in Lasso regression.
- The RSS for train set (9.55 to 9.59) and test set (6.95 to 6.97) got slightly increased, in the case of Ridge regression.
- The RSS for train set (9.53 to 9.54) got slightly increased, whereas the RSS for test set (6.92 to 6.91) got decreased slightly in comparison with the values we got from the optimal alpha values in Lasso regression.

**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?

**Solution:**

The evaluation metrics, in Linear regression, Ridge regression and Lasso regression is as follows:



|   | Metric | Linear Regression | Ridge Regression | Lasso Regression |
|---|---|---|---|---|
| 0 | R2 Score (Train) | 0.932543 | 0.932379 | 0.932521 |
| 1 | R2 Score (Test) | 0.884928 | 0.884664 | 0.885204 |
| 2 | RSS (Train) | 9.530369 | 9.553531 | 9.533471 |
| 3 | RSS (Test) | 6.941837 | 6.957769 | 6.925214 |
| 4 | RMSE (Train) | 0.098716 | 0.098835 | 0.098732 |
| 5 | RMSE (Test) | 0.128562 | 0.128709 | 0.128408 |

I will choose to go with Lasso regression over others because the R2 score for Train set is almost same as normal Linear regression, whereas r2 score for test set got slightly improved in Lasso regression as compared to Ridge regression.

Also, the RSS value is slightly less in Lasso regression for test set.

**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?

**Solution:**

The First five most predictor variable were :

-MSZoning_RL

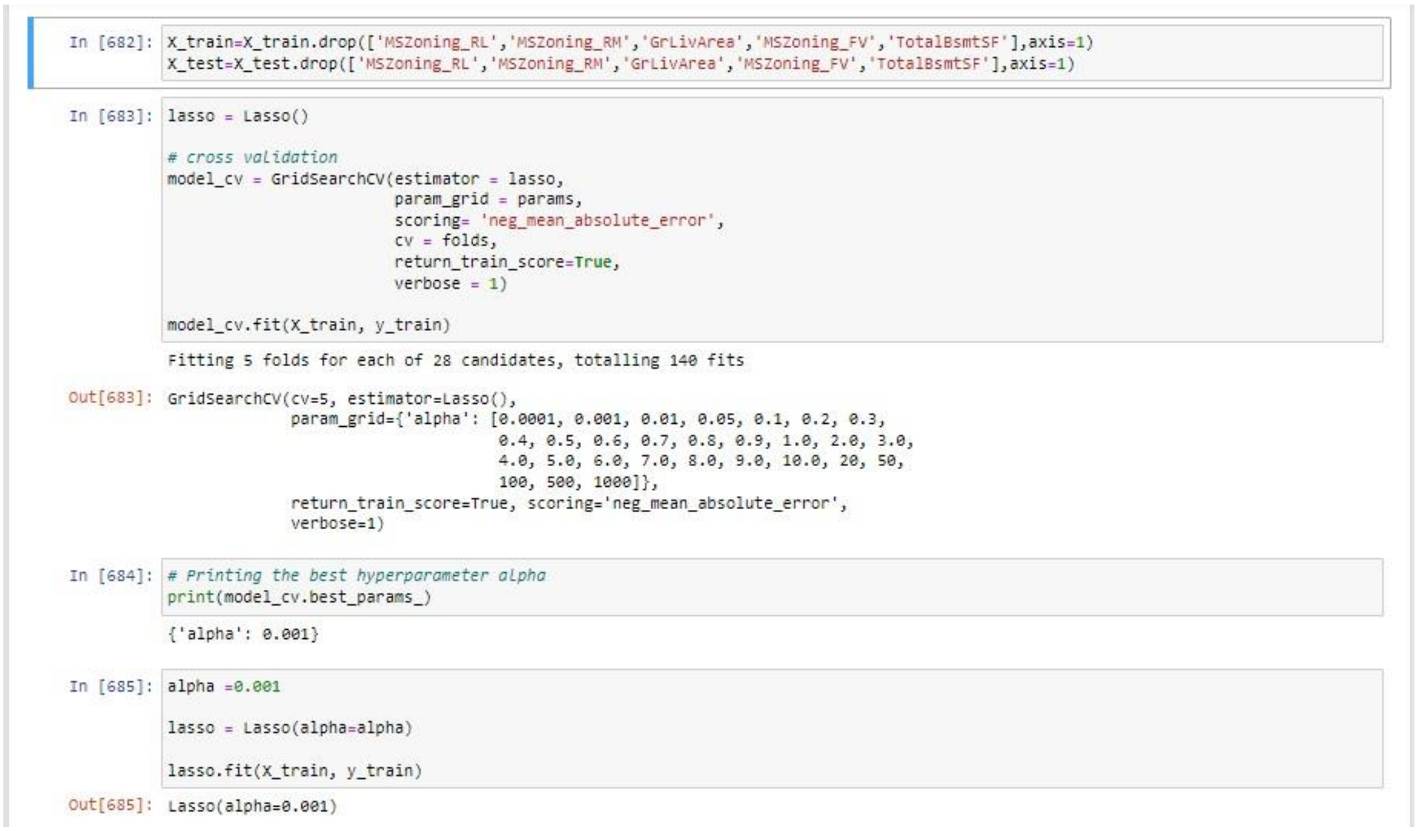
- GrLivArea

- MSZoning_RM

- MSZoning_FV

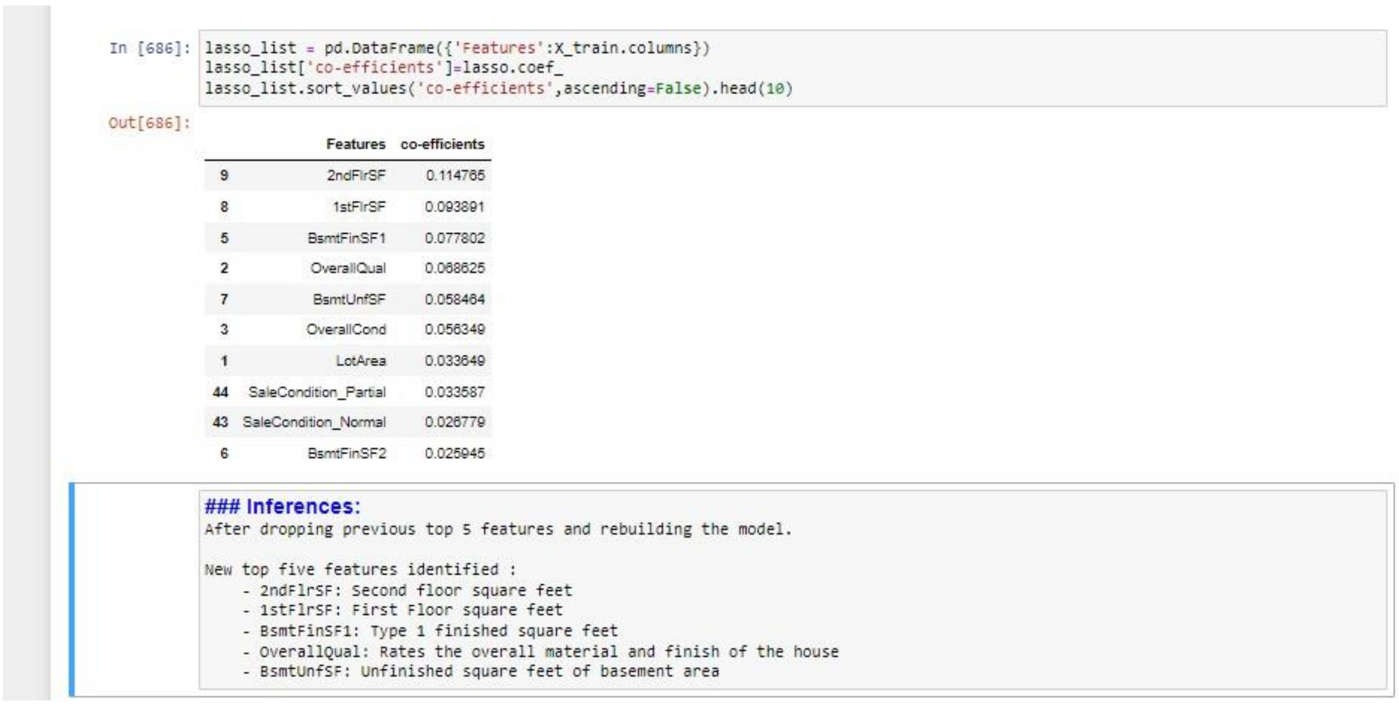
- TotalBsmtSF

If the above variables are not present then we will go for creating the model by removing the above features.



```
In [682]: X_train=X_train.drop(['MSZoning_RL','MSZoning_RM','GrLivArea','MSZoning_FV','TotalBsmtSF'],axis=1)
          X_test=X_test.drop(['MSZoning_RL','MSZoning_RM','GrLivArea','MSZoning_FV','TotalBsmtSF'],axis=1)

In [683]: lasso = Lasso()

          # cross validation
          model_cv = GridSearchCV(estimator = lasso,
                                  param_grid = params,
                                  scoring= 'neg_mean_absolute_error',
                                  cv = folds,
                                  return_train_score=True,
                                  verbose = 1)

          model_cv.fit(X_train, y_train)

          Fitting 5 folds for each of 28 candidates, totalling 140 fits

Out[683]: GridSearchCV(cv=5, estimator=Lasso(),
                       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',
                       verbose=1)

In [684]: # Printing the best hyperparameter alpha
          print(model_cv.best_params_)

          {'alpha': 0.001}

In [685]: alpha =0.001

          lasso = Lasso(alpha=alpha)

          lasso.fit(X_train, y_train)

Out[685]: Lasso(alpha=0.001)
```

Let’s now check the Most important variables affecting the target variable :



```
In [686]: lasso_list = pd.DataFrame({'Features':X_train.columns})
          lasso_list['co-efficients']=lasso.coef_
          lasso_list.sort_values('co-efficients',ascending=False).head(10)

Out[686]:
```

| | Features | co-efficients |
|---|---|---|
| 9 | 2ndFlrSF | 0.114765 |
| 8 | 1stFlrSF | 0.093891 |
| 5 | BsmtFinSF1 | 0.077802 |
| 2 | OverallQual | 0.068625 |
| 7 | BsmtUnfSF | 0.058464 |
| 3 | OverallCond | 0.056349 |
| 1 | LotArea | 0.033649 |
| 44 | SaleCondition_Partial | 0.033587 |
| 43 | SaleCondition_Normal | 0.026779 |
| 6 | BsmtFinSF2 | 0.025945 |

```
### Inferences:
After dropping previous top 5 features and rebuilding the model.

New top five features identified :
    - 2ndFlrSF: Second floor square feet
    - 1stFlrSF: First Floor square feet
    - BsmtFinSF1: Type 1 finished square feet
    - OverallQual: Rates the overall material and finish of the house
    - BsmtUnfSF: Unfinished square feet of basement area
```

Hence, we can conclude that if top five most important predictor variables are not available then the new top five predictor variables in a lasso model will be:

- 2ndFlrSF: Second floor square feet
- 1stFlrSF: First Floor square feet
- BsmtFinSF1: Type 1 finished square feet
- OverallQual: Rates the overall material and finish of the house
- BsmtUnfSF: Unfinished square feet of basement area

**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?

## Solution:

A model is said to be robust and generalisable if the predictor variable is consistently accurate even one or more of the independent variables are changed drastically.

So, we need to make a model which has more explainability, i.e

To trust the model we need to have some access to the decision making process and also be able to monitor and control it.

And to increase the explainability, sometimes it’s worth sacrificing a small percentage of accuracy in order to have a more explainable model.

The other way could be, to use Property based testing on the model.

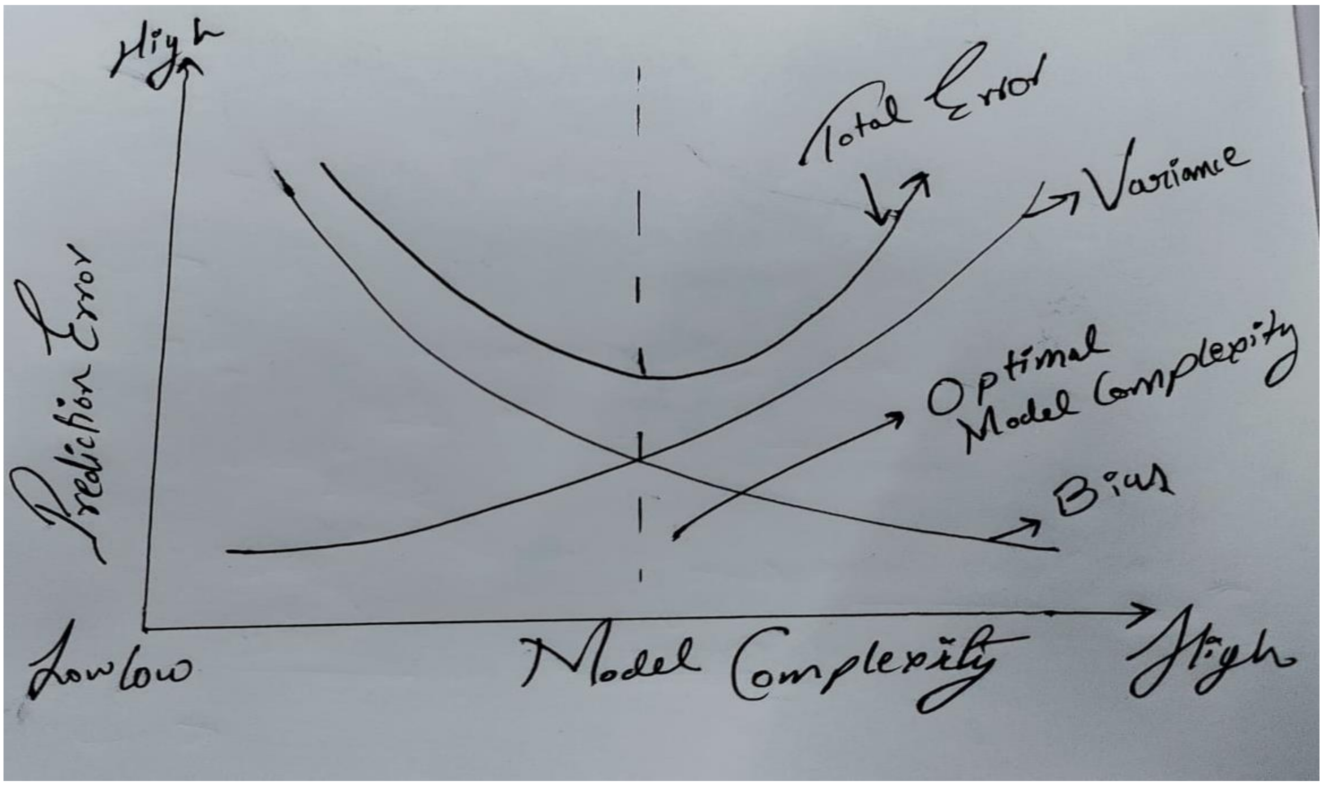
Property based testing aims to maintain a certain property given an operational domain, allowing us to estimate the robustness of the model.

In terms of accuracy,

The model should not have much difference in train and test accuracy. (It should have a considerably good accuracy in both train and test sets)

i.e. The model should be accurate enough for the datasets other than the ones which were used during the training.

In order for it to be generalisable, we need to ensure that the ML model is not getting into overfitting and underfitting stages, but have a right balance of Bias and Variance.


High
Total Error
Variance
Prediction Error
Optimal Model Complexity
Bias
Low
Low
Model Complexity
High