# Problem Statement – 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:**
Optimal value of alpha for:
- ridge regression: 7.0
- lasso regression: 0.001

It can also be seen that for Ridge and lasso, on doubling the alpha value, gap between training and test R-Squared goes up.
Root mean squared error has gone up for both ridge and lasso.

Top 5 most important predictor variables using:

- ridge:
    - Fence_GdWo: Fence quality, good wood
    - OverallCond: Overall condition of the house
    - LotFrontage: Linear feet of street connected to property
    - GarageType_N.A.: No Garage
    - YearBuilt_diff: Original construction date (difference from from recent)

- lasso:
    - Fence_GdWo: Fence quality, good wood
    - GarageType_N.A.: No Garage
    - YearBuilt_diff: Original construction date (difference from from recent)
    - OverallCond: Overall condition of the house
    - LotFrontage: Linear feet of street connected to property

Note: Relevant code could be found here:
https://github.com/rahul2july/housingpriceprediction/blob/main/Housing_Price_Prediction.ipynb
[Under Solving Subjective Questions]

**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?

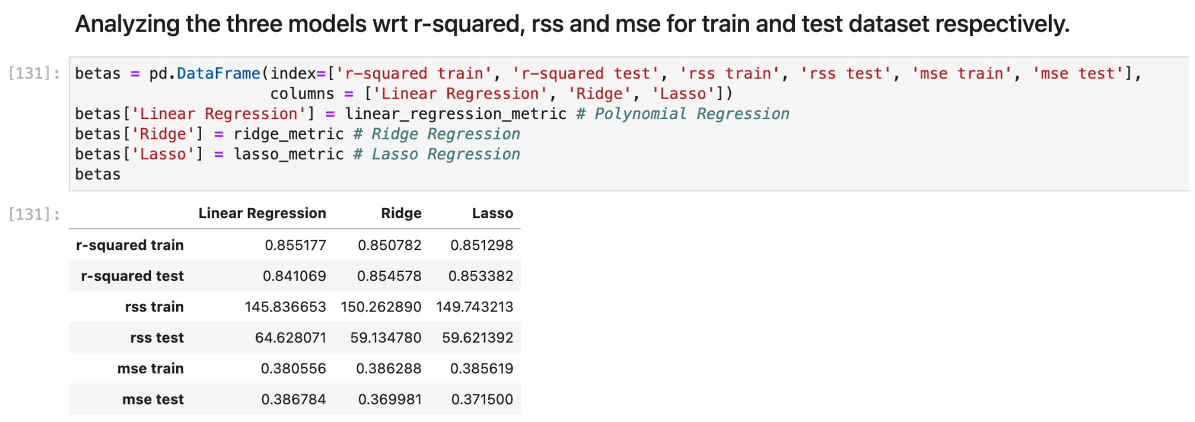
**Answer:**

Based on the assignment, we could see that Residual Sum of Squares(RSS) is very close for both ridge and lasso regression.

We could see that Lasso seems to be performing a little better out of the three models wrt. Root mean square error (RMSE)

Since Lasso will penalize more on the dataset and this could also help in feature elimination and making model more robust, we will choose this for the model.

Attaching the results for reference:



**Analyzing the three models wrt r-squared, rss and mse for train and test dataset respectively.**

```
[131]: betas = pd.DataFrame(index=['r-squared train', 'r-squared test', 'rss train', 'rss test', 'mse train', 'mse test'],
                             columns = ['Linear Regression', 'Ridge', 'Lasso'])
       betas['Linear Regression'] = linear_regression_metric # Polynomial Regression
       betas['Ridge'] = ridge_metric # Ridge Regression
       betas['Lasso'] = lasso_metric # Lasso Regression
       betas
```

[131]:

| | Linear Regression | Ridge | Lasso |
|---|---|---|---|
| r-squared train | 0.855177 | 0.850782 | 0.851298 |
| r-squared test | 0.841069 | 0.854578 | 0.853382 |
| rss train | 145.836653 | 150.262890 | 149.743213 |
| rss test | 64.628071 | 59.134780 | 59.621392 |
| mse train | 0.380556 | 0.386288 | 0.385619 |
| mse test | 0.386784 | 0.369981 | 0.371500 |

**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:**

**Code and Solution:**
Top 5 features defining "SalePrice" using lasso model[with alpha as 0.001] currently are:
-Fence_GdWo
-GarageType_N.A.
-YearBuilt_diff
-OverallCond
-LotFrontage

Code for removing the five most important predictor variables in lasso model:

```
# lasso regression
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)

# best hyperparameters
model_cv.best_params_

# drop top 5 most important features as per Lasso
top_features = ['Fence_GdWo', 'GarageType_N.A.', 'YearBuilt_diff', 'OverallCond', 'LotFrontage']
#X_train.drop(top_features, axis=1, inplace=True)
#X_test.drop(top_features, axis=1, inplace=True)

# Fit Lasso model for best params
alpha = model_cv.best_params_['alpha']
lasso = Lasso(alpha=alpha)
lasso.fit(X_train, y_train)
# lasso.coef_

# most important variables
pd.DataFrame(zip(X_train.columns[lasso.coef_ > 0], lasso.coef_[lasso.coef_ > 0]), columns=['ColumnName', 'Coeff']).sort_values('Coeff',
ascending=False).head(5).ColumnName.values
pd.DataFrame(zip(X_train.columns[lasso.coef_ > 0], lasso.coef_[lasso.coef_ > 0]), columns=['ColumnName', 'Coeff']).sort_values('Coeff',
ascending=False).head(5)
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
[138]:

      ColumnName              Coeff

  16  Neighborhood_NridgHt    0.481223

  15  Neighborhood_NoRidge    0.454904

  10  SaleType_New            0.410558

  3   GrLivArea               0.365664

  0   OverallQual             0.33709
```

So the five most important features now:

- Neighborhood_NridgHt: Physical locations within Ames city limits- Northridge Heights
- Neighborhood_NoRidge: Physical locations within Ames city limits- Northridge
- SaleType_New: Type of sale - Home just constructed and sold
- GrLivArea: Above grade (ground) living area square feet
- OverallQual: Overall material and finish of the house

**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:**
In summary, a model is robust when any variation in the data does not affect its performance much.
A generalizable model is able to adapt properly to new, previously unseen data, drawn from the same distribution as the one used to create the model.
The way to ensure model is robust and generalized is to check the test score and ensure training and test set are almost similar.
This eventually means that training set can reduce a bit to avoid overfitting.
So the model should not be too complex in order to be robust and generalizable.
Also outlier analysis and correlation analysis need to be done and only relevant attributes should be retained in the final dataset.
If model is not robust, it cannot be trusted for predictive analysis.

If we look at the from the perspective of accuracy, a too complex model will have a very high accuracy.
So, to make our model more robust and generalizable, we will have to decrease variance which will lead to some bias and this will a cause some decrease in accuracy.
In general, we have to find balance between model accuracy and complexity.
This could be achieved by Regularization techniques like Ridge Regression and Lasso Regression.