Part-II Problem Statement (Subjective Questions)

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?

- The optimal value for alpha = 100 in Ridge regression, now we double the alpha values i.e. alpha = 200 on ridge and we found the most important (15) predictor variables below.

  - *GrLivArea, OverallQual, 2ndFlrSF, 1stFlrSF, Neighborhood_NoRidge, TotalBsmtSF, RoofMatl_WdShngl, Neighborhood_NridgHt, BsmtExposure_Gd, BsmtFinSF1, MasVnrArea, GarageCars, OverallCond, LotArea, TotRmsAbvGrd*

- The optimal value for alpha = 500 in Lasso regression, now we double the alpha values i.e. alpha = 1000 on lasso and and we found the most important (15) predictor variables below.

  - *GrLivArea, OverallQual, TotalBsmtSF, YearBuilt, BsmtFinSF1, Neighborhood_NoRidge, BsmtExposure_Gd, Neighborhood_NridgHt, OverallCond, SaleType_New, RoofMatl_WdShngl, YearRemodAdd, LotArea, Neighborhood_Crawfor, MasVnrArea*

Below is the code calculates after doubling the lambda & find the elements for Ridge and Lasso regression.

Alpha Regression:
```
alpha = 200
ridge = Ridge(alpha = alpha)
ridge.fit(X_train, y_train)

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

r2_train = r2_score(y_train, y_pred_train)
r2_test = r2_score(y_test, y_pred_test)
rss_train = np.sum(np.square(y_train, y_pred_train))
rss_test = np.sum(np.square(y_test, y_pred_test))
mse_train = mean_squared_error(y_train, y_pred_train)
mse_test = mean_squared_error(y_test, y_pred_test)

betas = pd.DataFrame(index = X.columns)
betas.rows = X.columns
betas['ridge'] = ridge.coef_
betas.ridge.sort_values(ascending=False).head(15)
```

Lasso Regression
```
alpha = 1000
lasso = Lasso(alpha = alpha)
lasso.fit(X_train, y_train)

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

r2_train = r2_score(y_train, y_pred_train)
r2_test = r2_score(y_test, y_pred_test)
```

```
rss_train = np.sum(np.square(y_train, y_pred_train))
rss_test = np.sum(np.square(y_test, y_pred_test))
mse_train = mean_squared_error(y_train, y_pred_train)
mse_test = mean_squared_error(y_test, y_pred_test)

betas = pd.DataFrame(index = X.columns)
betas.rows = X.columns
betas['lasso'] = lasso.coef_
betas.lasso.sort_values(ascending=False).head(15)
```

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?

- Generally after determined the optimal value of lambda, we need to build both ridge and lasso regression. We can choose the Lasso regression when both the performance is same, since Lasso does automatic feature selection (and keep the number of predictor variables as less as possible.

- In House market analysis, I have determined the optimal value of lambda in both cases i.e. lambda = 100 in Ridge and lambda = 500 in Lasso. But in this scenario, based on this model build result, I choose lambda = 100 in Ridge, because Ridge regression gives better predictive result on unseen data (test data) than Lasso though automatic feature selection is not applied.

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 dropping the five most important predictor variables in the lasso model, We created another model excluding the five most important predictor variables.

Here is the five most important predictor variables from new model.

- 1stFlrSF
- 2ndFlrSF
- BsmtExposure_Gd
- Neighborhood_NoRidge
- Neighborhood_NridgHt

Test code to find predictor variables after dropping the first five important predictor elements.

```
X_train2 = X_train
X_test2 = X_test
y_train2 = y_train
y_test2 = y_test

X_train2 = X_train2.drop('GrLivArea', axis = 1)
X_train2 = X_train2.drop('OverallQual', axis = 1)
X_train2 = X_train2.drop('TotalBsmtSF', axis = 1)
X_train2 = X_train2.drop('YearBuilt', axis = 1)
X_train2 = X_train2.drop('BsmtFinSF1', axis = 1)
X_test2 = X_test2.drop('GrLivArea', axis = 1)
X_test2 = X_test2.drop('OverallQual', axis = 1)
```

```
X_test2 = X_test2.drop('TotalBsmtSF', axis = 1)
X_test2 = X_test2.drop('YearBuilt', axis = 1)
X_test2 = X_test2.drop('BsmtFinSF1', axis = 1)
print(X_train.shape, X_train2.shape)

params2 = {'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 ]}
lasso2 = Lasso()
model_cv2 = GridSearchCV(estimator=lasso2, param_grid = params2, scoring = 'neg_mean_absolute_error', cv = 5,
                    return_train_score = True, verbose = 1)
model_cv2.fit(X_train2, y_train2)

alpha = 500
lasso2 = Lasso(alpha = alpha)
lasso2.fit(X_train2, y_train2)
print(lasso2.coef_)

y_pred_train2 = lasso2.predict(X_train2)
y_pred_test2 = lasso2.predict(X_test2)

r2_train = r2_score(y_train2, y_pred_train2)
r2_test = r2_score(y_test2, y_pred_test2)
rss_train = np.sum(np.square(y_train2, y_pred_train2))
rss_test = np.sum(np.square(y_test2, y_pred_test2))
mse_train = mean_squared_error(y_train2, y_pred_train2)
mse_test = mean_squared_error(y_test2, y_pred_test2)

betas2 = pd.DataFrame(index = X_train2.columns)
betas2.rows = X_train2.columns
betas2['lasso2'] = lasso2.coef_
betas2.lasso2.sort_values(ascending=False).head(15)
```

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?

Model has to be build for robust and generalisable,

- For generalisable, the dataset does not have more noisy or outlier elements in the dataset. We can split the dataset to training, validation and test, the training & validation datas are used for developing the model and Test data is kept apart (unseen) to check the performance of this model. This method checks the generalisable of the model.

- When not much weightage is given to outliers in the data, we can build accuracy of the model high. So outlier analysis need to perform to remove the outlier data from dataset, this helps to build the model accuracy.