Subjective Questions Solutions

Assignment-based Subjective Questions

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 alpha for ridge and lasso regression

Ridge Alpha 3

lasso Alpha 0.0001

```
529]: # Changing the alpha for ridge from alpha = 3 to alpha = 6

#Fitting Ridge model for alpha = 6 and printing coefficients which have been penalised
alpha = 6
ridge2 = Ridge(alpha=alpha)

ridge2.fit(X_train, y_train)
# Lets calculate some metrics such as R2 score, RSS and RMSE
y_pred_train = ridge2.predict(X_train)
y_pred_test = ridge2.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)
```

```
0.9334276150039053
0.8842588176780908
10.699880310003639
8.341301620335436
0.010469550205483012
0.019044067626336612
```

For alpha = 3

R2 Score (Train) (Ridge) = 0.938890

R2 Score (Test) (Ridge) 0.884055

for alpha = 6

R2(Train) 0.9333

R2(Test) 0.88425

Both training and test R2 score after increasing to alpha = 6 would be almost same to R2 score having alpha = 3.

```
# Lasso alpha from 0.0001 to 0.0002
alpha = 0.0002
lasso2 = Lasso(alpha=alpha)
lasso2.fit(X_train, y_train)

# Lets calculate some metrics such as R2 score, RSS and RMSE
y_pred_train = lasso2.predict(X_train)
y_pred_test = lasso2.predict(X_test)

metric3 = []
r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric3.append(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)
metric3.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
metric3.append(rss1_lr)

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(rss2_lr)
metric3.append(rss2_lr)

mse_train_lr = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric3.append(mse_train_lr**0.5)

mse_test_lr = mean_squared_error(y_test, y_pred_test)
print(mse_test_lr)
metric3.append(mse_test_lr**0.5)
```

```
0.9407400457550951
0.881499257883629
9.524586172389213
8.540179151419705
0.009319555941672 42
0.01949812591648334 6
```

For alpha = 0.0001

R2 Score (Train) (Lasso) = 0.944417 R2 Score (Test) (Lasso) = 0.879994

for alpha = 0.0002

R2(Train) 0.9407400457550951 R2(Test) 0.881499257883629

R2 Score of training got decreased a very very small but test score got increased.

[531]:
```python
#important predictor variables
betas = pd.DataFrame(index=X_train.columns)
betas.rows = X_train.columns
betas['Ridge2'] = ridge2.coef_
betas['Ridge'] = ridge.coef_
betas['Lasso'] = lasso.coef_
betas['Lasso2'] = lasso2.coef_
pd.set_option('display.max_rows', None)
betas.head(68)
```

t[531]:

	Ridge2	Ridge	Lasso	Lasso2
mssub_class	0.008373	0.008433	0.004807	0.002245
lot_frontage	0.018181	0.013354	0.000000	0.000000
lot_area	0.105440	0.115872	0.129950	0.119203
mas_vnr_area	-0.013108	-0.013927	-0.010836	-0.007451
bsmt_fin_sf1	0.137310	0.141950	0.131529	0.136965
bsmt_unf_sf	0.031844	0.030678	0.017342	0.011926
total_bsmt_sf	0.146541	0.153040	0.155003	0.152418
1st_flr_sf	0.202886	0.219537	0.247965	0.250971
2nd_flr_sf	0.085213	0.092203	0.122518	0.119446
gr_liv_area	0.187892	0.208080	0.242888	0.244508
bedroom_abv_gr	0.045754	0.038370	0.016976	0.013558

Predictors are same but the coefficent of these predictor has changed

Predictors are same only the coefficients of these predictors have changed.

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 :

R2(Train) (Ridge)= 0.938890 (Lasso) = 0.944417

R2(Test) (Ridge)= 0.884055 (Lasso) = 0.879994

Ridge alpha = 3

Lasso alpha = 0.0001

The r2_score of lasso is better for train but for test ridge is slightly better.
However, We should still use lasso model for the predictions as it also does feature selections . Better to use lasso with respect to that and to make the model more robust.

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 : Firstly , these variables are most significant in predicting the prices of a house

1. mszoning_RH
2. overall_qual_10
3. mszoning_RL
4. mszoning_FV
5. mszoning_RM

Now let's drop these columns and again do the modelling

X_train1 = X_train.drop(['overall_qual_10', 'mszoning_RH', 'mszoning_RL', 'mszoning_FV', 'mszoning_RM'], axis=1)

X_test1 = X_test.drop(['overall_qual_10', 'mszoning_RH', 'mszoning_RL', 'mszoning_FV', 'mszoning_RM'], axis=1)

```
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_train1, y_train)
#Fitting Lasso model for alpha = 100 and printing coefficients which have been penalised

alpha = model_cv.best_params_['alpha']

lasso3 = Lasso(alpha=alpha)

lasso3.fit(X_train1, y_train)

# Lets calculate some metrics such as R2 score, RSS and RMSE

y_pred_train = lasso3.predict(X_train1)
y_pred_test = lasso3.predict(X_test1)

metric3 = []
r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric3.append(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)
metric3.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
metric3.append(rss1_lr)
```

```
rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(rss2_lr)
metric3.append(rss2_lr)

mse_train_lr = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric3.append(mse_train_lr**0.5)

mse_test_lr = mean_squared_error(y_test, y_pred_test)
print(mse_test_lr)
metric3.append(mse_test_lr**0.5)

# lasso model parameters
model_parameters = list(lasso3.coef_ )
model_parameters.insert(0, lasso3.intercept_)
model_parameters = [round(x, 3) for x in model_parameters]
cols = X_train1.columns
cols = cols.insert(0, "constant")
list(zip(cols, model_parameters))

mod = list(zip(cols, model_parameters))
para = pd.DataFrame(mod)
para.columns = ['Variable', 'Coeff']
# sort the coefficients in ascending order
para = para.sort_values((['Coeff']), axis = 0, ascending = False)
# Chose variables whose coefficients are non-zero
pred = pd.DataFrame(para[(para['Coeff'] != 0)])
pred
```

```
Fitting 5 folds for each of 28 candidates, totalling 140 fits
0.9403135645489316
0.8707219380952721
9.593132580343248
9.316885188199825
0.009386626790942513
0.021271427370319236
```

:[534]:

	Variable	Coeff
0	constant	11.449
8	1st_flr_sf	0.246
108	roof_matl_WdShngl	0.234
253	full_bath_3	0.227
10	gr_liv_area	0.216
96	overall_cond_9	0.183

Now , the 5 most important predictors are :

1.1st_flr_sf

2.roof_matl_WdShngl

3.full_bath_3

4.gr_liv_area

5.overall_cond_9

Q.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 : The model should be generalized so that the test accuracy should not be much lesser compared to the training score. The model should be accurate for test datasets and so it should be generic in nature . Too much importance should not given to the outliers so that the accuracy predicted by the model is high. To ensure that this should not happen , 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 handled or removed from the dataset.

If the model is not robust, It cannot be trusted for predictive analysis.