Subjective Questions & Answers – Advanced Regression Assignment

Please note that all these questions-answers are already written in jupyter notebook as some answers also include code snippets, hence screenshots have been taken from the jupyter notebook and pasted here for reference.

Subjective Questions & Answers

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

Optimum alpha for Ridge = 10.0
Optimum alpha for Lasso = 0.001

Training and Test scores for Ridge with alpha = 10.0
R2 score (train) : 0.9166751151617185
R2 score (test) : 0.8704201226046137
RMSE (train) : 0.11304414693007461
RMSE (test) : 0.15390088041290248

Training and Test scores for Ridge with alpha = 20.0
R2 score (train) : 0.9165
R2 score (test) : 0.8709
RMSE (train) : 0.1132
RMSE (test) : 0.1536

Training and Test scores for Lasso with alpha = 0.001
R2 score (train) : 0.9157339730212566
R2 score (test) : 0.8745163217343286
RMSE (train) : 0.1136807627429514
RMSE (test) : 0.15144883684391255

Training and Test scores for Lasso with alpha = 0.002
R2 score (train) : 0.9146
R2 score (test) : 0.8759
RMSE (train) : 0.1144
RMSE (test) : 0.1506

As we see, there is not significant change in R2 score and RMSE score for Ridge regression when value of alpha is doubled.
Similarly, there is not significant change in R2 score and RMSE score for Lasso regression when value of alpha is doubled.

The top 5 features are : 1stFlrSF (Positive Relation with target variable) 2ndFlrSF (Positive Relation with target variable) OverallQual (Positive Relation with target variable) OverallCond (Positive Relation with target variable) YearBuilt (Negative Relation with target variable)

The top 5 important variables continue to remain the same for both, Lasso and Ridge regression, even though we may see a very very small dip in the coefficient value as they move a bit closer to 0.
For example, For Ridge regression with alpha =10, 1stFlrSF values goes from 0.125 to 0.122 for alpha=20.0
Similarly, For Lasso regression with alpha=0.001, 1stFlrSF values goes from 0.126 to 0.124. for alpha=0.002

This explains that as the value of alpha increases, the penalty term will be impacted and coefficeints slowly tend to reach close to zero and model starts to become less complex. if value of alpha becomes too high, then model would infact start underfitting and give a low training score.

In [1010]:

```
# Model Building
ridge_model = Ridge(alpha=20.0)
ridge_model.fit(X_train_rfe, y_train)

# Predicting
y_train_pred = ridge_model.predict(X_train_rfe)
y_test_pred = ridge_model.predict(X_test_rfe)

print("Model Evaluation : Ridge Regression, alpha=20.0")
print('R2 score (train) : ',round(r2_score(y_train,y_train_pred), 4))
print('R2 score (test) : ',round(r2_score(y_test,y_test_pred), 4))
print('RMSE (train) : ', round(np.sqrt(mean_squared_error(y_train, y_train_pred)), 4))
print('RMSE (test) : ', round(np.sqrt(mean_squared_error(y_test, y_test_pred)), 4))
```

```
Model Evaluation : Ridge Regression, alpha=20.0
R2 score (train) :  0.9165
R2 score (test) :  0.8709
RMSE (train) :  0.1132
RMSE (test) :  0.1536
```

In [1011]:
```
lasso_model = Lasso(alpha=0.002)
lasso_model.fit(X_train_rfe, y_train)
y_train_pred = lasso_model.predict(X_train_rfe)
y_test_pred = lasso_model.predict(X_test_rfe)

print("Model Evaluation : Lasso Regression, alpha=0.002")
print('R2 score (train) : ',round(r2_score(y_train,y_train_pred), 4))
print('R2 score (test) : ',round(r2_score(y_test,y_test_pred), 4))
print('RMSE (train) : ', round(np.sqrt(mean_squared_error(y_train, y_train_pred)), 4))
print('RMSE (test) : ', round(np.sqrt(mean_squared_error(y_test, y_test_pred)), 4))
```

```
Model Evaluation : Lasso Regression, alpha=0.002
R2 score (train) :  0.9146
R2 score (test) :  0.8759
RMSE (train) :  0.1144
RMSE (test) :  0.1506
```

In [1012]:
```
model_coefficients['Ridge (alpha = 20.0)'] = ridge_model.coef_
model_coefficients['Lasso (alpha = 0.002)'] = lasso_model.coef_
pd.set_option('display.max_rows', None)
model_coefficients
```

Out[1012]:

	Ridge (alpha=10.0)	Lasso (alpha=0.001)	Ridge (alpha = 20.0)	Lasso (alpha = 0.002)
LotFrontage	0.007800	0.005624	0.008157	0.004615
LotArea	0.030600	0.031021	0.030998	0.031297
LandSlope	0.009797	0.009664	0.009740	0.008895
OverallQual	0.078235	0.080717	0.078339	0.083190
OverallCond	0.048387	0.049037	0.047556	0.047834
YearBuilt	-0.040842	-0.041477	-0.038739	-0.039811
BsmtQual	0.022767	0.023370	0.022967	0.024329
BsmtExposure	0.009889	0.009376	0.009926	0.008211
BsmtFinSF1	0.025977	0.026318	0.026145	0.026647
HeatingQC	0.014794	0.014940	0.015158	0.015546
CentralAir	0.011926	0.010310	0.012012	0.009219
1stFlrSF	0.125737	0.126662	0.122502	0.124616
2ndFlrSF	0.106067	0.105968	0.103016	0.102877
BsmtFullBath	0.018309	0.016930	0.017786	0.015675
HalfBath	0.007532	0.006656	0.008406	0.006544
KitchenQual	0.014826	0.014684	0.015665	0.015467

KitchenQual	0.014826	0.014684	0.015665	0.015467
Functional	-0.026196	-0.025267	-0.025716	-0.024064
Fireplaces	0.021436	0.021161	0.022011	0.021051
GarageFinish	0.011738	0.010284	0.011774	0.009884
GarageArea	0.020937	0.021494	0.022175	0.023466
GarageQual	0.015475	0.006407	0.013933	0.008162
OpenPorchSF	0.008934	0.007928	0.009235	0.007096
MSZoning_RL	0.027728	0.026250	0.027386	0.024460
Street_Pave	0.009121	0.008553	0.009341	0.008293
LotConfig_CulDSac	0.008891	0.007908	0.009030	0.006818
Neighborhood_Edwards	-0.015200	-0.013074	-0.014979	-0.011529
Neighborhood_NAmes	-0.009540	-0.006864	-0.009623	-0.004887
Neighborhood_NridgHt	0.015024	0.013960	0.014837	0.012931
Neighborhood_Somerst	0.022859	0.022008	0.022545	0.020652
Condition1_Feedr	0.011705	0.009474	0.011172	0.006807
Condition1_Norm	0.025519	0.023238	0.024816	0.020406
Condition2_Norm	0.008702	0.008009	0.008730	0.007449
RoofStyle_Gable	-0.022221	-0.006335	-0.019641	-0.003485
RoofStyle_Hip	-0.017955	-0.002035	-0.015043	-0.000000
Exterior1st_HdBoard	-0.019540	-0.009784	-0.018266	-0.007540
Exterior1st_Plywood	-0.007230	-0.004658	-0.006763	-0.002466
Exterior1st_Wd Sdng	-0.018363	-0.010931	-0.017754	-0.005690
Exterior2nd_HdBoard	0.008569	0.000000	0.007360	-0.000000
Exterior2nd_Wd Sdng	0.012820	0.004839	0.012002	0.000000
MasVnrType_BrkFace	0.013389	-0.000000	0.010436	-0.000000
MasVnrType_None	0.014021	0.000000	0.010554	0.000000
MasVnrType_Stone	0.011333	0.002048	0.009556	0.001420
Foundation_PConc	0.018365	0.019212	0.018938	0.019558
Heating_GasA	-0.008553	-0.006935	-0.008651	-0.005566
GarageType_Attchd	0.014757	0.008778	0.013637	0.003396
GarageType_Detchd	0.016613	0.009228	0.014768	0.002375
GarageType_Not_applicable	0.012381	0.000000	0.010485	-0.000000

GarageType_Detchd	0.016613	0.009228	0.014768	0.002375
GarageType_Not_applicable	0.012381	0.000000	0.010485	-0.000000

PavedDrive_Y	0.009950	0.008313	0.010009	0.007205
SaleCondition_Normal	0.029024	0.028205	0.028394	0.026453
SaleCondition_Partial	0.034283	0.033529	0.033619	0.031926

In [1013]:
```
model_coefficients.sort_values(by='Lasso (alpha = 0.002)', ascending=False).head(4)
```

Out[1013]:

	Ridge (alpha=10.0)	Lasso (alpha=0.001)	Ridge (alpha = 20.0)	Lasso (alpha = 0.002)
1stFlrSF	0.125737	0.126662	0.122502	0.124616
2ndFlrSF	0.106067	0.105968	0.103016	0.102877
OverallQual	0.078235	0.080717	0.078339	0.083190
OverallCond	0.048387	0.049037	0.047556	0.047834

In [1014]:
```
model_coefficients.sort_values(by='Ridge (alpha = 20.0)', ascending=False).head(4)
```

Out[1014]:

	Ridge (alpha=10.0)	Lasso (alpha=0.001)	Ridge (alpha = 20.0)	Lasso (alpha = 0.002)
1stFlrSF	0.125737	0.126662	0.122502	0.124616
2ndFlrSF	0.106067	0.105968	0.103016	0.102877
OverallQual	0.078235	0.080717	0.078339	0.083190
OverallCond	0.048387	0.049037	0.047556	0.047834

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

Optimum alpha for Ridge = 10.0
Optimum alpha for Lasso = 0.001

Training and Test scores for Ridge with alpha = 10.0
R2 score (train) : 0.9166751151617185
R2 score (test) : 0.8704201226046137
RMSE (train) : 0.11304414693007461
RMSE (test) : 0.15390088041290248

Training and Test scores for Lasso with alpha = 0.001
R2 score (train) : 0.9157339730212566
R2 score (test) : 0.8745163217343286
RMSE (train) : 0.1136807627429514
RMSE (test) : 0.15144883684391255

Even though the R2 scores and RMSE scores (the evaluation metrics) are similar for both Ridge and Lasso, if we compare them precisely side by side, Lasso provides better R2 score on Test set and a lower RMSE score on Test set, although the difference is not by huge margin compared to Ridge.

SO in this particular case, we have opted to go with Lasso regression.

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

Before dropping the Top 5 variables, the Top 5 variables are 1stFlrSF (Positive Relation with target variable) 2ndFlrSF (Positive Relation with target variable) OverallQual (Positive Relation with target variable) OverallCond (Positive Relation with target variable) YearBuilt (Negative Relation with target variable)

After dropping the Top 5 variables, the Top 5 variables are GarageArea, KitchenQual, Fireplaces, LotArea, BsmtQual. All have a positive relation with target variable as the coefficient values are positive.

In [1015]:
```
X_train_new = X_train_rfe.drop(['1stFlrSF', '2ndFlrSF', 'OverallQual', 'OverallCond', 'YearBuilt'], axis=1)
```

In [1016]:
```
X_test_new = X_test_rfe.drop(['1stFlrSF', '2ndFlrSF', 'OverallQual', 'OverallCond', 'YearBuilt'], axis=1)
```

In [1017]:
```
alpha = 0.001
lasso_model = Lasso(alpha=alpha)
lasso_model.fit(X_train_new, y_train)
y_train_pred = lasso_model.predict(X_train_new)
y_test_pred = lasso_model.predict(X_test_new)
```

In [1018]:
```
lasso_model.coef_
```

Out[1018]:
```
array([ 0.01863706,  0.06080904,  0.00570992,  0.04216284,  0.00799507,
        0.03699888,  0.03120818,  0.02010528, -0.        ,  0.03887738,
        0.06236488, -0.02065214,  0.06086222,  0.01428569,  0.07484103,
        0.        ,  0.02321138,  0.02256109,  0.01959409,  0.01043322,
       -0.02107716, -0.0222155 ,  0.01360233,  0.01788036,  0.        ,
        0.01210415,  0.00415945, -0.02195281,  0.        , -0.02014932,
       -0.00103924, -0.02499817,  0.00454838,  0.02125675,  0.01635029,
       -0.        ,  0.01403271,  0.02827423, -0.01819907,  0.00315015,
       -0.0064887 ,  0.00490023,  0.00991415,  0.02564046,  0.03139158])
```

In [1019]:
```
model_coeff = pd.DataFrame(index=X_test_new.columns)
model_coeff.rows = X_test_new.columns
model_coeff['Lasso'] = lasso_model.coef_
model_coeff.sort_values(by='Lasso', ascending=False).head(45)
```

Out[1019]:

	Lasso
GarageArea	0.074841
KitchenQual	0.062365
Fireplaces	0.060862
LotArea	0.060809
BsmtQual	0.042163
HalfBath	0.038877
BsmtFinSF1	0.036999
SaleCondition_Partial	0.031392
HeatingQC	0.031208
Foundation_PConc	0.028274
SaleCondition_Normal	0.025640
OpenPorchSF	0.023211
MSZoning_RL	0.022561
Exterior2nd_Wd Sdng	0.021257
CentralAir	0.020105
Street_Pave	0.019594
LotFrontage	0.018637
Neighborhood_Somerst	0.017880
MasVnrType_BrkFace	0.016350
GarageFinish	0.014286
MasVnrType_Stone	0.014033
Neighborhood_NridgHt	0.013602
Condition1_Norm	0.012104
LotConfig_CulDSac	0.010433
PavedDrive_Y	0.009914
BsmtExposure	0.007995
LandSlope	0.005710
GarageType_Not_applicable	0.004900
Exterior2nd_HdBoard	0.004548
Condition2_Norm	0.004159
GarageType_Attchd	0.003150
BsmtFullBath	-0.000000
MasVnrType_None	-0.000000
GarageQual	0.000000
RoofStyle_Hip	0.000000
Condition1_Feedr	0.000000
Exterior1st_Plywood	-0.001039
GarageType_Detchd	-0.006489
Heating_GasA	-0.018199
Exterior1st_HdBoard	-0.020149
Functional	-0.020652
Neighborhood_Edwards	-0.021077
RoofStyle_Gable	-0.021953
Neighborhood_NAmes	-0.022216
Exterior1st_Wd Sdng	-0.024998

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.
Bias-variance tradeoff - If our model is too simple and has very few parameters then it may have high bias and low variance. On the other hand if our model has large number of parameters then it's going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data.

If a model is too complex, it will have low bias and high variance. But as it has overfitted the training data, model will give high accuracy on training dataset, but is more likely to perform poorly in unseen test dataset.

If a model is too simple, it will have high bias and low variance. As it is too simple, it will fail to identify the underlying patterns in the data, and as aresult it will high a low training score as well as test score.

If we take the point in the bais variance trade off graph, where both intersect each other, that point will give perfect balance between bias-variance. It will ensure that model does not overfit while still having good variance.