Assignment Advance Regression (Assignment Part-II)

Ramandeep Mehra
upGrad and IIITB Machine Learning & AI Program-Dec 2023

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 1: Optimal value of alpha:

1. For Ridge regression :1.0
2. For Lasso Regression :0.001

Ridge Regression:

- **Higher Alpha, Stronger Regularization:** Increasing alpha from 1 to 2 strengthens the regularization penalty in ridge regression. This forces the model to prioritize simpler explanations by shrinking the coefficients of all features more aggressively.

- **Trading Off Fit for Generalizability:** While a higher alpha reduces variance (makes the model less likely to overfit), it can also increase bias (make the model underfit).

Lasso Regression:

- **Sparsity with Higher Alpha:** Unlike ridge regression, lasso performs variable selection. Doubling alpha encourages more coefficients to shrink to zero, effectively removing less important features from the model.

- **Interpretability vs. Performance:** This creates a sparser, easier-to-interpret model, but overly aggressive shrinkage (high alpha) can harm prediction accuracy.

Summary:

- Doubling alpha affects ridge and lasso differently. Both experience stronger regularization, but lasso also benefits from feature selection.

- Choosing the optimal alpha requires evaluating the trade-off between model complexity and performance (often through techniques like cross-validation).

Model	Effect of Doubling Alpha
Ridge Regression	Increased regularization, reduced variance (potentially lower bias if alpha is chosen carefully)
Lasso Regression	More feature sparsity, increased interpretability (potential performance impact if alpha is too high)

For Ridge Regression: The most important variable after the changes has been implemented for ridge regression are as follows:-

1. OverallCond
2. 2ndFlrSF
3. BsmtFullBath
4. LotFrontage
5. GarageQual
6. OverallQual
7. BsmtExposure
8. KitchenQual
9. ExterQual
10. BsmtCond

```
# Model building using optimal alpha
ridge_modified = Ridge(alpha=2.0)
ridge_modified.fit(X_train, y_train)
```

```
▾ Ridge
Ridge(alpha=2.0)
```

```
#creating coeffcients for the ridge regression
model_parameter = list(ridge.coef_)
model_parameter.insert(0,ridge.intercept_)
cols = X_train.columns
cols.insert(0,'const')
ridge_coef = pd.DataFrame(list(zip(cols,model_parameter,(abs(ele) for ele in model_parameter))))
ridge_coef.columns = ['Features','Coefficient','Mod']
#selecting the top 10 variables
ridge_coef.sort_values(by='Mod',ascending=False).head(10)
```

	Features	Coefficient	Mod
3	OverallCond	106002.939206	106002.939206
19	2ndFlrSF	97870.113437	97870.113437
22	BsmtFullBath	92888.078281	92888.078281
0	LotFrontage	-69841.617334	69841.617334
34	GarageQual	56761.400063	56761.400063
2	OverallQual	55981.320161	55981.320161
9	BsmtExposure	-43193.920218	43193.920218
28	KitchenQual	-41174.077080	41174.077080
5	ExterQual	40335.807321	40335.807321
8	BsmtCond	39151.848239	39151.848239

```
y_train_pred = ridge_modified.predict(X_train)
y_test_pred = ridge_modified.predict(X_test)

print("Ridge Regression train r2:",r2_score(y_true=y_train,y_pred=y_train_pred))
print("Ridge Regression test r2:",r2_score(y_true=y_test,y_pred=y_test_pred))
```

```
Ridge Regression train r2: 0.8154593239001291
Ridge Regression test r2: 0.8266638959039371
```

For Lasso Regression: The most important variable after the changes has been implemented for lasso regression are as follows:-

1. 2ndFlrSF
2. OverallCond
3. OverallQual
4. LotFrontage

5. LowQualFinSF
6. GarageQual
7. PavedDrive
8. KitchenQual
9. BsmtCond
10. BsmtExposure

```
[118] # Model building using optimal alpha
      lasso_modified = Lasso(alpha=0.002)
      lasso_modified.fit(X_train, y_train)
```

```
▾     Lasso
Lasso(alpha=0.002)
```

```
[119] y_train_pred = lasso_modified.predict(X_train)
      y_test_pred = lasso_modified.predict(X_test)

      print("Lasso Regression train r2:",r2_score(y_true=y_train,y_pred=y_train_pred))
      print("Lasso Regression test r2:",r2_score(y_true=y_test,y_pred=y_test_pred))
```

```
Lasso Regression train r2: 0.8187298005750374
Lasso Regression test r2: 0.8267432687133583
```

```
[120] model_param = list(lasso.coef_)
      model_param.insert(0,lasso.intercept_)
      cols = X_train.columns
      cols.insert(0,'const')
      lasso_coef = pd.DataFrame(list(zip(cols,model_param,(abs(ele) for ele in model_param))))
      lasso_coef.columns = ['Feature','Coef','mod']
```

```
#selecting the top 10 variables
lasso_coef.sort_values(by='mod',ascending=False).head(10)
```

	Feature	Coef	mod
19	2ndFlrSF	213152.782140	213152.782140
3	OverallCond	111663.259873	111663.259873
2	OverallQual	80791.120892	80791.120892
0	LotFrontage	-78582.472463	78582.472463
20	LowQualFinSF	70975.025631	70975.025631
34	GarageQual	58082.388105	58082.388105
36	PavedDrive	-50823.972524	50823.972524
28	KitchenQual	-47237.936412	47237.936412
8	BsmtCond	46944.549332	46944.549332
9	BsmtExposure	-45164.650739	45164.650739

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 2: It is important to regularize coefficients and improve the prediction accuracy also with the decrease in variance, and making the model interpretable.

Ridge Regression: Shrinking Coefficients for Stability

- Ridge regression uses a parameter "lambda" to control the amount of regularization.
- It shrinks the values of all the model's coefficients (like weights) a little bit. This makes the model simpler and less prone to overfitting.
- Increasing lambda reduces variance but can also introduce some bias (underfitting).
- Unlike lasso regression, ridge regression typically includes all features in the final model.

Lasso Regression: Shrinking and Selecting Features

- Lasso regression also uses lambda for regularization, but in a different way.
- It shrinks coefficients towards zero, and some coefficients can even become exactly zero! This effectively removes unimportant features from the model.
- This makes the model easier to interpret, as you can focus on the remaining features that have a significant impact.
- With a small lambda, lasso acts like regular linear regression. Increasing lambda leads to shrinkage and feature selection.

Hence, We will make use of the **Lasso Regression** model because it is using fewer numbers of variables and giving almost the same accuracy. Its more efficient model than Ridge regression model.

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 3: Those 5 most important predictor variables that will be excluded are:

1. BsmtFullBath
2. GrLivArea
3. MasVnrArea
4. GarageQual

5. PavedDrive

Question: Double the alpha values and evaluate model

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?

Drooping the first five important predictors

```
[122] X_train_new = X_train.drop(["2ndFlrSF" ,  "OverallCond" ,  "OverallQual" ,  "LotFrontage" ,  "LowQualFinSF"],axis=1)
      X_test_new = X_test.drop(["2ndFlrSF" ,  "OverallCond" ,  "OverallQual" ,  "LotFrontage" ,  "LowQualFinSF"],axis=1)

      X_test_new.head()
      X_train_new.shape
```

```
(1021, 41)
```

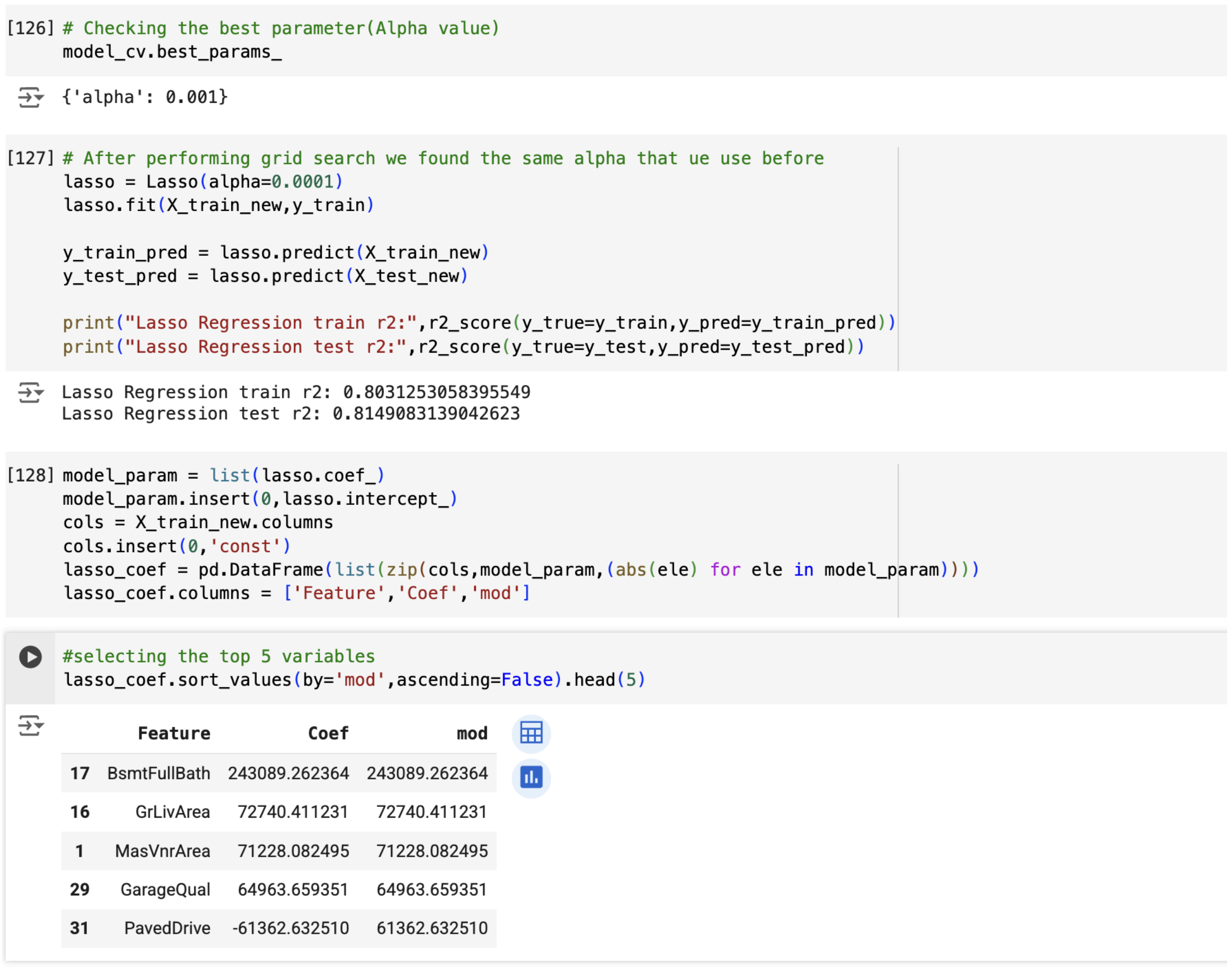
```
[123] X_test_new.shape
```

```
(438, 41)
```

```
lasso_modified = Lasso()
param = {'alpha': [0.0001, 0.001, 0.01]}
folds = 5
# cross validation
lasso_cv_model_modified = GridSearchCV(estimator = lasso,
                        param_grid = param,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)

lasso_cv_model_modified.fit(X_train_new, y_train)
```

```
Fitting 5 folds for each of 3 candidates, totalling 15 fits
```



```
[126] # Checking the best parameter(Alpha value)
      model_cv.best_params_
```

```
{'alpha': 0.001}
```

```
[127] # After performing grid search we found the same alpha that ue use before
      lasso = Lasso(alpha=0.0001)
      lasso.fit(X_train_new,y_train)

      y_train_pred = lasso.predict(X_train_new)
      y_test_pred = lasso.predict(X_test_new)

      print("Lasso Regression train r2:",r2_score(y_true=y_train,y_pred=y_train_pred))
      print("Lasso Regression test r2:",r2_score(y_true=y_test,y_pred=y_test_pred))
```

```
Lasso Regression train r2: 0.8031253058395549
Lasso Regression test r2: 0.8149083139042623
```

```
[128] model_param = list(lasso.coef_)
      model_param.insert(0,lasso.intercept_)
      cols = X_train_new.columns
      cols.insert(0,'const')
      lasso_coef = pd.DataFrame(list(zip(cols,model_param,(abs(ele) for ele in model_param))))
      lasso_coef.columns = ['Feature','Coef','mod']
```

```
#selecting the top 5 variables
lasso_coef.sort_values(by='mod',ascending=False).head(5)
```

	Feature	Coef	mod
17	BsmtFullBath	243089.262364	243089.262364
16	GrLivArea	72740.411231	72740.411231
1	MasVnrArea	71228.082495	71228.082495
29	GarageQual	64963.659351	64963.659351
31	PavedDrive	-61362.632510	61362.632510

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 4: The model should be as simple as possible, though its accuracy will decrease but it will be more robust and generalisable. It can be also understood using the Bias-Variance trade-off. The simpler the model the more the bias but less variance and more generalizable. Its implication in terms of accuracy is that a robust and generalisable model will perform equally well on both training and test data i.e. the accuracy does not change much for training and test data.

Bias: Bias is an error in a model, when the model is weak to learn from the data. High bias means the model is unable to learn details in the data. Model performs poorly on training and testing data.

Variance: Variance is error in model, when model tries to over learn from the data. High variance means the model performs exceptionally well on training data as it has very well trained on this data but performs very poorly on testing data as it was unseen data for the model.

It is important to have balance in Bias and Variance to avoid overfitting and under-fitting of data.