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:

Ridge Alpha :2

lasso Alpha :50

```
[139] #Increase the alpha value from 2 to 3
```

```
[140] alpha = 3
      ridge2 = Ridge(alpha=alpha)
      ridge2.fit(X_train1, y_train)
```

```
Ridge(alpha=3)
```

```
[141] y_pred_train = ridge2.predict(X_train1)
      y_pred_test = ridge2.predict(X_test1)

      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)
```

```
0.9254673600172272
0.9202956337084576
376786221298.1054
198333265627.28186
421933058.5645077
450757421.880186
```

```
[143] #Lasso
      #changing the value from 50-100
      alpha =100
      lasso100 = Lasso(alpha=alpha)
      lasso100.fit(X_train1, y_train)
```

```
Lasso(alpha=100)
```

```
#Let's compute a few measures, including the R2 score, RSS, and RMSE.
y_pred_train = lasso100.predict(X_train1)
y_pred_test = lasso100.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)
```

```
0.9208779766398706
0.9206768502085073
399987015221.9784
197384661217.21353
447913790.84208107
448601502.7663944
```

```
[147] #R2 score of training data has decreased, whereas R2 score of testing data has increased.
      betas = pd.DataFrame(index=X_train1.columns)
      betas.rows = X_train1.columns
      betas['Ridge2'] = ridge2.coef_
      betas['Ridge'] = ridge.coef_
      betas['Lasso'] = lasso.coef_
      betas['Lasso100'] = lasso100.coef_
      pd.set_option('display.max_rows', None)
      betas.head(68)

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: Pandas doesn't allow c
  This is separate from the ipykernel package so we can avoid doing imports until
```

	Ridge2	Ridge	Lasso	Lasso100
MSSubClass	-19490.421039	-20369.740130	-21224.762957	-20077.813080
LotFrontage	11290.725246	11428.200578	8231.325034	5061.382271
LotArea	27093.577925	29774.295463	30517.460012	24462.786082
OverallQual	61751.532794	62715.839184	72557.554506	80806.220610
OverallCond	33021.185764	35587.079466	38896.865888	34497.673017
YearBuilt	37297.336110	40598.381481	45380.369624	41082.159642
MasVnrArea	22368.573661	21973.625990	18034.586738	17159.190131
BsmtFinSF1	53345.176831	54117.582345	46050.401802	46482.028656
BsmtFinSF2	8063.220163	8944.886849	2912.767910	384.158377
BsmtUnfSF	8613.257148	8894.164866	0.000000	0.000000
TotalBsmtSF	44961.039635	45942.017088	53247.157269	46785.928486
1stFlrSF	60524.182753	63864.444649	5879.750523	11348.172096
2ndFlrSF	35986.354416	37575.532049	0.000000	0.000000
LowQualFinSF	0.000000	0.000000	0.000000	0.000000

- LotArea
- OverallQual
- OverallCond
- YearBuilt
- BsmtFinSF1
- TotalBsmtSF
- GrLivArea
- TotRmsAbvGrdStreet_Pave
- RoofMatl_Metal

Predictors are the same, but their coefficacy has altered.

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: We will use lasso regression to address this issue because its r2 score is marginally greater than that of lasso for the test dataset.

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:

```
X_train1.columns
```

```
Index(['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond',
       'YearBuilt', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF',
       ...
       'GarageCond_Po', 'GarageCond_TA', 'PavedDrive_P', 'SaleType_CWD',
       'SaleType_Con', 'SaleType_ConLD', 'SaleType_New', 'SaleType_Oth',
       'SaleCondition_Alloca', 'SaleCondition_Family'],
      dtype='object', length=110)
```

LotFrontage , LotArea,OverallQUal,YearBuilt,BsmtFinSF1 are the 5 most important predictors.

```
#Lasso
# alpha 50
alpha =50
lasso101 = Lasso(alpha=alpha)
lasso101.fit(X_train2, y_train)
```

```
Lasso(alpha=50)
```

```
y_pred_train = lasso101.predict(X_train2)
y_pred_test = lasso101.predict(X_test2)

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)
```

```
0.9151697786963947
0.9098253048604359
428843772932.9759
224387227401.77045
480228189.17466503
509970971.3676601
```

Data from training and testing have a lower R2 value.

```
] betas = pd.DataFrame(index=X_train2.columns)
  betas.rows = X_train1.columns
  betas['Lasso101'] = lasso101.coef_
  pd.set_option('display.max_rows', None)
  betas.head(68)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: UserWarning:
```

	Lasso101
MSSubClass	-20523.102184
OverallCond	30879.109125
MasVnrArea	19448.938215
BsmtFinSF2	-11965.171775
BsmtUnfSF	-47136.780397
TotalBsmtSF	137826.130360
1stFlrSF	0.000000
2ndFlrSF	0.000000
LowQualFinSF	0.000000
GrLivArea	187648.608955
BedroomAbvGr	-31928.634265
KitchenAbvGr	0.000000
TotRmsAbvGrd	22566.495470
GarageCars	44144.233790

5 most important predictors

- MasVnrArea
- BsmtFinSF2
- BsmtUnfSF
- TotalBsmtSF
- 1stFlrSF

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 needs to be generalised to ensure that test accuracy does not fall short of training results. For datasets other than the ones that were used during training, the model should be accurate. The outliers shouldn't be given an excessive amount of weight in order to maintain the high level of model accuracy. Only those outliers that are significant to the dataset should be preserved after conducting the outliers analysis to verify that this is not the case. The dataset must be cleaned up of any outliers that don't make sense to preserve. Predictive analysis cannot be believed if the model is not robust.