1.Lasso and Ridge Regresssion

October 28, 2021

Regularization

[1]:
```
# Import necessary package
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

0.0.1 Step 1: Load the dataset

[2]:
```
# Load dataset into pandas dataframe
df=pd.read_csv("E:\\MY LECTURES\\DATA SCIENCE\\3.
 ↪Programs\\dataset\\Melbourne_housing_price.csv")
# Change this location based on the location of dataset in your machine
```

[3]:
```
# Display the first five records
df.head()
```

[3]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	\
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	

	Date	Distance	Postcode	…	Bathroom	Car	Landsize	BuildingArea	\
0	03-09-2016	2.5	3067.0	…	1.0	1.0	126.0	NaN	
1	03-12-2016	2.5	3067.0	…	1.0	1.0	202.0	NaN	
2	04-02-2016	2.5	3067.0	…	1.0	0.0	156.0	79.0	
3	04-02-2016	2.5	3067.0	…	2.0	1.0	0.0	NaN	
4	04-03-2017	2.5	3067.0	…	2.0	0.0	134.0	150.0	

	YearBuilt	CouncilArea	Lattitude	Longtitude	Regionname	\
0	NaN	Yarra City Council	-37.8014	144.9958	Northern Metropolitan	
1	NaN	Yarra City Council	-37.7996	144.9984	Northern Metropolitan	
2	1900.0	Yarra City Council	-37.8079	144.9934	Northern Metropolitan	
3	NaN	Yarra City Council	-37.8114	145.0116	Northern Metropolitan	

```
4     1900.0  Yarra City Council  -37.8093    144.9944  Northern Metropolitan

  Propertycount
0        4019.0
1        4019.0
2        4019.0
3        4019.0
4        4019.0

[5 rows x 21 columns]
```

[4]:
```
# Dataset shape (number of rows and columns)
df.shape
```

[4]:
```
(34857, 21)
```

0.0.2 Step 2: Apply EDA

You may apply univariate and bivariate analysis

0.0.3 Step 3. Pre-process and extract the features

Unique values in the dataset

[5]:
```
df.nunique()
```

[5]:
```
Suburb            351
Address         34009
Rooms              12
Type                3
Price            2871
Method              9
SellerG           388
Date               78
Distance          215
Postcode          211
Bedroom2           15
Bathroom           11
Car                15
Landsize         1684
BuildingArea      740
YearBuilt         160
CouncilArea        33
Lattitude       13402
Longtitude      14524
Regionname          8
```

```
Propertycount      342
dtype: int64
```

Filter the columns

[6]:
```
cols_to_use =␣
 ↪['Suburb','Rooms','Type','Method','SellerG','Regionname','Propertycount','Distance','Counci
df = df[cols_to_use]
df.head()
```

[6]:

	Suburb	Rooms	Type	Method	SellerG	Regionname	\
0	Abbotsford	2	h	SS	Jellis	Northern Metropolitan	
1	Abbotsford	2	h	S	Biggin	Northern Metropolitan	
2	Abbotsford	2	h	S	Biggin	Northern Metropolitan	
3	Abbotsford	3	u	VB	Rounds	Northern Metropolitan	
4	Abbotsford	3	h	SP	Biggin	Northern Metropolitan	

	Propertycount	Distance	CouncilArea	Bedroom2	Bathroom	Car	\
0	4019.0	2.5	Yarra City Council	2.0	1.0	1.0	
1	4019.0	2.5	Yarra City Council	2.0	1.0	1.0	
2	4019.0	2.5	Yarra City Council	2.0	1.0	0.0	
3	4019.0	2.5	Yarra City Council	3.0	2.0	1.0	
4	4019.0	2.5	Yarra City Council	3.0	2.0	0.0	

	Landsize	BuildingArea	Price
0	126.0	NaN	NaN
1	202.0	NaN	1480000.0
2	156.0	79.0	1035000.0
3	0.0	NaN	NaN
4	134.0	150.0	1465000.0

[7]:
```
df.shape
```

[7]:
```
(34857, 15)
```

How many 'NaN' available in the dataset?

[8]:
```
df.isna().sum()
```

[8]:
```
Suburb              0
Rooms               0
Type                0
Method              0
SellerG             0
Regionname          3
Propertycount       3
Distance            1
```

```
CouncilArea          3
Bedroom2          8217
Bathroom          8226
Car               8728
Landsize         11810
BuildingArea     21115
Price             7610
dtype: int64
```

Let us fill 0 for some NaN in the features: Propertycount, Distance, Bedroom2, Bathroom, and Car

[9]:
```
cols_to_fill_zero = ['Propertycount','Distance','Bedroom2','Bathroom','Car']
df[cols_to_fill_zero] = df[cols_to_fill_zero].fillna(0)
```

[10]:
```
df.isna().sum()
```

[10]:
```
Suburb               0
Rooms                0
Type                 0
Method               0
SellerG              0
Regionname           3
Propertycount        0
Distance             0
CouncilArea          3
Bedroom2             0
Bathroom             0
Car                  0
Landsize         11810
BuildingArea     21115
Price             7610
dtype: int64
```

Let us fill mean for some NaN in the features: Landsize, and BuildingArea

[11]:
```
df['Landsize'] = df['Landsize'].fillna(df.Landsize.mean())
df['BuildingArea'] = df['BuildingArea'].fillna(df.BuildingArea.mean())
```

[12]:
```
df.isna().sum()
```

[12]:
```
Suburb           0
Rooms            0
Type             0
Method           0
SellerG          0
```

```
Regionname          3
Propertycount       0
Distance            0
CouncilArea         3
Bedroom2            0
Bathroom            0
Car                 0
Landsize            0
BuildingArea        0
Price            7610
dtype: int64
```

Let us drop some records that contain NaN (possible when dataset is huge)

[13]:
```
df.dropna(inplace=True)
```

[14]:
```
df.isna().sum()
```

[14]:
```
Suburb          0
Rooms           0
Type            0
Method          0
SellerG         0
Regionname      0
Propertycount   0
Distance        0
CouncilArea     0
Bedroom2        0
Bathroom        0
Car             0
Landsize        0
BuildingArea    0
Price           0
dtype: int64
```

One hot encoding - replacing categorical values with numerical number - pre-processing technique

[15]:
```
df = pd.get_dummies(df, drop_first=True)
df.head()
```

[15]:

	Rooms	Propertycount	Distance	Bedroom2	Bathroom	Car	Landsize	\
1	2	4019.0	2.5	2.0	1.0	1.0	202.0	
2	2	4019.0	2.5	2.0	1.0	0.0	156.0	
4	3	4019.0	2.5	3.0	2.0	0.0	134.0	
5	3	4019.0	2.5	3.0	2.0	1.0	94.0	

```
6      4         4019.0       2.5       3.0       1.0  2.0     120.0

   BuildingArea      Price  Suburb_Aberfeldie  …  \
1      160.2564  1480000.0                  0  …
2       79.0000  1035000.0                  0  …
4      150.0000  1465000.0                  0  …
5      160.2564   850000.0                  0  …
6      142.0000  1600000.0                  0  …

   CouncilArea_Moorabool Shire Council  CouncilArea_Moreland City Council  \
1                                    0                                  0
2                                    0                                  0
4                                    0                                  0
5                                    0                                  0
6                                    0                                  0

   CouncilArea_Nillumbik Shire Council  CouncilArea_Port Phillip City Council  \
1                                    0                                     0
2                                    0                                     0
4                                    0                                     0
5                                    0                                     0
6                                    0                                     0

   CouncilArea_Stonnington City Council  CouncilArea_Whitehorse City Council  \
1                                     0                                   0
2                                     0                                   0
4                                     0                                   0
5                                     0                                   0
6                                     0                                   0

   CouncilArea_Whittlesea City Council  CouncilArea_Wyndham City Council  \
1                                    0                                 0
2                                    0                                 0
4                                    0                                 0
5                                    0                                 0
6                                    0                                 0

   CouncilArea_Yarra City Council  CouncilArea_Yarra Ranges Shire Council
1                               1                                      0
2                               1                                      0
4                               1                                      0
5                               1                                      0
6                               1                                      0

[5 rows x 745 columns]
```

[16]:
```
X = df.drop('Price', axis=1)
Y = df['Price']
```

input feature independent feature or predictor feature. All features except Price. output feature dependent feature or response feature or target feature. Price feature.

0.0.4 Step 4. Split the data for training and testing

[17]:
```
# Splitting dataset into training and testing set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2,␣
 ↪random_state = 2)
```

0.0.5 Step 5: Training phase (bulding the model)

1. Multiple Linear regression

[18]:
```
# Fitting line on two dimension on the training set
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train, y_train)
```

[18]:
```
LinearRegression()
```

[19]:
```
# R2 score for training data
linear_train_R2 = model.score(x_train, y_train)
linear_train_R2
```

[19]:
```
0.6792421760392956
```

[20]:
```
# R2 score for testing data
linear_test_R2 = model.score(x_test, y_test)
linear_test_R2
```

[20]:
```
0.6748321429524691
```

2. Lasso (L1) regression

[21]:
```
from sklearn import linear_model
lasso_model = linear_model.Lasso(alpha=50, max_iter=100, tol=0.1)
lasso_model.fit(x_train, y_train)
```

```
C:\Users\Rathinaraja Jeyaraj\anaconda3\lib\site-
packages\sklearn\linear_model\_coordinate_descent.py:529: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
```

```
Duality gap: 1505479056754043.2, tolerance: 899595916346279.9
  model = cd_fast.enet_coordinate_descent(
```

[21]:
```
Lasso(alpha=50, max_iter=100, tol=0.1)
```

[22]:
```
# R2 score for training data
lasso_train_R2 = lasso_model.score(x_train, y_train)
lasso_train_R2
```

[22]:
```
0.6750163113569139
```

[23]:
```
# R2 score for testing data
lasso_test_R2 = lasso_model.score(x_test, y_test)
lasso_test_R2
```

[23]:
```
0.6782522159085694
```

3. Ridge (L2) regression

[24]:
```
from sklearn.linear_model import Ridge
ridge_model = Ridge(alpha=50, max_iter=100, tol=0.1)
ridge_model.fit(x_train, y_train)
```

[24]:
```
Ridge(alpha=50, max_iter=100, tol=0.1)
```

[25]:
```
# R2 score for training data
ridge_train_R2 = ridge_model.score(x_train, y_train)
ridge_train_R2
```

[25]:
```
0.663472813002645
```

[26]:
```
# R2 score for testing data
ridge_test_R2 = ridge_model.score(x_test, y_test)
ridge_test_R2
```

[26]:
```
0.6715982779092569
```

0.0.6 Underfitting and overfitting observation

[27]:
```
print("Method \t R2_Taining   R2_Testing")
print("============================")
print("Linear  ",round(linear_train_R2,2)*100,"\t   ",␣
 ↪round(linear_test_R2,2)*100)
print("Lasso   ",round(lasso_train_R2,2)*100,"\t   ",␣
 ↪round(lasso_test_R2,2)*100)
```

```
print("Ridge   ",round(ridge_train_R2,2)*100,"\t   ",␣
 ↪round(ridge_test_R2,2)*100)
```

```
Method   R2_Taining   R2_Testing
============================
Linear   68.0        67.0
Lasso    68.0        68.0
Ridge    66.0        67.0
```