19MIS1018_ML_LAB-5_REGRESSION(Comparison)

August 23, 2022

Name: B DEVI PRASAD

Reg.No : 19MIS1018

Slot: L13+L14

Faculty: Dr. G. Bharadwaja Kumar

1 Linear, Lasso, and Ridge Regression with scikit-learn

Build, Predict and Evaluate the Regression Model , Training data,Definig the Data set

[1]:
```
import pandas as pd
import numpy as np
from sklearn import model_selection
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
```

[2]:
```
df = pd.read_csv('hungary_chickenpox.csv')
print(df.shape)
df.describe()
```

```
(522, 20)
```

[2]:

	India	china	BACS	BEKES	BORSOD	CSONGRAD	\
count	522.000000	522.000000	522.000000	522.000000	522.000000	522.000000	
mean	101.245211	34.204981	37.166667	28.911877	57.082375	31.488506	
std	76.354872	32.567222	36.843095	37.618092	50.725437	33.790208	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	

25%	34.250000	8.000000	8.000000	4.000000	14.250000	6.000000
50%	93.000000	25.000000	29.500000	14.000000	46.500000	20.500000
75%	149.000000	51.000000	53.000000	38.750000	83.750000	47.000000
max	479.000000	194.000000	274.000000	271.000000	355.000000	199.000000

	FEJER	GYOR	HAJDU	HEVES	JASZ	KOMAROM \
count	522.000000	522.000000	522.000000	522.000000	522.000000	522.000000
mean	33.272031	41.436782	47.097701	29.691571	40.869732	25.643678
std	31.397989	36.014297	44.610836	31.857750	37.283299	24.467995
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	7.000000	9.000000	11.000000	6.250000	10.000000	6.000000
50%	24.000000	35.000000	37.000000	21.000000	31.000000	19.000000
75%	51.750000	63.000000	68.000000	41.000000	61.750000	39.000000
max	164.000000	181.000000	262.000000	210.000000	224.000000	160.000000

	NOGRAD	PEST	SOMOGY	SZABOLCS	TOLNA	VAS \
count	522.000000	522.000000	522.000000	522.000000	522.000000	522.000000
mean	21.850575	86.101533	27.609195	29.854406	20.352490	22.467433
std	22.025999	66.773741	26.724236	31.814630	23.273025	25.006638
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	4.000000	28.250000	6.000000	6.000000	4.000000	3.000000
50%	15.000000	81.000000	20.500000	18.500000	12.000000	13.000000
75%	32.750000	129.750000	41.000000	45.000000	29.000000	34.000000
max	112.000000	431.000000	155.000000	203.000000	131.000000	141.000000

	VESZPREM	ZALA
count	522.000000	522.000000
mean	40.636015	19.873563
std	40.699471	21.999636
min	0.000000	0.000000
25%	7.250000	4.000000
50%	32.000000	13.000000
75%	59.000000	31.000000
max	230.000000	216.000000

[6]:
```
target_column = ['china','India','BACS']
```

[7]:
```
predictors = list(set(list(df.columns))-set(target_column))
df[predictors] = df[predictors]/df[predictors].max()
df.describe()
```

[7]:

	India	china	BACS	BEKES	BORSOD	CSONGRAD \
count	522.000000	522.000000	522.000000	522.000000	522.000000	522.000000
mean	0.211368	34.204981	37.166667	0.106686	0.160795	0.158234
std	0.159405	32.567222	36.843095	0.138812	0.142889	0.169800
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.071503	8.000000	8.000000	0.014760	0.040141	0.030151

```
50%     0.194154   25.000000   29.500000    0.051661    0.130986    0.103015
75%     0.311065   51.000000   53.000000    0.142989    0.235915    0.236181
max     1.000000  194.000000  274.000000    1.000000    1.000000    1.000000

            FEJER        GYOR       HAJDU       HEVES        JASZ     KOMAROM  \
count  522.000000  522.000000  522.000000  522.000000  522.000000  522.000000
mean     0.202878    0.228932    0.179762    0.141388    0.182454    0.160273
std      0.191451    0.198974    0.170270    0.151704    0.166443    0.152925
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.042683    0.049724    0.041985    0.029762    0.044643    0.037500
50%      0.146341    0.193370    0.141221    0.100000    0.138393    0.118750
75%      0.315549    0.348066    0.259542    0.195238    0.275670    0.243750
max      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000

           NOGRAD        PEST      SOMOGY    SZABOLCS       TOLNA         VAS  \
count  522.000000  522.000000  522.000000  522.000000  522.000000  522.000000
mean     0.195094    0.199772    0.178124    0.147066    0.155363    0.159343
std      0.196661    0.154927    0.172414    0.156722    0.177657    0.177352
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.035714    0.065545    0.038710    0.029557    0.030534    0.021277
50%      0.133929    0.187935    0.132258    0.091133    0.091603    0.092199
75%      0.292411    0.301044    0.264516    0.221675    0.221374    0.241135
max      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000

         VESZPREM        ZALA
count  522.000000  522.000000
mean     0.176678    0.092007
std      0.176954    0.101850
min      0.000000    0.000000
25%      0.031522    0.018519
50%      0.139130    0.060185
75%      0.256522    0.143519
max      1.000000    1.000000
```

[8]:

```python
X = df[predictors].values
y = df[target_column].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
 ↪random_state=40)
print(X_train.shape); print(X_test.shape)
```

```
(365, 17)
(157, 17)
```

2 Linear Regression

[9]:
```
lr = LinearRegression()
lr.fit(X_train, y_train)
```

[9]: LinearRegression()

[10]:
```
pred_train_lr= lr.predict(X_train)
print(np.sqrt(mean_squared_error(y_train,pred_train_lr)))
print(r2_score(y_train, pred_train_lr))

pred_test_lr= lr.predict(X_test)
print(np.sqrt(mean_squared_error(y_test,pred_test_lr)))
print(r2_score(y_test, pred_test_lr))
```

```
18.908260071178237
0.6183584480300238
18.934877066381894
0.5745636063579794
```

3 Ridge Regression

[11]:
```
rr = Ridge(alpha=0.01)
rr.fit(X_train, y_train)
pred_train_rr= rr.predict(X_train)
print(np.sqrt(mean_squared_error(y_train,pred_train_rr)))
print(r2_score(y_train, pred_train_rr))

pred_test_rr= rr.predict(X_test)
print(np.sqrt(mean_squared_error(y_test,pred_test_rr)))
print(r2_score(y_test, pred_test_rr))
```

```
18.908273892085823
0.618357761314323
18.930003694792607
0.5748015225336987
```

4 Lasso Regression

[12]:
```
model_lasso = Lasso(alpha=0.01)
model_lasso.fit(X_train, y_train)
pred_train_lasso= model_lasso.predict(X_train)
print(np.sqrt(mean_squared_error(y_train,pred_train_lasso)))
print(r2_score(y_train, pred_train_lasso))

pred_test_lasso= model_lasso.predict(X_test)
print(np.sqrt(mean_squared_error(y_test,pred_test_lasso)))
```

```
print(r2_score(y_test, pred_test_lasso))
```

```
18.91066204969824
0.5404051324084427
18.92519891425323
0.5200367203638928
```

5 ElasticNet Regression

[13]:
```
model_enet = ElasticNet(alpha = 0.01)
model_enet.fit(X_train, y_train)
pred_train_enet= model_enet.predict(X_train)
print(np.sqrt(mean_squared_error(y_train,pred_train_enet)))
print(r2_score(y_train, pred_train_enet))

pred_test_enet= model_enet.predict(X_test)
print(np.sqrt(mean_squared_error(y_test,pred_test_enet)))
print(r2_score(y_test, pred_test_enet))
```

```
19.10944239477544
0.5830439917225758
18.540644854700595
0.5805164870884001
```

Comparision: Regression regularization methods(Lasso, Ridge and ElasticNet) works well in case of high dimensionality and multicollinearity among the variables in the data set. Results: Lasso: mean_squared_error(train data) = 18.91066204969824 r2_score(for train data) = 0.5404051324084427 mean_squared_error(test data) = 18.92519891425323 r2_score(for test data) = 0.5200367203638928 ElasticNet: mean_squared_error(train data) = 19.10944239477544 r2_score(for train data) = 0.5830439917225758 mean_squared_error(test data) = 18.540644854700595 r2_score(for test data) = 0.5805164870884001 Ridge : mean_squared_error(train data) = 18.908273892085823 r2_score(for train data) = 0.618357761314323 mean_squared_error(test data) = 18.930003694792607 r2_score(for test data) = 0.5748015225336987

Regression regularization methods(Lasso, Ridge and ElasticNet) works well in case of high dimensionality and multicollinearity among the variables in the data set.