lasso-ridge-elastic-regression

April 13, 2025

```
[4]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns

     import warnings
     warnings.filterwarnings('ignore')
```

```
[7]: df=sns.load_dataset('mpg')
     df
```

```
[7]:       mpg  cylinders  displacement  horsepower  weight  acceleration  \
     0    18.0          8         307.0       130.0    3504          12.0
     1    15.0          8         350.0       165.0    3693          11.5
     2    18.0          8         318.0       150.0    3436          11.0
     3    16.0          8         304.0       150.0    3433          12.0
     4    17.0          8         302.0       140.0    3449          10.5
     ..    …          …            …          …       …             …
     393  27.0          4         140.0        86.0    2790          15.6
     394  44.0          4          97.0        52.0    2130          24.6
     395  32.0          4         135.0        84.0    2295          11.6
     396  28.0          4         120.0        79.0    2625          18.6
     397  31.0          4         119.0        82.0    2720          19.4

          model_year  origin                       name
     0            70     usa  chevrolet chevelle malibu
     1            70     usa          buick skylark 320
     2            70     usa         plymouth satellite
     3            70     usa              amc rebel sst
     4            70     usa                ford torino
     ..          …       …                        …
     393          82     usa            ford mustang gl
     394          82  europe                  vw pickup
     395          82     usa              dodge rampage
     396          82     usa                ford ranger
     397          82     usa                 chevy s-10
```

```
[398 rows x 9 columns]
```

[ ]:

```
df.drop("name",axis=1,inplace=True)
```

[10]:

```
df
```

[10]:

```
      mpg  cylinders  displacement  horsepower  weight  acceleration  \
0    18.0          8         307.0       130.0    3504          12.0
1    15.0          8         350.0       165.0    3693          11.5
2    18.0          8         318.0       150.0    3436          11.0
3    16.0          8         304.0       150.0    3433          12.0
4    17.0          8         302.0       140.0    3449          10.5
..    …          …           …         …    …           …
393  27.0          4         140.0        86.0    2790          15.6
394  44.0          4          97.0        52.0    2130          24.6
395  32.0          4         135.0        84.0    2295          11.6
396  28.0          4         120.0        79.0    2625          18.6
397  31.0          4         119.0        82.0    2720          19.4

     model_year  origin
0            70     usa
1            70     usa
2            70     usa
3            70     usa
4            70     usa
..          …     …
393          82     usa
394          82  europe
395          82     usa
396          82     usa
397          82     usa

[398 rows x 8 columns]
```

[11]:

```
df.isnull().sum()
```

[11]:

```
mpg             0
cylinders       0
displacement    0
horsepower      6
weight          0
acceleration    0
model_year      0
origin          0
dtype: int64
```

[14]:

```
df["horsepower"]=df["horsepower"].fillna(df["horsepower"].median())
```

[16]:

```
df.dtypes
```

[16]:

```
mpg             float64
cylinders         int64
displacement    float64
horsepower      float64
weight            int64
acceleration    float64
model_year        int64
origin           object
dtype: object
```

[17]:

```
df["origin"].value_counts()
```

[17]:

```
origin
usa       249
japan      79
europe     70
Name: count, dtype: int64
```

[18]:

```
df["origin"]=df["origin"].map({"usa":1,"japan":2,"europe":3})
```

[20]:

```
df["origin"].astype(int)
df.dtypes
```

[20]:

```
mpg             float64
cylinders         int64
displacement    float64
horsepower      float64
weight            int64
acceleration    float64
model_year        int64
origin            int64
dtype: object
```

[21]:

```
# Now Divide x and y
x=df.drop("mpg",axis=1)
y=df['mpg']
```

[23]:

```
# Train Test split
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=1,test_size=0.3)
print(x_train.shape,x_test.shape)
```

```
(278, 7) (120, 7)
```

[27]:

```python
# Simple Linear Regression Model
from sklearn.linear_model import LinearRegression
regression_model=LinearRegression()
regression_model.fit(x_train,y_train)
for i,col_name in enumerate (x_train.columns):
    print(f"The cofficient for {col_name} is : {regression_model.coef_[i]}")
```

```
The cofficient for cylinders is : -0.3176142302799355
The cofficient for displacement is : 0.02623748259907893
The cofficient for horsepower is : -0.018270764913124602
The cofficient for weight is : -0.007487750398361897
The cofficient for acceleration is : 0.0504067346197135
The cofficient for model_year is : 0.8470951427061368
The cofficient for origin is : 1.519095838797505
```

[28]:

```python
# The cofficients are relatively smaller if one independent variable changes slightly not much difference in prediction
# sometime they called as Smoother model

from sklearn.metrics import r2_score
y_pred_linear=regression_model.predict(x_test)
r2_linear=r2_score(y_test,y_pred_linear)
print(f"The R square of Linear Regresion model is: {r2_linear}")
```

```
The R square of Linear Regresion model is: 0.8348001123742286
```

[29]:

```python
from sklearn.linear_model import Ridge
Ridge_model=Ridge(alpha=0.1)
Ridge_model.fit(x_train,y_train)

for i,col_name in enumerate (x_train.columns):
    print(f"The cofficient for {col_name} is : {Ridge_model.coef_[i]}")
```

```
The cofficient for cylinders is : -0.3170032101006609
The cofficient for displacement is : 0.026213249757982955
The cofficient for horsepower is : -0.01826325248144886
The cofficient for weight is : -0.0074873260502131105
The cofficient for acceleration is : 0.05036896947442607
The cofficient for model_year is : 0.8470062938903142
The cofficient for origin is : 1.517452828565376
```

[30]:

```python
y_pred_ridge=Ridge_model.predict(x_test)
r2_ridge=r2_score(y_test,y_pred_ridge)
print(f"The R square of Ridge Regresion model is: {r2_ridge}")
```

```
The R square of Ridge Regresion model is: 0.8348084889168357
```

[31]:

```
from sklearn.linear_model import Lasso
lasso_model=Lasso(alpha=0.5)
lasso_model.fit(x_train,y_train)
for i,col_name in enumerate (x_train.columns):
    print(f"The cofficient for {col_name} is : {lasso_model.coef_[i]}")
```

```
The cofficient for cylinders is : -0.0
The cofficient for displacement is : 0.006208198888300381
The cofficient for horsepower is : -0.011058382987169605
The cofficient for weight is : -0.006982673168 02309
The cofficient for acceleration is : 0.0
The cofficient for model_year is : 0.7446549520038191
The cofficient for origin is : 0.0
```

[32]:

```
y_pred_lasso=lasso_model.predict(x_test)
r2_lasso=r2_score(y_test,y_pred_lasso)
print(f"The R square of Lasso Regresion model is: {r2_lasso}")
```

```
The R square of Lasso Regresion model is: 0.8277934716635554
```

[33]:

```
from sklearn.linear_model import ElasticNet
elastic_model=ElasticNet(alpha=1,l1_ratio=0.5)
elastic_model.fit(x_train,y_train)
for i,col_name in enumerate (x_train.columns):
    print(f"The cofficient for {col_name} is : {elastic_model.coef_[i]}")
```

```
The cofficient for cylinders is : -0.0
The cofficient for displacement is : 0.005888869953667564
The cofficient for horsepower is : -0.012403874933570128
The cofficient for weight is : -0.006934550516257631
The cofficient for acceleration is : 0.0
The cofficient for model_year is : 0.7133150744603873
The cofficient for origin is : 0.0
```

[34]:

```
y_pred_elastic=elastic_model.predict(x_test)
r2_elastic=r2_score(y_test,y_pred_elastic)
print(f"The R square of ElasticNet Regresion model is: {r2_elastic}")
```

```
The R square of ElasticNet Regresion model is: 0.8284840073256803
```

[35]:

```
from sklearn.linear_model import LassoCV
model=LassoCV(cv=5)
model.fit(x_train,y_train)

y_pred=model.predict(x_test)
print(f"The R square of LassoCV is: {r2_score(y_test,y_pred)}")
```

```
The R square of LassoCV is: 0.8082805983844751
```

[37]:

```
from sklearn.linear_model import RidgeCV
RidgeCV_model=RidgeCV(cv=5)
RidgeCV_model.fit(x_train,y_train)

y_pred_ridgecv=RidgeCV_model.predict(x_test)
print(f"The R sqaure of RidgeCV is {r2_score(y_test,y_pred_ridgecv)}")
```

```
The R sqaure of RidgeCV is 0.8354145247502054
```