Chaitanya M. Nawale
BE COB 215

PAGE NO.
DATE :

ML Assignment No. 3

* Title :- Implementation of Linear Regression, Ridge, lasso, and Elastic Net Regression Techniques

* Objective :-
  - To learn Linear Regression technique to solve the real world problems.

* Outcome :-
  Ability to apply Regression techniques to solve real world problems.

* Software Requirements :-
  Python 3, Jupyter Notebook, scikit-learn.

* Problem Statement :-
  The following table shows the results of a recently conducted study on the correlation of the number of hours spent driving with the risk of developing acute backache. Find the equation of the best fit line for this data.

Number of hours spent driving (x)	Risk score 0-100 (y)
10	95
9	80
2	10
15	50
10	45
16	98
11	38

PAGE NO.
DATE:

* Theory :-

• Linear Models :-

Consider a dataset of real-values vectors:

$X = \{\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}\}$ where $\overline{x_i} \in \mathbb{R}^m$

Each input vector is associated with real value y_i

$Y = \{y_1, y_2, \ldots y_n\}$ where $y_n \in \mathbb{R}$

A linear model is based on the assumption that it's possible to approximate the output values through a regression process based on the rule:

$$\tilde{y} = \alpha_0 + \sum_{i=1}^{m} \alpha_i x_i \quad \text{where } A = \{\alpha_0, \alpha_1, \ldots \alpha_m\}$$

As we're working on a plane, the regressor we're looking for is a function of only two parameters:

$$\tilde{y} = \alpha + \beta x$$

In order to fit our model, we must find the best parameters and to do that we choose an ordinary least squares approach. The loss function to minimize is:

$$L = \frac{1}{2} \sum_{i=1}^{n} \| \tilde{y}_i - y_i \|_2^2 \quad \text{which becomes} \quad \frac{1}{2} \sum_{i=1}^{n} (\text{output} -$$

$$L = \frac{1}{2} \sum_{i=1}^{n} (\alpha + \beta x_i - y_i)^2$$

With an analytic approach, in order to find the global minimum, we must impose:

$$\frac{\partial L}{\partial \alpha} = \sum_{i=1}^{n} (\alpha + \beta x_i - y_i) = 0$$

PAGE NO.
DATE :

$$\frac{\partial L}{\partial \beta} = \sum_{i=1}^{n} (\alpha + \beta x_i - y_i) x_i = 0$$

Linear Regression with scikit-learn :-
scikit-learn offers the class LinearRegression which works with n-dimensional spaces. For this purpose, we're going to use the Boston dataset:

```
from sklearn.datasets import load_boston
>>> boston = load_boston()
>>> boston.data.shape
(506L, 13L)
>>> boston.target.shape
(506L,)

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
>>> X_train, X_test, Y_train, Y_test =
    train_test_split(boston.data, boston.target,
    test_size = 0.1)
>>> lr = LinearRegression(normalize= True)
>>> lr.fit(X_train, Y_train)
LinearRegression(copy_X= True, fit_intercept =True,
 n_jobs=1, normalize = True)
>>> lr.score(X_test, Y_test)
    0.773719
from sklearn.model_selection import cross_val_sco
>>> scores = cross_val_score(lr, boston.data,
 boston.target, cv=7, scoring ='neg_mean_squared
 error')
```

PAGE NO.
DATE:

```
array([-11.32601065, -10.96365388, -32.12770594,
  -33.62294354, -10.55 95 71 39, -146.42926647,
  -12.98 53 8412])
>>> scores.mean()
   - 36.85 921
>>>  scores.std()
  45: 704973
```

Ridge, Lasso, and ElasticNet:-

Ridge regression imposes an additional shrikage penalty to the ordinary least Squares losss function to limit its squared L2 norm:

$$L(\bar{\omega}) = \|X\bar{\omega} - \bar{y}\|_2^2 + \alpha\|\bar{\omega}\|_2^2$$

X is a matrix containing all samples as columns and the term ω represents the weight vector. The additional term (through the coefficient alpha - if large it implies a stronger regularization and smaller values) forces the loss function to disallow an infinite growth of ω, which can be caused by multicollinearity or ill-conditioning.

```
from sklearn.datasets import load_diabetes
from sklearn.linear_model import RidgeCV
>>> rg.fit(diabetes.data, diabetes.target)
>>> rg.alpha_
   0.005000
```

Lasso regressor imposes a penalty on the L1 norm of ω to determine a potentially

higher number of null coefficients:

$$L(\bar{w}) = \frac{1}{2n} \|X\bar{w} - \bar{y}\|_2^2 + \alpha \|\bar{w}\|_1$$

The sparsity is a consequence of the penalty term.

```
from sklearn.linear_model import Lasso
>>> ls = Lasso (alpha=0.001, normalize=True)
>>> ls_scores = cross_val_score(ls, diabetes.data,
  diabetes.target , cv=10)
>>>  ls_scores.mean()
  0.462157478
```

ElasticNet which combines both Lasso and Ridge into a single model with two penalty factors: one proportional to L1 norm and the other to L2 norm. In this way, the resulting model will be sparse like a pure Lasso, but with the same regularization ability as provided by Ridge. The resulting loss function is:

```
from sklearn.linear_model import ElasticNet, ElasticNetC
>>>en = ElasticNet(alpha=0.001, l1_ratio=0.8, normalize=True)
>>>en_scores = cross_val_score(en, diabetes.data,
  diabetes.target, cv=10)
>>>  en_scores.mean()
  0.463588588
```

Linear Regression Example:-

PAGE NO.
DATE :

for given problem statement%

$y = slope * x + intercept$

i.e. RiskScore = slope *(Hours) + Intercept -(i)

where,

$slope = \frac{n \Sigma xy - \Sigma x \Sigma y}{n \Sigma x^2 - (\Sigma x)^2}$

$Intercept = \bar{y} - slope * \bar{x}$

we get,

$\bar{x} = 11.125$, $\bar{y} = 63.625$, $\Sigma x = 89$, $\Sigma y = 509$

$\Sigma xy = 6364$, $\Sigma x^2 = 1143$

$\therefore Slope = \frac{8(6364) - (89)(509)}{8(1143) - (89)^2} = 4.58$

intercept = 63625 - (4.58) (11.128)

= 12.59

eqn (1) & RiskScore = (4.58) (Hours) + 12.59

for Hours = 8, RiskScore = (4.58)(8) + 12.59

= 49.23

* Conclusion:- Thus we have success fully implemented different Linear Regression technique for different datasets.