SIDDHANT PATIL 26.02.2024

Maulana Azad National Institute Of Technology, Bhopal

Roll No. 211112238 Scholar No. 211112238

Name: Siddhant Patil CSE-2 ML Assignment-2

Scholar Number: 211112238

Machine Learning (Assignment-2)

(1)

```
# imports
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear-model import LinearRegression, LassoCV,
  RidgeCV, ElasticNetCV
from sklearn.model-selection import train-test-split
from sklearn.metrics import r2-score

df = pd.read-csv("Student-Performance.csv")
df.head()
df['Extracurricular Activities'] = df['Extracurricular Aitivities'].
        apply(lamda x: 0 if x=='No' else 1)

x = df.drop(columns = ['Performance Index']
y = df['Performance Index']
x-train, x-test, y-train, y-test = train-test-split(x, y, test size=
        0.2, random-state = 50)
#Linear Regression
model = LinearRegression()
model.fit(x-train, y-train)
y-pred = model.predict(x-test)
r2 = r2-score(y-test, y-pred)
print(r2)
```

$\boxed{r2 = 0.9889385}$

```
alphas = np.logspace(-6, 2, 100)
l1_ratios = np.linspace(0.01, 1, 100)    # amount of mixing
# Ridge Regularization :-
reg_ridge = RidgeCV(alphas = alphas, cv=5).fit(x_train,
                                               y_train)
reg_ridge.score(reg)        reg_ridge.alpha_
```

| r2 - score = 0.98893~~806~~ 95
| α - value = ~~0.00031992~~ 12.91549

```
# Lasso Regularization :-
reg_lasso = LassoCV(alphas=alphas, cv=5, random_state=50)
             .fit(x_train, y_train)
reg_lasso.score(x_test, y_test)
reg_lasso.alpha_
```

| r2-score = 0.98893884
| α-value = 0.00031,99267

```
# Elastic Net :-
model = ElasticNetCV(alphas=alphas, l1_ratio=l1_ratios,
                     cv=5)
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
r2 = r2_score(y_test, y_pred)
print(r2)
model.score(x_test, y_test)
model.alpha_
```

model. l1-ratio-

$$\boxed{\begin{array}{l} r2\text{-score} = 0.98893946 \\ \alpha\text{-value} = 0.00141747 \\ l1\text{-ratio} = 0.01 \end{array}}$$

Clearly,

In comparision of r2 scores

Ridge Regularalization	>	Elastic Net Regularization	>	Lasso Regularization	>	Simple Linear Regression
$r2 = 0.9889395$ $\alpha = 12.91549$		$r2 = 0.98893946$ $\alpha = 0.00141747$ $l1$ ratio $= 0.01$		$r2 = 0.98893884$ $\alpha = 0.00031992\text{6}7$		$r2 = 0.9889385$

(2) Softmax Regression

Softmax Regression is a form of logistic regression that normalizes an input value into a vector of values that follows a probability distribution whose total sum up to 1.

Softmax Regression is also known as Multinomial logistic regression and Maximum entropy (MaxEnt) classifier.

$$\boxed{\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}}$$

Softmax function

K: No. of class labels

It is used for multiclass classification.

logistic Regression is a special case of softmax regression with classes = 2

Training of model

$$\boxed{\text{Loss function} = \frac{-1}{m}\sum_{i=1}^{m}\sum_{k=1}^{k} y_k^{(i)} \log(\hat{y}_k^{(i)})}$$

m : No. of instances (No. of rows)

k : No. of classes (target variable) (column)

Example of generic dataset,

x_1	x_2	Y	$Y_{k=1}$	$Y_{k=2}$	$Y_{k=3}$	i
x_{11}	x_{12}	1	1	0	0	1
x_{21}	x_{22}	2	0	1	0	2
x_{31}	x_{32}	3	0	0	1	3

$\underbrace{\qquad\qquad}_{\text{One hot encoding}}$ $\quad$ for understanding

$k = \{1, 2, 3\}$

$$L = y_1^{(1)}\log(\hat{y}_1^{(1)}) + y_2^{(1)}\log(\hat{y}_2^{(1)}) + y_3^{(1)}\log(\hat{y}_3^{(1)}) +$$
$$y_1^{(2)}\log(\hat{y}_1^{(2)}) + y_2^{(2)}\log_2(\hat{y}_2^{(2)}) + y_3^{(2)}\log(\hat{y}_3^{(2)}) +$$
$$y_1^{(3)}\log(\hat{y}_1^{(3)}) + y_2^{(3)}\log_2(\hat{y}_2^{(3)}) + y_3^{(3)}\log(\hat{y}_3^{(3)})$$

$$\Rightarrow L = y_1^{(1)}\log(\hat{y}_1^{(1)}) + y_2^{(2)}\log_2(\hat{y}_2^{(2)}) + y_3^{(3)}\log(\hat{y}_3^{(3)}) \quad \text{(I)}$$

For $\hat{y}_1^{(1)}$, according to softmax

Let weights for $y_{k=1}$ are : $w_0^{(1)}, w_1^{(1)}, w_2^{(1)}$

$$\hat{y}_1^{(1)} = \sigma(w_1^{(1)}x_{11} + w_2^{(1)}x_{12} + w_0^{(1)})$$

σ → softmax function

Similarly,

$$\hat{y}_2^{(2)} = \sigma(w_1^{(2)}x_{21} + w_2^{(2)}x_{12} + w_0^{(2)})$$

and

$$\hat{y}_3^{(2)} = \sigma(w_1^{(3)}x_{31} + w_2^{(3)}x_{12} + w_0^{(3)})$$

No. of coefficients
$$\begin{bmatrix} w_1^{(1)} & w_2^{(1)} & w_0^{(1)} \\ w_1^{(2)} & w_2^{(2)} & w_0^{(2)} \\ w_1^{(3)} & w_2^{(3)} & w_0^{(3)} \end{bmatrix}$$

9 variables,

Apply ~~9 d~~ gradient descent.

Find 9 derivatives :—

$$\frac{\partial L}{\partial w_1^{(1)}}, \frac{\partial L}{\partial w_2^{(1)}}, \frac{\partial L}{\partial w_0^{(1)}}, \frac{\partial L}{\partial w_1^{(2)}}, \frac{\partial L}{\partial w_2^{(2)}}, \dots, \frac{\partial L}{\partial w_0^{(3)}}$$

Initialize 9 values (weights)
$$\begin{bmatrix} - & - & - \\ - & - & - \\ - & - & - \end{bmatrix}$$

Loop → 1000 epochs (any number)

//update values of weights of 9 variables

$$w_1^{(1)} = w_1^{(1)} - \eta \frac{\partial L}{\partial w_1^{(1)}}$$
$$w_2^{(1)} = w_2^{(1)} - \eta \frac{\partial L}{\partial w_2^{(1)}}$$
$$\vdots$$
$$w_0^{(3)} = w_0^{(3)} - \eta \frac{\partial L}{\partial w_0^{(3)}}$$

//vectorization can be applied.

Thus we will get weights of all.

Finally we have a trained softmax model.

<u>Prediction from model</u> for (x_a, x_b)

We have 3 classes

$k=1$	$k=2$	$k=3$
$w_0^{(1)}, w_1^{(1)}, w_2^{(1)}$	$w_0^{(2)}, w_1^{(2)}, w_2^{(2)}$	$w_0^{(3)}, w_1^{(3)}, w_2^{(3)}$
$z_1 = x_a w_1^{(1)} + x_b w_2^{(1)} + w_0^{(1)}$	$z_2 = x_a w_1^{(2)} + x_b w_2^{(2)} + w_0^{(2)}$	$z_3 = x_a w_1^{(3)} + x_b w_2^{(3)} + w_0^{(3)}$
$\sigma(k=1) = \frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}$	$\sigma(k=2) = \frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}}$	$\sigma(k=3) = \frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}$

Find the max $\{ e(k=1), e(k=2), e(k=3) \}$
Corresponding would be the class label
for ~~(x_a, y_a)~~ (x_a, x_b)

$$x_a \downarrow x_1, \quad x_b \downarrow x_2$$

This is the full method of Softmax regression.