Jigar Siddhpura
6000421015
C2

(1)

Experiment 2

Aim: To perform linear & find error associated in the model.

Theory:

Linear regression is a supervised ML algorithm. It is used for predictions for continuous (real or numeric variables (like salary, sales, etc.). It shows linear relationship between a dependent variable (y) & 1 or more independent var (x). It shows how dependent variable changes acc. to independent var.

Mathematically, LR is repr as: $y = b_0 + b_1 x + \varepsilon$

y – dependent var x – independent var

b_0 – intercept of line b_1 → LR- co-efficient.

ε – error

Different values of 'x' & 'y' variables, weights or co-efficients of line (b_0, b_1) gives diff. line of regression & cost function is used to estimate best fit line. It is used to find acc. of mapping function, which maps input var. to output var. Here, we used R^2 & MSE where

$$MSE = \frac{1}{N} \sum_{i=1}^{n} [y_i - (b_i x_i - b_0)]^2$$

where, N = total observation

y_i = actual value

$y_0 = b_i x_i + b_0$ = predicted val.

Prod. Procedure :

Part 1.1 (without sklearn)
- import of all libraries
- Here dataset of 2 var (x & y) (array)
- Reshape the lists
- Using formulas for calculating gradients & updating weights

Part 1.2 (with sklearn)
- used 'LinearRegression' library from sklearn.linear_model
- fit the data using fit() function.
- accuracy → used score() function

Part 2.1 (without sklearn, new dataset)
- used the same steps as in 1.1

Part 2.2 (with sklearn, new dataset)
- Here, using new dataset 'Student Performance'
- Perform one hot encoding
- Use library as in 1.2

Part 3.1 (without sklearn, dataset 2)
- used the same steps as in 1.1

Part 3.2 (with library, dataset 2)
- Used the salary data that only has 2 columns
- Here X = YearsExperience & Y = Salary.
- Used the library from here as in 1.2

Observations : for part 1.1, w = [[1.8853]]
b = -0.2708
MSE = 4.1559

for part 1.2, ~~w = [[1.9763]]~~ score = 0.1873
~~b = -0.4873~~
~~MSE = 0.8993~~

for part 2.1 (with formula), w = [[2.7762]]
b = 0.3872
MSE = 3.3126

for part 2.2, ~~w~~ score = 0.1967

for part 3.1, w = [[12315.527]]
b = 6481.917
MSE = 23754857.111

part 3.2, ~~y~~ y_pred = [[63592.049]]

Conclusions : Hence, we learned how to apply LR model on a linear dependent dataset using ~~Statistical~~ method & using library, Also, learned how to calc. cost function i.e. MSE & R^2

1. Describe the procedure that is used to perform Linear regression using Least Square Method carry out the experiment step-by-step for simple linear regression for following dataset with and without using scikit library. Describe every line of code with the proper interpretation of the output.

X	2	3	4	5	6	7	8	9	10
Y	1	3	6	9	11	13	15	17	20

```
"""### 1. without sklearn"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

X = [i for i in range(2,11)]
Y = [1,3,6,9,11,13,15,17,20]

df = pd.DataFrame({'X':X,'Y':Y})

X = df['X'].values.reshape(-1,1)
Y = df['Y'].values.reshape(-1,1)

learning_rate=0.01
no_of_itr=100
y_pred_arr = []

m,n = X.shape
w = np.zeros((n,1))
b = 0

def predict(X):
    return X.dot(w) + b

def update_wt():
    global w
    global b
    y_pred = predict(X)
    y_pred_arr.append(y_pred)
    # calculating gradients
    dw = -(X.T).dot(Y - y_pred)/m

    db = -np.sum(Y - y_pred)/m

    # updating weights
    w = w - learning_rate * dw
    b = b - learning_rate * db
for i in range(no_of_itr):
  update_wt()

print(w)
print(b)
MSE = np.square(np.subtract(Y,y_pred_arr)).mean()
print(MSE)
```

OUTPUT:

```
[[1.88536856]]
-0.27035591489738453
4.155960303200094
```

WITH SKLEARN:

```
"""### 1.2 with *sklearn*"""

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

X = df['X'].values.reshape(-1,1)
Y = df['Y'].values.reshape(-1,1)

model = LinearRegression()
model.fit(X,Y)
print(model.score)
```

OUTPUT:

```
0.196734563733
```

2. DATASET : STUDENT PERFORMANCE

WITH SKLEARN:

```
"""### 2.1 with dataset with lib"""

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('/content/gdrive/MyDrive/DMW/datasets/StudentsPerformance.csv')
df.head()
df['final_score'] = df.apply(lambda x: (x['math score'] + x['reading score'] +
x['writing score'])/3, axis=1)

df2 = df
df2 = df2.drop(['math score','reading score','writing score'],axis=1)

df2.head()
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

```
df2 = pd.get_dummies(df2, columns=['gender','lunch','parental level of
education','race/ethnicity','test preparation course'])
df2.head()
```

	final_score	gender_female	gender_male	lunch_free/reduced	lunch_standard	parental level of education_associate's degree	parental level of education_bachelor's degree	parental level of education_high school	parental level of education_master's degree	parental level of education_some college	paren...
0	72.666667	1	0	0	1	0	1	0	0	0	
1	82.333333	1	0	0	1	0	0	0	0	1	
2	92.666667	1	0	0	1	0	0	0	1	0	
3	49.333333	0	1	1	0	1	0	0	0	0	
4	76.333333	0	1	0	1	0	0	0	0	1	

```
# multi-variate
y = df2['final_score']
x = df2.drop(['final_score'],axis=1)

xtrain, xtest, ytrain, ytest = train_test_split(x,y,test_size=0.25,random_state=10)

model = LinearRegression()
model.fit(xtrain,ytrain)
score = model.score(xtest,ytest)
print(score)
ypred = model.predict(xtest)
```

OUTPUT:

```
0.19674412629893356
```

WITHOUT SKLEARN:

```
"""### 2.2 with formula"""

X = df['X'].values.reshape(-1,1)
Y = df['Y'].values.reshape(-1,1)

learning_rate=0.01
no_of_itr=100
y_pred_arr = []

m,n = X.shape
w = np.zeros((n,1))
b = 0

def predict(X):
    return X.dot(w) + b

def update_wt():
    global w
    global b
    y_pred = predict(X)
    y_pred_arr.append(y_pred)

    # calculating gradients
    dw = -(X.T).dot(Y - y_pred)/m
    db = -np.sum(Y - y_pred)/m
    # updating weights
    w = w - learning_rate * dw
    b = b - learning_rate * db

for i in range(no_of_itr):
  update_wt()
```

```
print(w)
print(b)
MSE = np.square(np.subtract(Y,y_pred_arr)).mean()
print(MSE)
```

OUTPUT:

```
[[2.776264564763]]
0.3872355565633
3.312643565737
```

3. DATASET : SALARY

WITHOUT SKLEARN:

```
"""### 3.1 without lib"""

df = pd.read_csv('/content/gdrive/MyDrive/BDI/salary_data.csv')
df.head()
```

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

```
X = df['YearsExperience'].values.reshape(-1,1)
Y = df['Salary'].values.reshape(-1,1)
learning_rate=0.01
no_of_itr=100
y_pred_arr = []

m,n = X.shape
w = np.zeros((n,1))
b = 0
def predict(X):
    return X.dot(w) + b

def update_wt():
    global w
    global b
    y_pred = predict(X)
    y_pred_arr.append(y_pred)

    # calculating gradients
```

```
    dw = -(X.T).dot(Y - y_pred)/m

    db = -np.sum(Y - y_pred)/m

    # updating weights
    w = w - learning_rate * dw
    b = b - learning_rate * db

for i in range(no_of_itr):
  update_wt()

print(w)
print(b)
MSE = np.square(np.subtract(Y,y_pred_arr)).mean()
print(MSE)
```

OUTPUT:

```
[[12315.52743012]]
6481.917498721374
237548657.81185108
```

WITH SKLEARN:

```
"""# 3.2"""

model = LinearRegression()
model.fit(X,Y)
ypred = model.predict([[4]])
print(ypred)
```

OUTPUT:

```
[[63592.04948449]]
```