UNIVERSITY OF ASIA PACIFIC

Department of Computer Science of Engineering

Project-2:

Lab Report

Course code: CSE(404)

Course Title: Artificial Intelligence and Expert System Lab

Submitted By:

Anik Chandra

ID : 18101070

Sec: B

Submitted To:

Dr. Nasima Begum

Assistant Professor

Index

Objective: In this project the main idea is to implement the multivariable linear regression without SK-Learn and with SK-learn.In this project try to find out accurate predict result for using selected dataset. And also find out Cost,Hypothesis Function value and Updated parameters.

About(Multivariable Linear Regression): Multivariable Linear Regression also known simply as multiple regression is a statistical technique that uses several explanatory variables to predict the outcome of a response variable.The goal of Multivariable Linear Regression is to model the linear relationship between the explanatory variables and response variable.

Formula and calculation of Multivariable Linear Regression

$$y = t0 + t1x1 + t2x2 + \ldots + tpxp$$

Where,

y= dependent variable

xp= explanatory variable

t0= y-intercept

tp= slope coefficients for each explanatory variable

Multivariate Linear Regression

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Gradient descent:

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

}

(simultaneously update for every $j = 0, \ldots, n$)

About(Scikit-Learn): Scikit-Learn is a Python package that simplifies the implementation of a wide range of Machine Learning(ML) methods for predictive data analysis,including linear regression.

The following are some key concepts we will come across when we work with scikit-learn linear regression methods here some method we used:

-> **Coefficient** - also known as a parameter, is the factor a variable is multiplied by. In linear regression, a coefficient represents changes in a Response Variable**.**

-> **Intercept** - the location where the Slope intercepts the Y-axis denoted b in the slope equation y=ax+b.

-> **Mean** - an average of a set of numbers, but in linear regression, Mean is modeled by a linear function.

-> **Regression Model** - the ideal formula for approximating a regression.

Linear Regression Class Definition: A scikit-learn linear regression script begins by importing the Linear Regression class

```
from sklearn.linear_model import LinearRegression
sklearn.linear_model.LinearRegression().
```

Dataset: In this project i used a data form kaggle the dataset name are, Name: 'Petrol Consumption' ,In this dataset 5 column these are,

‘Petrol_tax’, ’Average_income’, ’Paved_Highways’, ’Population_Driver_Licence(%)’,’Petrol_Consumption’

Link->https://www.kaggle.com/anikchandra70/petrol-consumption-multivariable-linear-regression

Implementation:

For implementation I used Python programming language,Excel dataset,Scikit-learn and Colab notebooks.

In this part I discuss some points of my implementation and code.

First I discuss how to read an excel dataset in Colab. In this case i use python pandas libraries

Implement Multivariable Linear Regression without Scikit-Learn->

First import pandas libraries and read the file **“pd.read_csv()”**.and put the excel content path in this function.

```
# import data and view data
df=pd.read_csv('/content/petrol_consumption.csv') # A simple way to store big
df.head() # Return the first 5 rows of the DataFrame.
```

	Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence(%)	Petrol_Consumption
0	9.0	3571	1976	0.525	541
1	9.0	4092	1250	0.572	524
2	9.0	3865	1586	0.580	561
3	7.5	4870	2351	0.529	414
4	8.0	4399	431	0.544	410

Then I rename() the data frame [‘Petrol_tax’ : ‘x1’,....] for further calculation.

```
[ ]  1  # rename() Rename the row,column indexes of the DataFrame
     2  df = df.rename(columns={'Petrol_tax': 'x1', 'Average_income': 'x2', 'Paved_Highways'
     3  df.head() # Return the first 5 rows of the DataFrame.
```

	x1	x2	x3	x4	y
0	9.0	3571	1976	0.525	541
1	9.0	4092	1250	0.572	524
2	9.0	3865	1586	0.580	561
3	7.5	4870	2351	0.529	414
4	8.0	4399	431	0.544	410

Then I define a **predict()** function for prediction.According to Multivariable Linear Regression formula

```
[ ]  1  # Define Predict function
     2  # initialize t0-t4 for value of theta or Coefficient
     3  # equation  y =  t0 + t1 * x1 + t2 * x2 + t3 * x3 + t4 * x4
     4  def predict(t0,t1,t2,t3,t4,x1,x2,x3,x4):
     5      return t0 + t1 * x1 + t2 * x2 + t3 * x3 + t4 * x4
```

Then I define a **computeCost()** function for cost,hypothesis,parameter according to formula.

```
 1  # Define computeCost function
 2  def computeCost(t0,t1,t2,t3,t4,x1,x2,x3,x4,y):
 3      # Getting number of data
 4      m = float(len(y))
 5      loss = [] # initialize list for stroe
 6      # Iterating over all of the data
 7      for i in range(len(y)):
 8          # Getting prediction using the parameter [t0, t1, t2 ,t3,4]
 9          h = predict(t0,t1,t2,t3,t4,x1[i],x2[i],x3[i],x4[i])
10          # Adding the losses to the list
11          loss.append((h - y[i])**2)
12      # Printing Hypothesis Function Value after each epoch
13      print("Hypothesis Function Value is: ",loss)
14      # Printing Updated parameters after each epoch
15      print("Updated parameters are: ",t0,t1,t2,t3,t4)
16      return (sum(loss) / (2 * m))
```

Then i apply **epoch = 30** for cost,hypothesis and parameter value using batch gradient descent formula.and finally predict result

```
# prediction result
predict(425.599332,-40.016660,-0.065413,-0.004741,1341.862121,9.0,3571,1976,0.525)
```

```
526.9689665249999
```

Implement Multivariable Linear Regression with Scikit-Learn->

First import pandas libraries and read the file “pd.read_csv()”.and put the excel content path in this function.

```
# import data and view data
df=pd.read_csv('/content/petrol_consumption.csv') # A simple way to store big
df.head() # Return the first 5 rows of the DataFrame.
```

	Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence(%)	Petrol_Consumption
0	9.0	3571	1976	0.525	541
1	9.0	4092	1250	0.572	524
2	9.0	3865	1586	0.580	561
3	7.5	4870	2351	0.529	414
4	8.0	4399	431	0.544	410

Then show the basic statistical details like count,mean,std etc using panda **describe()** method

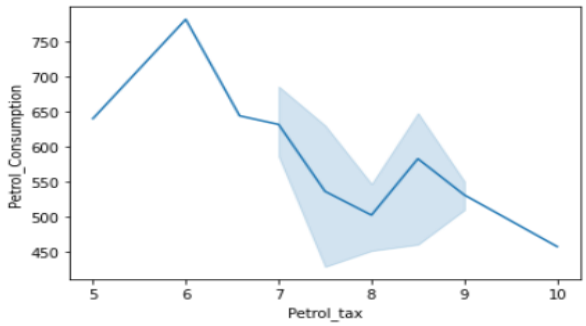
```
# Pandas describe() is used to view some basic statistical details like percentile,
df.describe()
```

	Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence(%)	Petrol_Consumption
count	48.000000	48.000000	48.000000	48.000000	48.000000
mean	7.668333	4241.833333	5565.416667	0.570333	576.770833
std	0.950770	573.623768	3491.507166	0.055470	111.885816
min	5.000000	3063.000000	431.000000	0.451000	344.000000
25%	7.000000	3739.000000	3110.250000	0.529750	509.500000
50%	7.500000	4298.000000	4735.500000	0.564500	568.500000
75%	8.125000	4578.750000	7156.000000	0.595250	632.750000
max	10.000000	5342.000000	17782.000000	0.724000	968.000000

Then draw a line plot **Petrol_tax vs Pertrol_Consumption** for data visualization in this case use **Seaborn packages** for 3d visualization.

```
# Draw a line plot with possibility of several semantic
sns.lineplot(x=df['Petrol_tax'],y=df['Petrol_Consumption'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fadd3d7f390>
```



For model train purpose we creating an attribute set and label

```
# We are going to use column names for creating an attribute set and label.
X=df[['Petrol_tax','Average_income','Paved_Highways','Population_Driver_licence(%)']]
y=df['Petrol_Consumption']
```

Then dividing the data into training and test purpose

```
[ ]  1  # Dividing our data into training and test sets
     2  from sklearn.model_selection import train_test_split
     3  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

Then training the model use scikit-learn **LinearRegression()** method and fit the model

```
[ ]  1  # Training the Algorithm
     2  from sklearn.linear_model import LinearRegression
     3  reg=LinearRegression()
     4  reg.fit(X_train,y_train)
```

Using scikit-learn and find out **Coefficient and Intercept**

```
[ ]  1  # In case of multivariable linear regression, the regression model has to find the
     2  coeff_df = pd.DataFrame(reg.coef_, X.columns, columns=['Coefficient'])
     3  coeff_df
```

	Coefficient
Petrol_tax	-40.016660
Average_income	-0.065413
Paved_Highways	-0.004741
Population_Driver_licence(%)	1341.862121

```
[ ]  1  # The regression model has to find the most optimal intercept for all the attributes.
     2  int_df = pd.DataFrame(reg.intercept_, X.columns, columns=['intercept'])
     3  int_df
```

	intercept
Petrol_tax	425.599332
Average_income	425.599332
Paved_Highways	425.599332
Population_Driver_licence(%)	425.599332

Then making prediction and predict the result

```
[ ]  1  # Making Predictions
     2  y_pred=reg.predict(X_test)
```

```
[ ]  1  # Predictions
     2  y_pred=reg.predict([[9.0,3571,1976,0.525]])
     3  y_pred
```

```
array([526.97067878])
```

Some visualization on the report part we discuss all visualization in code .

Result Analysis:

Formula and calculation of Multivariable Linear Regression

$$y = t0 + t1x1 + t2x2 + \ldots + tpxp$$

So in this case we found value from code result

Petrol_Consumption-> y = ?

Intercept-> t0= 425.599332

Petrol_tax -> t1= -40.016660

Average_income-> t2= -0.065413

Paved_Highways-> t3= -0.004741

Population_Driver_Licence(%)-> t4= 1341.862121

We assume the x1,x2,x3 and x4 form dataset these are

x1= 9.0

x2= 3571

x3= 1976

x4= 0.525

y= t0 + t1x1 + t2x2 + t3x3 + t4x4
y= 425.599332 + 9.0*(-40.016660) + 3571*(-0.065413) + 1976*(-0.004741) + 0.525*1341.862121
y= 526.9689665249999

Both with Scikit-learn and without Scikit-learn prediction result and manually calculation result are the same.

Conclusion:

In this project I implement Multivariable Linear Regression with Scikit-learn and without Scikit-learn.And also visualize the necessary parameters and data using **'Seaborn packages'** , and Predict **Petrol_Consumption** use some data me already discuss and calculation the result.Both with Scikit-learn and without Scikit-learn prediction result and manually calculation result are the same.In this project my implementation result and calculation result are the same so I hopefully say that this the **Petrol_Consumption** prediction model is right.