Green University Of Bangladesh

Department Of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Fall, Year: 2023), B.Sc. in CSE (DAY)

LAB REPORT NO - 05

Course Title: Data Mining Lab

Course Code: CSE-436 **Section:** D2

Lab Experiment Name: Linear & Logistic Regression

Student Details

	Name	ID
1	Sk. Nahid	201902073

Lab Date : 25/11/2023
Submission Date : 01/12/2023
Course Teacher Name : **Rezwanul Haque**

Lab Report Status	
Mark:.........................	**Signature:...............**
Comments:.................	**Date:.....................**

1 INTRODUCTION

Regression is a method for understanding the relationship between independent variables or features and a dependent variable or outcome. Linear regression analysis is used to predict the value of a variable based on the value of another variable. Logistic regression estimates the probability of an event occurring.

2 OBJECTIVE

This lab report aims to determine logistic & linear regression and their implementation in python.

3 IMPLEMENTATION

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,mean_squared_error
from sklearn.linear_model import LogisticRegression
```

Listing 1: Import Library & Dataset

```
df = pd.read_csv('/kaggle/input/average-temperature-from-1900-to-2023/
    Average Temperature 1900-2023.csv')
df.head()
```

Listing 2: Read the dataset

```
X = np.array(df['Year']).reshape((-1, 1))
y = np.array(df['Average_Fahrenheit_Temperature']).reshape((-1, 1))
df.dropna(inplace = True)
```

Listing 3: Creating X Y

```
Linear regression with 20% split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

regr = LinearRegression()

regr.fit(X_train, y_train)
print("Linear Regression: ",regr.score(X_test, y_test))

y_pred = regr.predict(X_test)
```

```
plt.scatter(X_test, y_test, color ='b')
plt.plot(X_test, y_pred, color ='k')
plt.show()
mae = mean_absolute_error(y_true=y_test,y_pred=y_pred)
mse = mean_squared_error(y_true=y_test,y_pred=y_pred) #default=True
rmse = mean_squared_error(y_true=y_test,y_pred=y_pred,squared=False)
print("MAE:",mae)
print("MSE:",mse)
print("RMSE:",rmse)
```

Listing 4: Split the dataset in 8:2 and implementing Linear Regression

```
Linear regression with 30% split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)

regr = LinearRegression()

regr.fit(X_train, y_train)
print("Linear Regression: ",regr.score(X_test, y_test))

y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color ='b')
plt.plot(X_test, y_pred, color ='k')
plt.show()
#Evaluation Metrics For Regression
mae = mean_absolute_error(y_true=y_test,y_pred=y_pred)
mse = mean_squared_error(y_true=y_test,y_pred=y_pred) #default=True
rmse = mean_squared_error(y_true=y_test,y_pred=y_pred,squared=False)
print("MAE:",mae)
print("MSE:",mse)
print("RMSE:",rmse)
```

Listing 5: Split the dataset in 7:3 and implementing Linear Regression

```
    col_names = ['AGE', 'SMOKING']
df = pd.read_csv("/kaggle/input/lung-cancer/survey lung cancer.csv")
X = np.array(df['AGE']).reshape((-1, 1))
y = np.array(df['SMOKING']).reshape((-1, 1))

df.dropna(inplace = True)
```

Listing 6: Preparing dataset for logistic regression

```
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
   0.2, random_state=23)

logreg = LogisticRegression()

logreg.fit(X_train,y_train)
y_pred=logreg.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

```python

class_names=[0,1] # name of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

Listing 7: Logistic Regression on 20% testing

```python
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=23)

logreg = LogisticRegression()

logreg.fit(X_train,y_train)
y_pred=logreg.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix

class_names=[0,1] # name of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

Listing 8: Logistic Regression on 30% testing

4 OUTPUT

	Year	Average_Fahrenheit_Temperature
0	1900	53.9
1	1901	53.5
2	1902	52.1
3	1903	50.6
4	1904	51.8

Figure 1: Dataset details

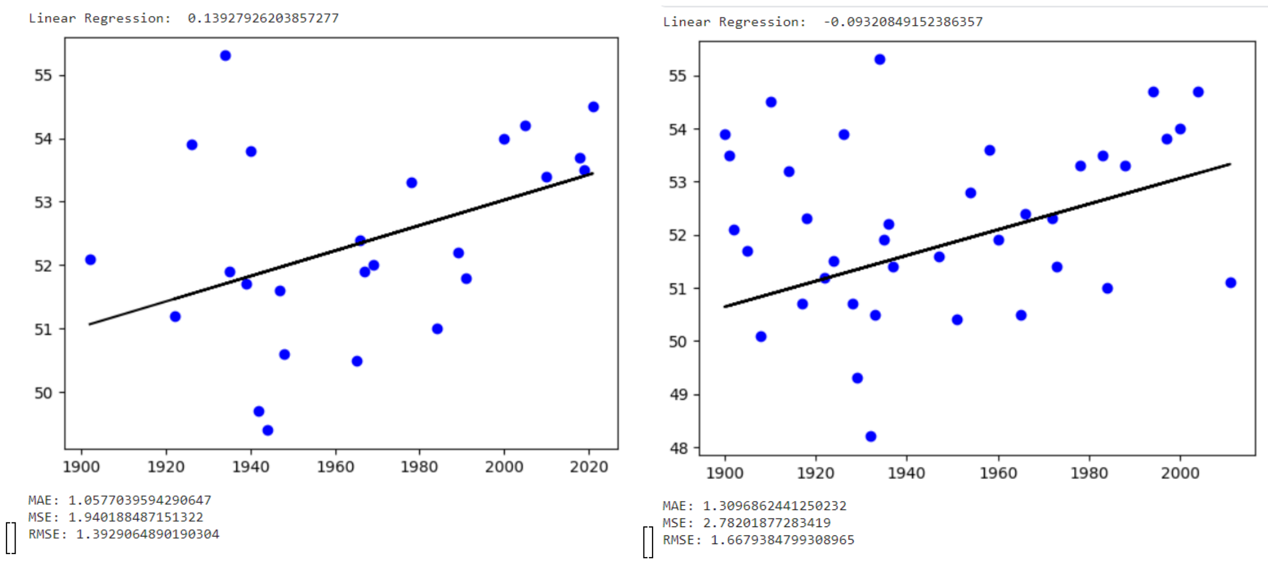

Figure 2: Linear Regression (a) 20% Dataset (b) 30% Dataset

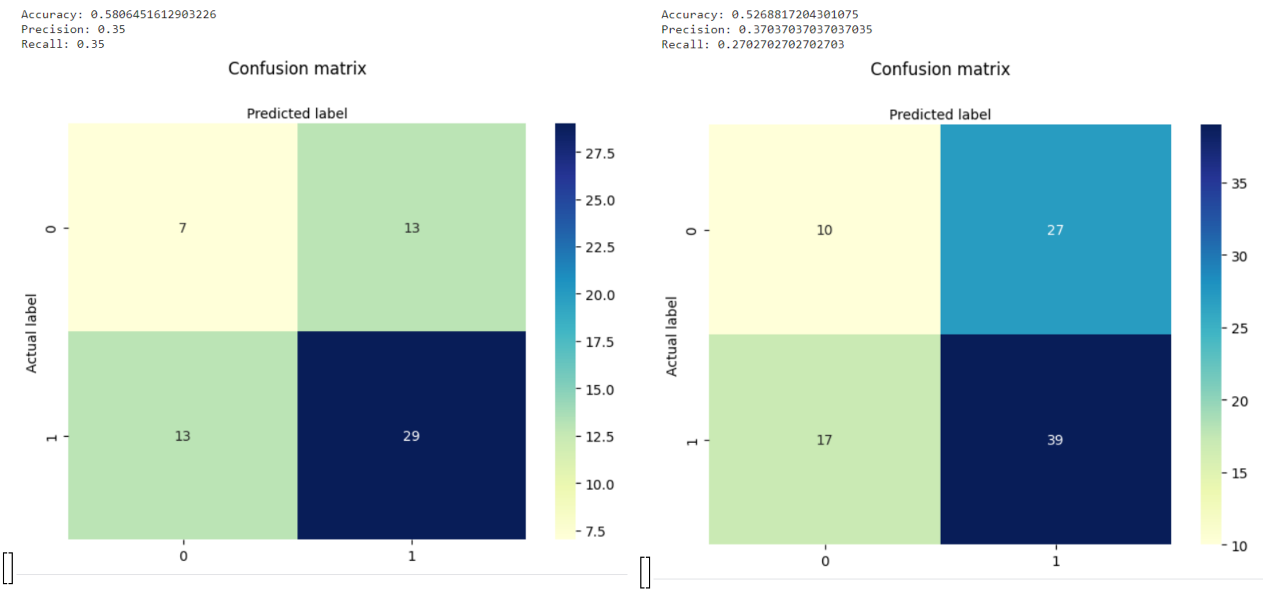

Figure 3: Logistic Regression (a) 20% Dataset (b) 30% Dataset

5 DISCUSSION & ANALYSIS

In this study, logistic regression was applied to a dataset with categorical outcomes, while linear regression was used for a dataset with continuous numerical outcomes. This approach allowed us to address distinct aspects of our research question, leveraging the strengths of each regression technique. The use of different datasets was intentional and aligned with the nature of the variables and the specific objectives of each analysis.