7_March_Reg

March 7, 2025

1 Name : Tufan Kundu

2 Reg no. : 24MDT0184

3 Experiment 6

4 7 March, 2025

4.1 Study and Implementation of Logistic Regression in Python programming language

4.2 Objective:

Consider data published on n = 27 leukemia patients. The data logistic_regression_data.csv has a response variable of whether leukemia remission occurred (REMISS), which is given by a 1. The predictor variables are cellularity of the marrow clot section (CELL), smear differential percentage of blasts (SMEAR), percentage of absolute marrow leukemia cell infiltrate (INFIL), percentage labeling index of the bone marrow leukemia cells (LI), absolute number of blasts in the peripheral blood (BLAST), and the highest temperature prior to start of treatment (TEMP)

4.2.1 Importing the necessary libraries

[34]:
```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,␣
↪confusion_matrix,classification_report
import matplotlib.pyplot as plt
import seaborn as sns
```

4.2.2 Loading the dataset

[35]:
```
df = pd.read_excel(r"C:\Users\Batch1\Downloads\TK\Regression\data.xlsx")
df
```

[35]:
```
    REMISS  CELL  SMEAR  INFIL   LI  BLAST  TEMP
0        1  0.80   0.83   0.66  1.9   1.10  1.00
1        1  0.90   0.36   0.32  1.4   0.74  0.99
2        0  0.80   0.88   0.70  0.8   0.18  0.98
3        0  1.00   0.87   0.87  0.7   1.05  0.99
4        1  0.90   0.75   0.68  1.3   0.52  0.98
5        0  1.00   0.65   0.65  0.6   0.52  0.98
6        1  0.95   0.97   0.92  1.0   1.23  0.99
7        0  0.95   0.87   0.83  1.9   1.35  1.02
8        0  1.00   0.45   0.45  0.8   0.32  1.00
9        0  0.95   0.36   0.34  0.5   0.00  1.04
10       0  0.85   0.39   0.33  0.7   0.28  0.99
11       0  0.70   0.76   0.53  1.2   0.15  0.98
12       0  0.80   0.46   0.37  0.4   0.38  1.01
13       0  0.20   0.39   0.08  0.8   0.11  0.99
14       0  1.00   0.90   0.90  1.1   1.04  0.99
15       1  1.00   0.84   0.84  1.9   2.06  1.02
16       0  0.65   0.42   0.27  0.5   0.11  1.01
17       0  1.00   0.75   0.75  1.0   1.32  1.00
18       0  0.50   0.44   0.22  0.6   0.11  0.99
19       1  1.00   0.63   0.63  1.1   1.07  0.99
20       0  1.00   0.33   0.33  0.4   0.18  1.01
21       0  0.90   0.93   0.84  0.6   1.59  1.02
22       1  1.00   0.58   0.58  1.0   0.53  1.00
23       0  0.95   0.32   0.30  1.6   0.89  0.99
24       1  1.00   0.60   0.60  1.7   0.96  0.99
25       1  1.00   0.69   0.69  0.9   0.40  0.99
26       0  1.00   0.73   0.73  0.7   0.40  0.99
```

4.3 Defining x(independent), y (dependent) variable

[36]:
```python
x = df.drop('REMISS',axis = 1)
y = df['REMISS']
```

4.4 Multiple linear Regression

[37]:
```python
X_const = sm.add_constant(x)
model_linear = sm.OLS(y,X_const).fit()
print("\nMultiple Linear Regression results:\n", model_linear.summary())
```

```
Multiple Linear Regression results:
                             OLS Regression Results
==============================================================================
Dep. Variable:                 REMISS   R-squared:                       0.349
Model:                            OLS   Adj. R-squared:                  0.153
Method:                 Least Squares   F-statistic:                     1.785
Date:                Fri, 07 Mar 2025   Prob (F-statistic):              0.153
```

```
Time:                        12:55:02   Log-Likelihood:                -12.216
No. Observations:                  27   AIC:                             38.43
Df Residuals:                      20   BIC:                             47.50
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.0018      6.717      0.745      0.465      -9.010      19.014
CELL          -0.2222      1.780     -0.125      0.902      -3.935       3.491
SMEAR         -1.5288      3.388     -0.451      0.657      -8.595       5.537
INFIL          1.5842      3.851      0.411      0.685      -6.449       9.617
LI             0.5350      0.267      2.006      0.059      -0.021       1.091
BLAST         -0.0092      0.335     -0.027      0.978      -0.709       0.690
TEMP          -4.9492      6.693     -0.739      0.468     -18.910       9.012
==============================================================================
Omnibus:                        0.828   Durbin-Watson:                   2.612
Prob(Omnibus):                  0.661   Jarque-Bera (JB):                0.742
Skew:                          -0.068   Prob(JB):                        0.690
Kurtosis:                       2.199   Cond. No.                         252.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

4.5 Logistic Regression

4.5.1 Train test split

[38]:
```python
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,␣
 ↪random_state=42)

logisticR = LogisticRegression()
logisticR.fit(x_train,y_train)

# prediction
y_pred = logisticR.predict(x_test)

# model evaluation
accuracy = accuracy_score(y_test,y_pred)
conf_matrix = confusion_matrix(y_test,y_pred)
report = classification_report(y_test,y_pred)

print("Logistic Regression accuracy:", accuracy*100,"%")
print("\nConfusion Matrix:\n", conf_matrix)
print("\nClassification Report:\n", report)
```

```
Logistic Regression accuracy: 100.0 %

Confusion Matrix:
 [[5 0]
 [0 1]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         5
           1       1.00      1.00      1.00         1

    accuracy                           1.00         6
   macro avg       1.00      1.00      1.00         6
weighted avg       1.00      1.00      1.00         6
```

4.6 Logistic Regression on the Diabetes dataset

4.7 Loading the dataset

[39]:
```python
df = pd.read_excel(r"C:\Users\Batch1\Downloads\TK\Regression\diabetes.xlsx")
df
```

[39]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           …        …              …              …      …   …
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        …  …      …
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
```

```
767                     0.315   23        0

[768 rows x 9 columns]
```

[40]:
```
df['Outcome'].value_counts()
```

[40]:
```
Outcome
0    500
1    268
Name: count, dtype: int64
```

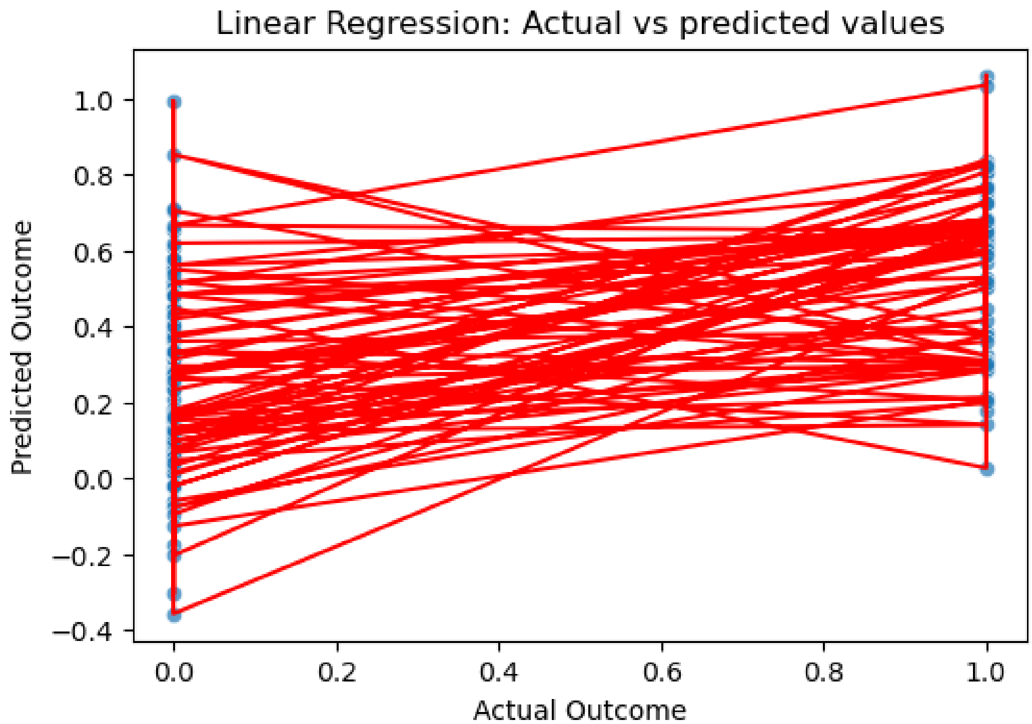
4.8 The data is imbalanced, having records of more people without diabetes

4.8.1 Defining the features and the target variable

[41]:
```
x = df.drop('Outcome', axis = 1)
y = df['Outcome']

## Train test split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,␣
 ↪random_state=42)
```

[46]:
```
##==================Linear Regression=================================
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

lin_reg = LinearRegression()
lin_reg.fit(x_train,y_train)
y_pred_lin = lin_reg.predict(x_test)


mse = mean_squared_error(y_test,y_pred_lin)
r2  = r2_score(y_test,y_pred_lin)

print("Mean Squared error of the linear Regression:", mse)
print("R2 score of the linear Regression:", r2)

## Scatter plot of actual vs predicted values for Linear Regression

plt.figure(figsize=(6,4))
sns.scatterplot(x = y_test, y = y_pred_lin, alpha = 0.7)
plt.plot(y_test,y_pred_lin,color = 'red')
plt.title("Linear Regression: Actual vs predicted values")
plt.xlabel("Actual Outcome")
plt.ylabel("Predicted Outcome")
plt.show()
```

```
Mean Squared error of the linear Regression: 0.17104527280850096
R2 score of the linear Regression: 0.2550028117674178
```



[27]:
```
#=========================Logistic Regression==============================

log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(x_train,y_train)
y_pred_log = log_reg.predict(x_test)

## Evaluating Logistic regression
accuracy_log = accuracy_score(y_test,y_pred_log)
conf_mat_log = confusion_matrix(y_test,y_pred_log)
print("Logistic Regression accuracy:", accuracy_log*100,"%")
print("\nConfusion Matrix:\n", conf_mat_log)
print("\nClassification Report:\n", classification_report(y_test,y_pred_log))


## Heatmap for confusion matrix

plt.figure(figsize = (6,4))
sns.heatmap(conf_mat_log,annot = True, fmt='d', cmap='coolwarm', xticklabels =␣
 ↪['Non-Diabetic','Diabetic'], yticklabels=['Non-Diabetic','Diabetic'])
plt.xlabel("Predicted")
```

```python
plt.ylabel("Actual")
plt.title("Logistic Regression: Confusion matrix")
plt.show()
```

```
Logistic Regression accuracy: 74.67532467532467 %

Confusion Matrix:
 [[78 21]
 [18 37]]

Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.79      0.80        99
           1       0.64      0.67      0.65        55

    accuracy                           0.75       154
   macro avg       0.73      0.73      0.73       154
weighted avg       0.75      0.75      0.75       154
```