LogisticRegression

March 7, 2025

1 Regression Analysis Lab: $7^{th} March, 2025$

1.1 Name: Soumyadeep Ganguly

1.2 Reg No: $24MDT0082$

[65]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
import seaborn as sns
import matplotlib.pyplot as plt
```

2 Problem 1

[30]:
```
df = pd.read_csv("data.csv")
df.head()
```

[30]:

	REMISS	CELL	SMEAR	INFIL	LI	BLAST	TEMP
0	1	0.8	0.83	0.66	1.9	1.10	1.00
1	1	0.9	0.36	0.32	1.4	0.74	0.99
2	0	0.8	0.88	0.70	0.8	0.18	0.98
3	0	1.0	0.87	0.87	0.7	1.05	0.99
4	1	0.9	0.75	0.68	1.3	0.52	0.98

[31]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

data_scaled = scaler.fit_transform(df)
df_scaled = pd.DataFrame(data_scaled, columns=df.columns)
df_scaled.head()
```

[31]:

	REMISS	CELL	SMEAR	INFIL	LI	BLAST	TEMP
0	1.0	0.750	0.784615	0.690476	1.000000	0.533981	0.333333

1	1.0	0.875	0.061538	0.285714	0.666667	0.359223	0.166667
2	0.0	0.750	0.861538	0.738095	0.266667	0.087379	0.000000
3	0.0	1.000	0.846154	0.940476	0.200000	0.509709	0.166667
4	1.0	0.875	0.661538	0.714286	0.600000	0.252427	0.000000

2.1 Multiple Linear Regression

[37]:
```
X = df_scaled.drop(["REMISS"], axis = 1)
y = df_scaled["REMISS"]

X_const = sm.add_constant(X)
model_linear = sm.OLS(y, X_const).fit()
print("\n MLR Result", model_linear.summary())
```

```
 MLR Result                            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:                        12:26:55   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         -0.0414      0.515     -0.080      0.937      -1.116       1.034
CELL          -0.1777      1.424     -0.125      0.902      -3.148       2.793
SMEAR         -0.9938      2.202     -0.451      0.657      -5.587       3.599
INFIL          1.3308      3.235      0.411      0.685      -5.417       8.078
LI             0.8025      0.400      2.006      0.059      -0.032       1.637
BLAST         -0.0189      0.691     -0.027      0.978      -1.460       1.422
TEMP          -0.2970      0.402     -0.739      0.468      -1.135       0.541
==============================================================================
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.                         81.8
==============================================================================

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

[ ]:

2.2 Logistic Regression

[32]:
```
X = df_scaled.drop(["REMISS"], axis = 1)
y = df_scaled["REMISS"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state=42)

LR = LogisticRegression()
LR.fit(X_train, y_train)
```

[32]:
```
LogisticRegression()
```

[33]:
```
y_pred = LR.predict(X_test)
```

[47]:
```
print(f"Accuracy of the Logistic Regression Model: {accuracy_score(y_test,␣
 ↪y_pred)}")
```

```
Accuracy of the Logistic Regression Model: 1.0
```

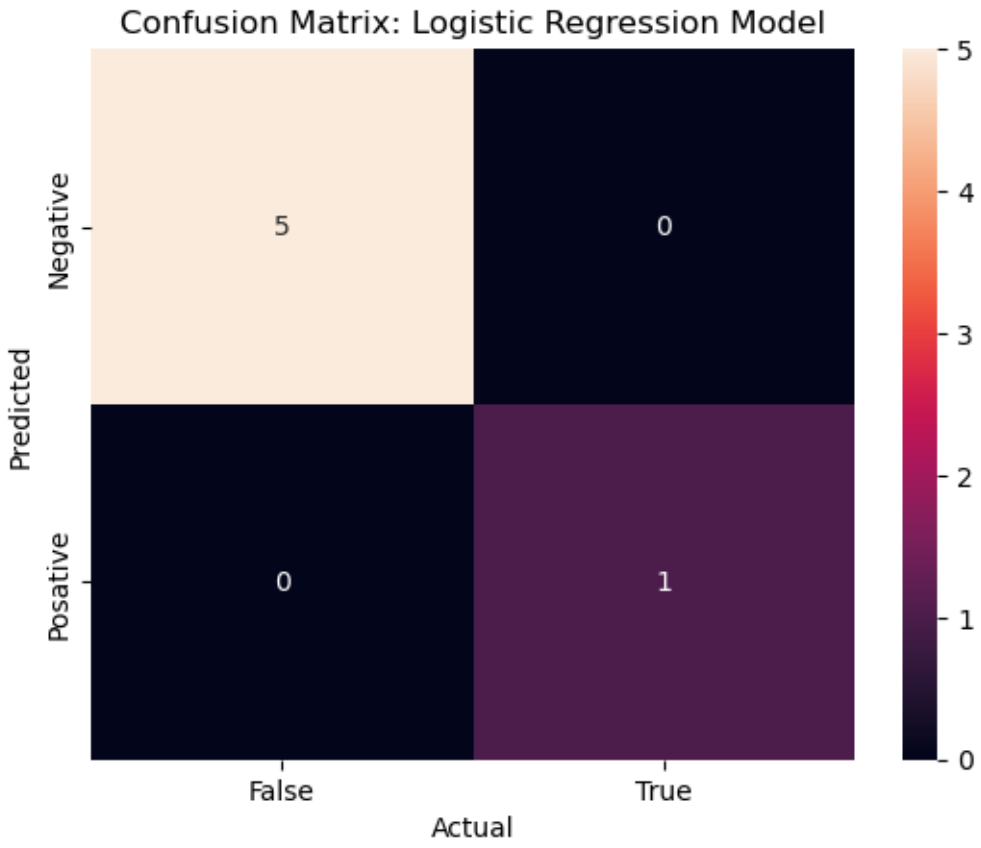
[49]:
```
print(f"Classification Report of the Logistic Regression Model:\n␣
 ↪{classification_report(y_test, y_pred)}")
```

```
Classification Report of the Logistic Regression Model:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00         5
         1.0       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
```

[52]:
```
cm =  confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, xticklabels=["False", "True"],␣
 ↪yticklabels=["Negative", "Posative"])
plt.title("Confusion Matrix: Logistic Regression Model")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()
```



3 Problem 2

[54]:
```
df2 = pd.read_csv("data2.csv")
df2.head()
```

[54]:

	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	

	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

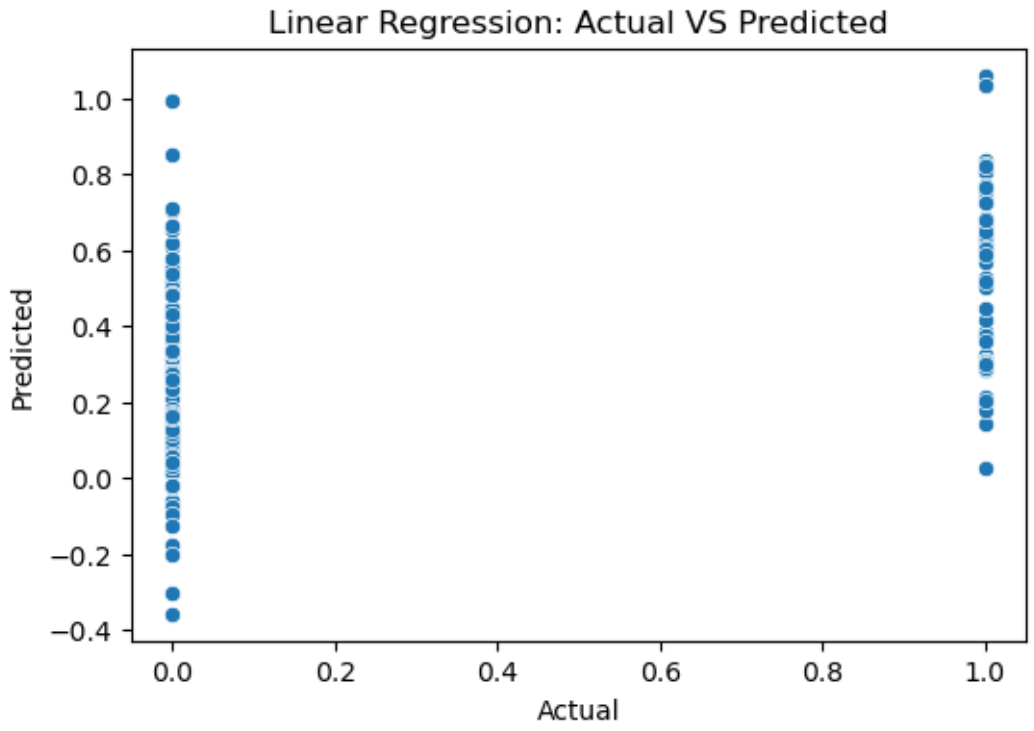
3.1 Linear Regression

[67]:
```
X2 = df2.drop(["Outcome"], axis=1)
y2 = df2["Outcome"]

X_train, X_test, y_train, y_test = train_test_split(X2, y2, test_size = 0.2,␣
↪random_state=42)

lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred_lin = lin_reg.predict(X_test)

plt.figure(figsize=(6,4))
sns.scatterplot(x=y_test, y=y_pred_lin)
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.title("Linear Regression: Actual VS Predicted")
plt.show()
```



3.2 Logistic Regression

[55]:
```
scaler2 = MinMaxScaler()
data_scaled2 = scaler2.fit_transform(df2)
df_scaled2 = pd.DataFrame(data_scaled2, columns=df2.columns)
df_scaled2.head()
```

[55]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	
1	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	
3	0.058824	0.447236	0.540984	0.232323	0.111111	0.418778	
4	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	

	DiabetesPedigreeFunction	Age	Outcome
0	0.234415	0.483333	1.0
1	0.116567	0.166667	0.0
2	0.253629	0.183333	1.0
3	0.038002	0.000000	0.0
4	0.943638	0.200000	1.0

[60]:
```
X2 = df_scaled2.drop(["Outcome"], axis=1)
y2 = df_scaled2["Outcome"]

X_train, X_test, y_train, y_test = train_test_split(X2, y2, test_size = 0.2,␣
 ↪random_state=42)
```

[61]:
```
LR2 = LogisticRegression()
LR2.fit(X_train, y_train)
```

[61]:
```
LogisticRegression()
```

[62]:
```
y_pred = LR2.predict(X_test)
```

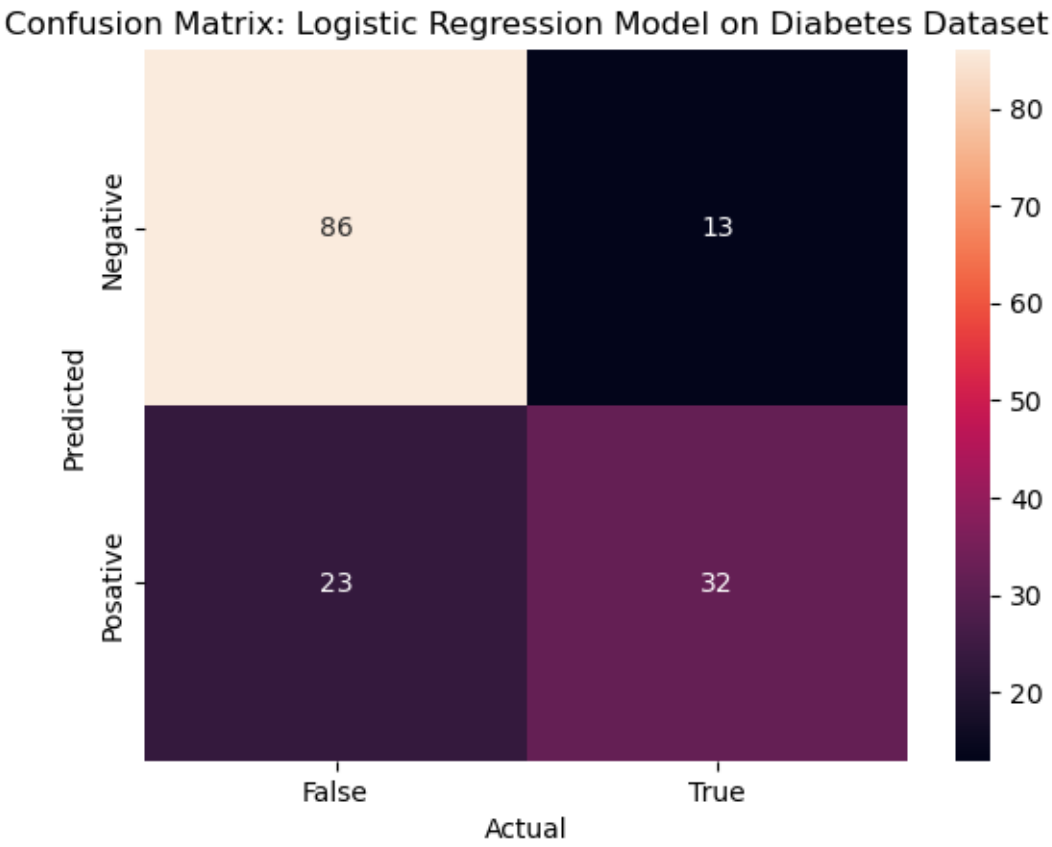
[64]:
```
print(f"Accuracy Score: {accuracy_score(y_test, y_pred)}")
print(f"\n Classification Report:\n {classification_report(y_test, y_pred)}")
cm2 = confusion_matrix(y_test, y_pred)
sns.heatmap(cm2, annot=True, xticklabels=["False", "True"],␣
 ↪yticklabels=["Negative", "Posative"])
plt.title("Confusion Matrix: Logistic Regression Model on Diabetes Dataset")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()
```

```
Accuracy Score: 0.7662337662337663

 Classification Report:
              precision    recall  f1-score   support
```

```
              0.0       0.79      0.87      0.83        99
              1.0       0.71      0.58      0.64        55

         accuracy                           0.77       154
        macro avg       0.75      0.73      0.73       154
     weighted avg       0.76      0.77      0.76       154
```



[ ]: