ML LAB ASSIGNMENT

Question 1

In [87]:

```
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.linear_model import LogisticRegression
df = pd.read_csv("C://Users/91947/OneDrive/Desktop/diabetes_zero.csv")
# Split the data into features (X) and target (y)
x = df.drop('Outcome', axis=1)
y = df['Outcome']

# Split the data into a training set and a testing set
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

# Initialize and train the logistic regression model
model = LogisticRegression(max_iter=1000)
model.fit(x_train, y_train)

# Make predictions on the test set
y_pred = model.predict(x_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"Accuracy of the logistic regression model: {accuracy:.6f}")
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy of the logistic regression model: 0.746753
Confusion Matrix:
[[78 21]
 [18 37]]
```

question no.4

In [6]:

```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error,r2_score
```

In [2]:

```
tips=sns.load_dataset("tips")
```

In [3]:

```
tips
```

Out[3]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [24]:

```
x=tips[['total_bill','size']]
y=tips["tip"].values
```

In [25]:

```
x
```

Out[25]:

	total_bill	size
0	16.99	2
1	10.34	3
2	21.01	3
3	23.68	2
4	24.59	4
...	...	...
239	29.03	3
240	27.18	2
241	22.67	2
242	17.82	2
243	18.78	2

244 rows × 2 columns

In [26]:

```
y
```

Out[26]:

```
array([ 1.01,  1.66,  3.5 ,  3.31,  3.61,  4.71,  2.  ,  3.12,  1.96,
        3.23,  1.71,  5.  ,  1.57,  3.  ,  3.02,  3.92,  1.67,  3.71,
        3.5 ,  3.35,  4.08,  2.75,  2.23,  7.58,  3.18,  2.34,  2.  ,
        2.  ,  4.3 ,  3.  ,  1.45,  2.5 ,  3.  ,  2.45,  3.27,  3.6 ,
        2.  ,  3.07,  2.31,  5.  ,  2.24,  2.54,  3.06,  1.32,  5.6 ,
        3.  ,  5.  ,  6.  ,  2.05,  3.  ,  2.5 ,  2.6 ,  5.2 ,  1.56,
        4.34,  3.51,  3.  ,  1.5 ,  1.76,  6.73,  3.21,  2.  ,  1.98,
        3.76,  2.64,  3.15,  2.47,  1.  ,  2.01,  2.09,  1.97,  3.  ,
        3.14,  5.  ,  2.2 ,  1.25,  3.08,  4.  ,  3.  ,  2.71,  3.  ,
        3.4 ,  1.83,  5.  ,  2.03,  5.17,  2.  ,  4.  ,  5.85,  3.  ,
        3.  ,  3.5 ,  1.  ,  4.3 ,  3.25,  4.73,  4.  ,  1.5 ,  3.  ,
        1.5 ,  2.5 ,  3.  ,  2.5 ,  3.48,  4.08,  1.64,  4.06,  4.29,
        3.76,  4.  ,  3.  ,  1.  ,  4.  ,  2.55,  4.  ,  3.5 ,  5.07,
        1.5 ,  1.8 ,  2.92,  2.31,  1.68,  2.5 ,  2.  ,  2.52,  4.2 ,
        1.48,  2.  ,  2.  ,  2.18,  1.5 ,  2.83,  1.5 ,  2.  ,  3.25,
        1.25,  2.  ,  2.  ,  2.  ,  2.75,  3.5 ,  6.7 ,  5.  ,  5.  ,
        2.3 ,  1.5 ,  1.36,  1.63,  1.73,  2.  ,  2.5 ,  2.  ,  2.74,
        2.  ,  2.  ,  5.14,  5.  ,  3.75,  2.61,  2.  ,  3.5 ,  2.5 ,
        2.  ,  2.  ,  3.  ,  3.48,  2.24,  4.5 ,  1.61,  2.  , 10.  ,
        3.16,  5.15,  3.18,  4.  ,  3.11,  2.  ,  2.  ,  4.  ,  3.55,
        3.68,  5.65,  3.5 ,  6.5 ,  3.  ,  5.  ,  3.5 ,  2.  ,  3.5 ,
        4.  ,  1.5 ,  4.19,  2.56,  2.02,  4.  ,  1.44,  2.  ,  5.  ,
        2.  ,  2.  ,  4.  ,  2.01,  2.  ,  2.5 ,  4.  ,  3.23,  3.41,
        3.  ,  2.03,  2.23,  2.  ,  5.16,  9.  ,  2.5 ,  6.5 ,  1.1 ,
        3.  ,  1.5 ,  1.44,  3.09,  2.2 ,  3.48,  1.92,  3.  ,  1.58,
        2.5 ,  2.  ,  3.  ,  2.72,  2.88,  2.  ,  3.  ,  3.39,  1.47,
        3.  ,  1.25,  1.  ,  1.17,  4.67,  5.92,  2.  ,  2.  ,  1.75,
        3.  ])
```

In [27]:

```
#train_test split

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

Type *Markdown* and LaTeX: α^2

In [29]:

```
model = LinearRegression()
model.fit(x_train, y_train)
```

Out[29]:

```
▾ LinearRegression
LinearRegression()
```

In [30]:

```
y_pred = model.predict(x_test)
```

In [31]:

```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
```

In [32]:

```
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")
```

```
Mean Absolute Error (MAE): 0.66
Mean Squared Error (MSE): 0.65
Root Mean Squared Error (RMSE): 0.81
R-squared (R2) Score: 0.48
```

question 3

In [46]:

```
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.linear_model import LogisticRegression
```

In [38]:

```
df = pd.read_csv("C://Users/91947/OneDrive/Desktop/diabetes_null.csv")
```

In [39]:

```
df
```

Out[39]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFun
0	6	148.0	72.0	35.0	NaN	33.6	
1	1	85.0	66.0	29.0	NaN	26.6	
2	8	183.0	64.0	NaN	NaN	23.3	
3	1	89.0	66.0	23.0	94.0	28.1	
4	0	137.0	4.0	35.0	168.0	43.1	
...	...	...	...	...	...	...	
763	10	11.0	76.0	48.0	18.0	32.9	
764	2	122.0	7.0	27.0	NaN	36.8	
765	5	121.0	72.0	23.0	112.0	26.2	
766	1	126.0	6.0	NaN	NaN	3.1	
767	1	93.0	7.0	31.0	NaN	3.4	

768 rows × 9 columns

In [72]:

```
df.fillna(df.mean(), inplace=True)
```

In [73]:

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

In [74]:

```
x
```

Out[74]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigre
0	6	148.0	72.0	35.000000	105.659898	33.6	
1	1	85.0	66.0	29.000000	105.659898	26.6	
2	8	183.0	64.0	25.876155	105.659898	23.3	
3	1	89.0	66.0	23.000000	94.000000	28.1	
4	0	137.0	4.0	35.000000	168.000000	43.1	
...	...	...	...	...	...	...	
763	10	11.0	76.0	48.000000	18.000000	32.9	
764	2	122.0	7.0	27.000000	105.659898	36.8	
765	5	121.0	72.0	23.000000	112.000000	26.2	
766	1	126.0	6.0	25.876155	105.659898	3.1	
767	1	93.0	7.0	31.000000	105.659898	3.4	

768 rows × 8 columns

In [75]:

```
y
```

Out[75]:

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [76]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

In [77]:

```
model = LogisticRegression(max_iter=1000)
model.fit(x_train, y_train)
```

Out[77]:

```
▾        LogisticRegression

LogisticRegression(max_iter=1000)
```

In [78]:

```
y_pred = model.predict(x_test)
```

In [79]:

```
accuracy = accuracy_score(y_test, y_pred)
```

In [80]:

```
conf_matrix = confusion_matrix(y_test, y_pred)
```

In [81]:

```
print(f"Accuracy of the logistic regression model: {accuracy:.6f}")
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy of the logistic regression model: 0.753247
Confusion Matrix:
[[93 12]
 [26 23]]
```

question 2

In [90]:

```
df1 = pd.read_csv("C://Users/91947/OneDrive/Desktop/diabetes_null.csv")
df1.dropna(inplace=True)

# Split the data into features (X) and target (y)
x1 = df1.drop('Outcome', axis=1)
y1 = df1['Outcome']

# Split the data into a training set and a testing set
x1_train, x1_test, y1_train, y1_test = train_test_split(x1, y1, test_size=0.2, random_sta

# Initialize and train the logistic regression model
model1 = LogisticRegression(max_iter=1000)
model1.fit(x1_train, y1_train)

# Make predictions on the test set
y1_pred = model.predict(x1_test)

# Calculate the accuracy of the model
accuracy1 = accuracy_score(y1_test, y1_pred)
conf_matrix1 = confusion_matrix(y1_test, y1_pred)

print(f"Accuracy of the logistic regression model: {accuracy1:.6f}")

print("Confusion Matrix:")
print(conf_matrix1)
```

```
Accuracy of the logistic regression model: 0.759494
Confusion Matrix:
[[46  6]
 [13 14]]
```

In [85]:

In [ ]: