```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

plt.rcParams['figure.figsize'] = (20.0,10.0)
```

```
#upload data
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
data = pd.read_csv("heart_attack_data.csv")
print(data.shape)
print(data.head())
```

```
(11, 2)
   Heart Rate Heart Attack
0          50            Y
1          50            N
2          50            N
3          50            N
4          70            N
```

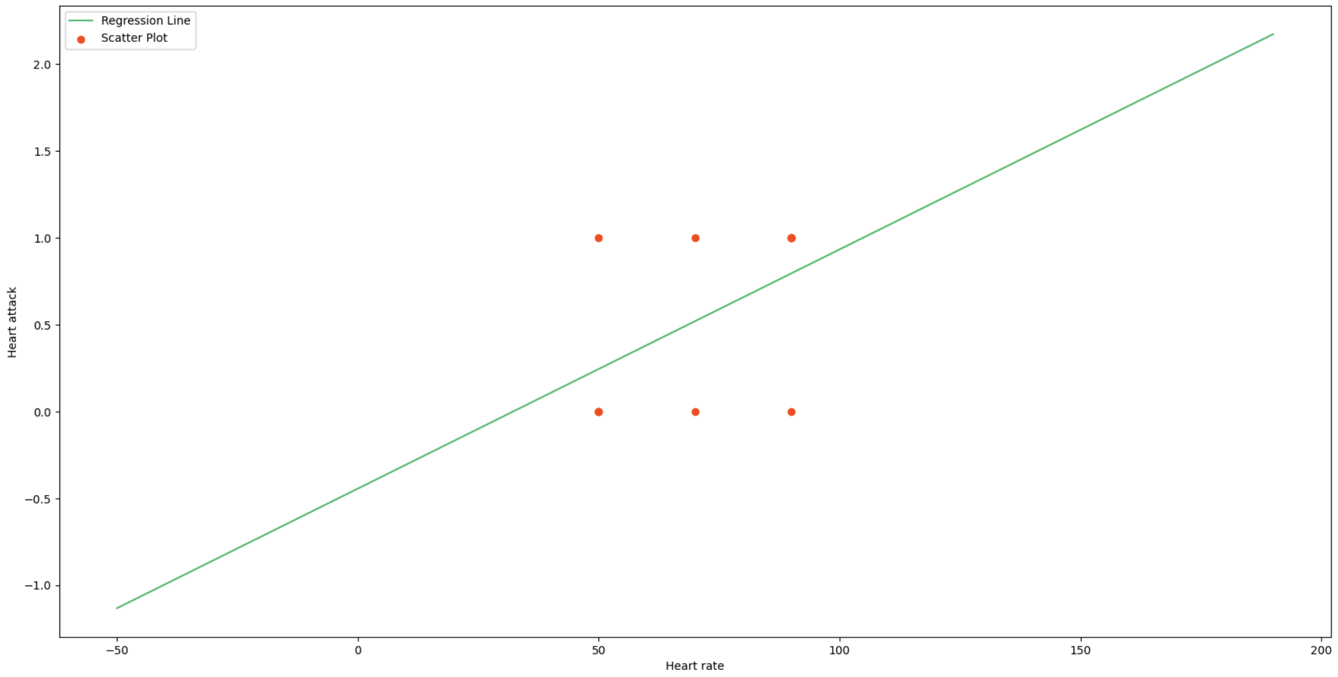
```
data['Heart Attack'] = data['Heart Attack'].map({'Y': 1, 'N': 0})
```

```
X = data[['Heart Rate']].values   # must be 2D
y = data['Heart Attack'].values   # 1D
```

```
MEAN_X = np.mean(X)
MEAN_Y = np.mean(y)

m=len(X)

numer=0
denom=0
for i in range(m):
  numer += (X[i]-MEAN_X)*(y[i]-MEAN_Y)
  denom += (X[i]-MEAN_X)**2

b1=numer/denom
b0 = MEAN_Y - (b1*MEAN_X)

print(b1,b0)
```

```
[0.01377551] [-0.44387755]
```

```
max_x = np.max(X)+100
min_x = np.min(X)-100

x = np.linspace(min_x,max_x,1000)
y_reg = b0 + b1*x # Renamed y to y_reg for clarity

plt.plot(x,y_reg,color='#58b970',label='Regression Line') # Use y_reg for the regression line
plt.scatter(X,y,c='#ef5423',label='Scatter Plot') # Use original X and y for the scatter plot
plt.xlabel('Heart rate')
plt.ylabel('Heart attack')
plt.legend()
plt.show()
```



```
ss_t =0
ss_r =0
for i in range(m):
  y_pred = b0 + b1*X[i]
  ss_t += (y[i]-MEAN_Y)**2
  ss_r += (y[i]-y_pred)**2
r2=1-(ss_r/ss_t)
print(r2)
```

```
[0.24795918]
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

x = X.reshape(-1,1)


lin_reg = LinearRegression()
lin_reg.fit(x,y)

y_pred = lin_reg.predict(x)

r2_score = lin_reg.score(x,y)
print(r2_score)
```

```
0.24795918367346914
```

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
log_reg.fit(X, y)
```

```
▾ LogisticRegression ⓘ ?
LogisticRegression()
```

```
b0 = log_reg.intercept_[0]
b1 = log_reg.coef_[0][0]
```

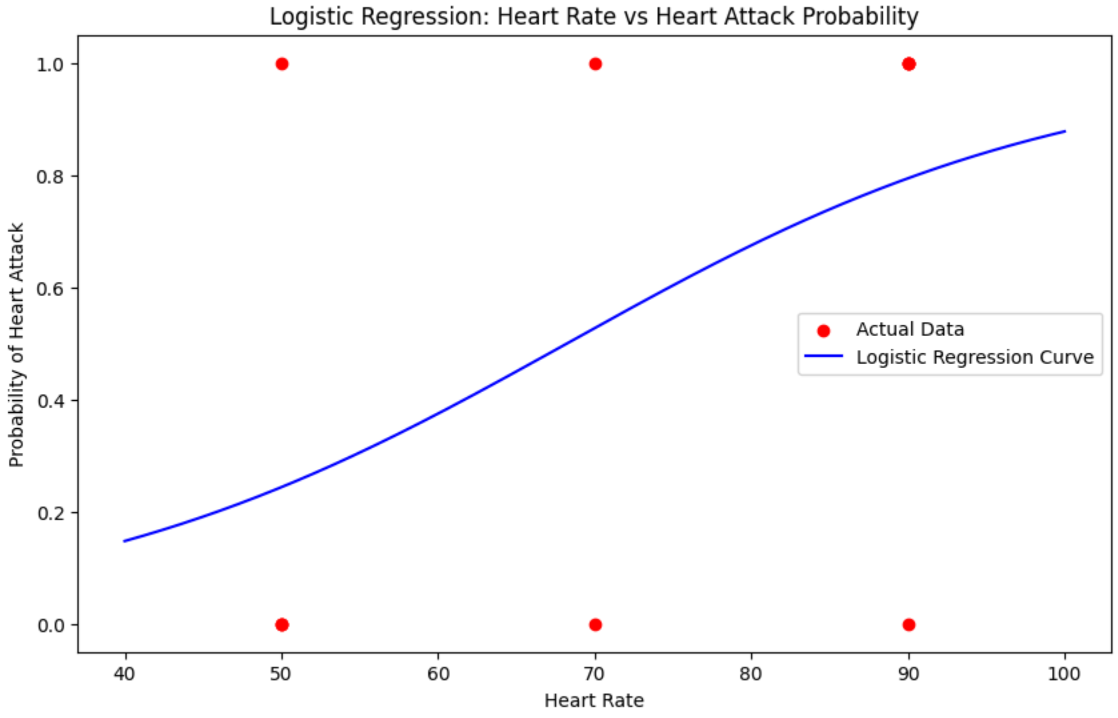
```
print("Intercept (b0):", b0)
print("Coefficient (b1):", b1)
```

```
Intercept (b0): -4.2378870732374985
Coefficient (b1): 0.06209602448014388
```

```
heart_rate = np.array([[60]])
prob = log_reg.predict_proba(heart_rate)[0][1]

print(f"Predicted probability of heart attack for heart rate 60: {prob:.4f}")
```

```
Predicted probability of heart attack for heart rate 60: 0.3747
```

```
x_vals = np.linspace(40, 100, 200).reshape(-1, 1)
y_probs = log_reg.predict_proba(x_vals)[:, 1]

plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='red', label='Actual Data')
plt.plot(x_vals, y_probs, color='blue', label='Logistic Regression Curve')
plt.xlabel("Heart Rate")
plt.ylabel("Probability of Heart Attack")
plt.legend()
plt.title("Logistic Regression: Heart Rate vs Heart Attack Probability")
plt.show()
```


Logistic Regression: Heart Rate vs Heart Attack Probability
Probability of Heart Attack
Heart Rate
● Actual Data
Logistic Regression Curve