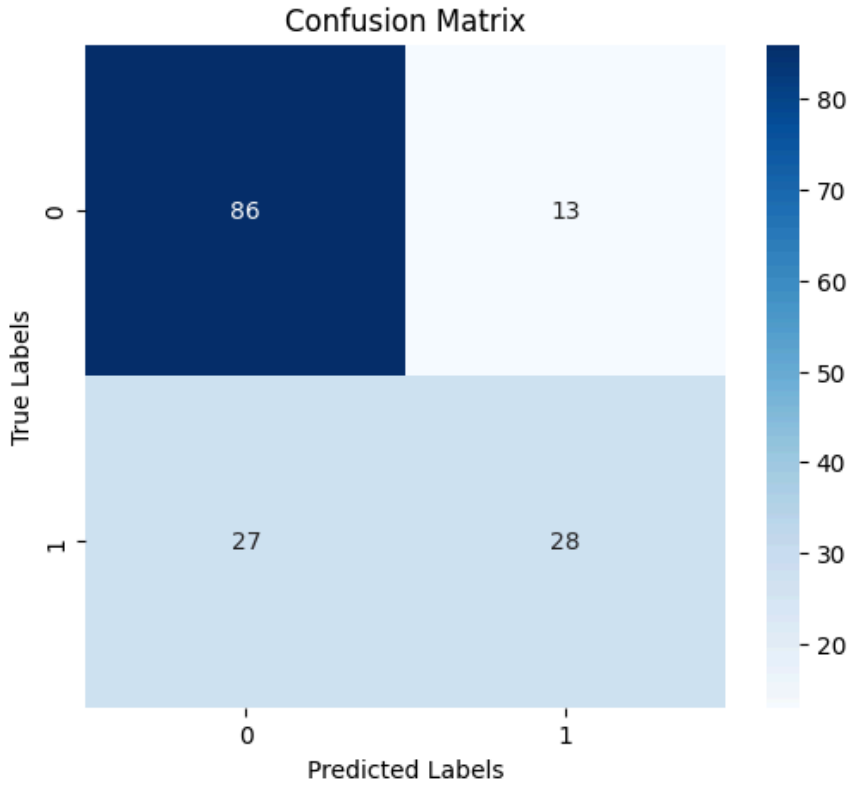
```
print('hello')
```

```
hello
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

file_path = input("Enter the path to your CSV file: ")
df = pd.read_csv(file_path)

# Automatically select the target column (assuming it's the last column)
target_column = df.columns[-1]

# Splitting features (X) and target variable (y)
X = df.iloc[:, :-1]  # All columns except last (features)
y = df.iloc[:, -1]   # Last column (target)

# Encode categorical target variable if necessary
if y.dtype == 'object':
    label_encoder = LabelEncoder()
    y = label_encoder.fit_transform(y)

# Convert categorical features to numerical if any
X = pd.get_dummies(X, drop_first=True)

# Split the dataset (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Determine optimal k (square root heuristic)
k = int(np.sqrt(len(y_train)))
if k % 2 == 0:
    k += 1

# Train the KNN classifier
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)

# Make predictions
y_pred = knn.predict(X_test)

# Display accuracy score
accuracy = accuracy_score(y_test, y_pred)
print(f"\nAccuracy Score: {accuracy:.2f}")

# Display confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:\n", conf_matrix)

# Display classification report
class_report = classification_report(y_test, y_pred)
print("\nClassification Report:\n", class_report)
```

```
# Plot Confusion Matrix
plt.figure(figsize=(6,5))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=np.unique(y), yticklabels=np.uniqu
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```

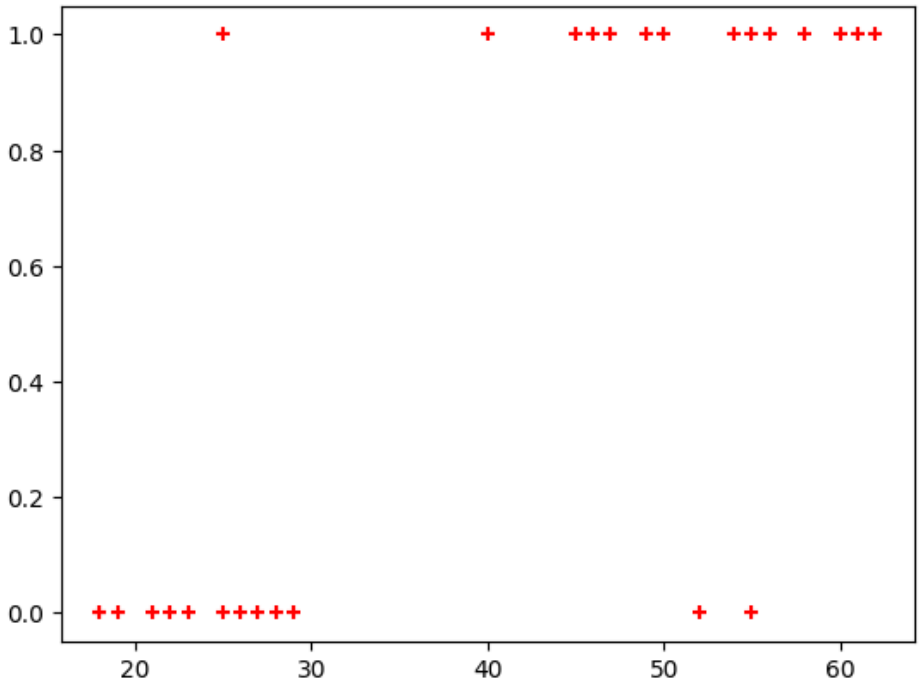
```
Enter the path to your CSV file: /content/diabetes.csv

Accuracy Score: 0.74

Confusion Matrix:
 [[86 13]
 [27 28]]

Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.87      0.81        99
           1       0.68      0.51      0.58        55

    accuracy                           0.74       154
   macro avg       0.72      0.69      0.70       154
weighted avg       0.73      0.74      0.73       154
```




```
# Commented out IPython magic to ensure Python compatibility.
import pandas as pd
from matplotlib import pyplot as plt
# %matplotlib inline
#"%matplotlib inline" will make your plot outputs appear and be stored within the notebook.

df = pd.read_csv("/content/insurance_data.csv")
df.head()
```

```
plt.scatter(df.age,df.bought_insurance,marker='+',color='red')

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df[['age']],df.bought_insurance,train_size=0.9,rando
X_train.shape

X_test

from sklearn.linear_model import LogisticRegression
model = LogisticRegression()

model.fit(X_train, y_train)

X_test

y_test

y_predicted = model.predict(X_test)
y_predicted

model.score(X_test,y_test)

model.predict_proba(X_test)

y_predicted = model.predict([[60]])
y_predicted

#model.coef_ indicates value of m in y=m*x + b equation
model.coef_

#model.intercept_ indicates value of b in y=m*x + b equation
model.intercept_

#Lets defined sigmoid function now and do the math with hand
import math
def sigmoid(x):
  return 1 / (1 + math.exp(-x))

def prediction_function(age):
   z = 0.127 * age - 4.973 # 0.12740563 ~ 0.0127 and -4.97335111 ~ -4.97
   y = sigmoid(z)
   return y

age = 35
prediction_function(age)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not ha
  warnings.warn(
0.3709834769552775
```



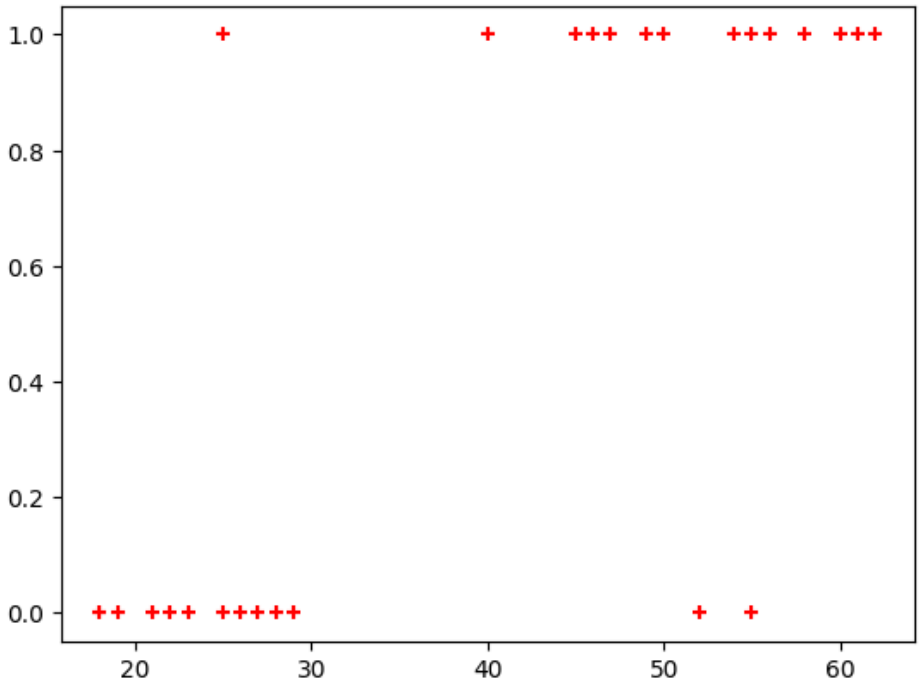

```
import pandas as pd
from matplotlib import pyplot as plt

df = pd.read_csv("/content/insurance_data.csv")
df.head()
```

	age	bought_insurance
0	22	0
1	25	0
2	47	1
3	52	0
4	46	1

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
plt.scatter(df.age, df.bought_insurance, marker='+', color='red')
```

```
<matplotlib.collections.PathCollection at 0x7c2fe3ff4f50>
```




```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df[['age']], df.bought_insurance, train_size=0.9, ra
```

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression()
```

```
y_predicted = model.predict(X_test)
```

```
model.score(X_test, y_test)
```

```
1.0
```

```
model.predict_proba(X_test)
```

```
array([[0.06470655, 0.93529345],
       [0.10327333, 0.89672667],
       [0.92775258, 0.07224742]])
```

```
y_predicted = model.predict([[60]])
y_predicted
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not ha
  warnings.warn(
array([1])
```

```
model.coef_   # Value of m
model.intercept_   # Value of b
```

```
array([-4.97339194])
```

```
import math
def sigmoid(x):
  return 1 / (1 + math.exp(-x))
```

```
def prediction_function(age):
   z = 0.127 * age - 4.973  # Approximate model parameters
```