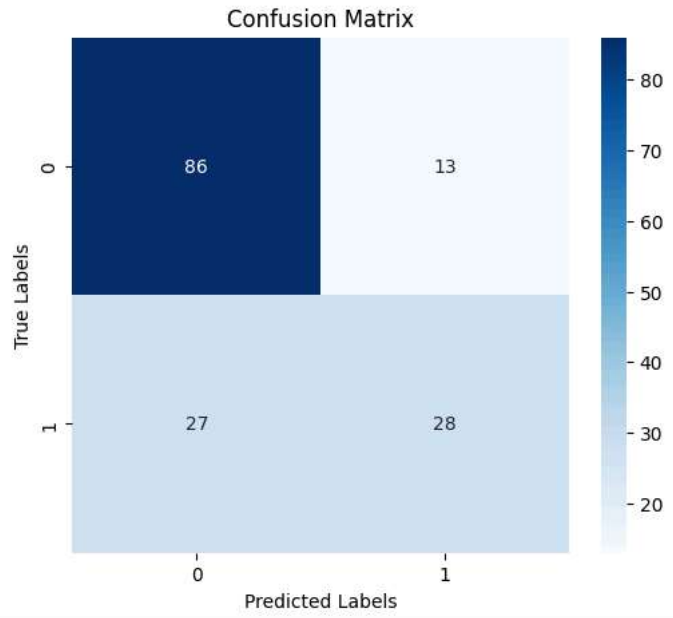
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, 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.unique(y))
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```

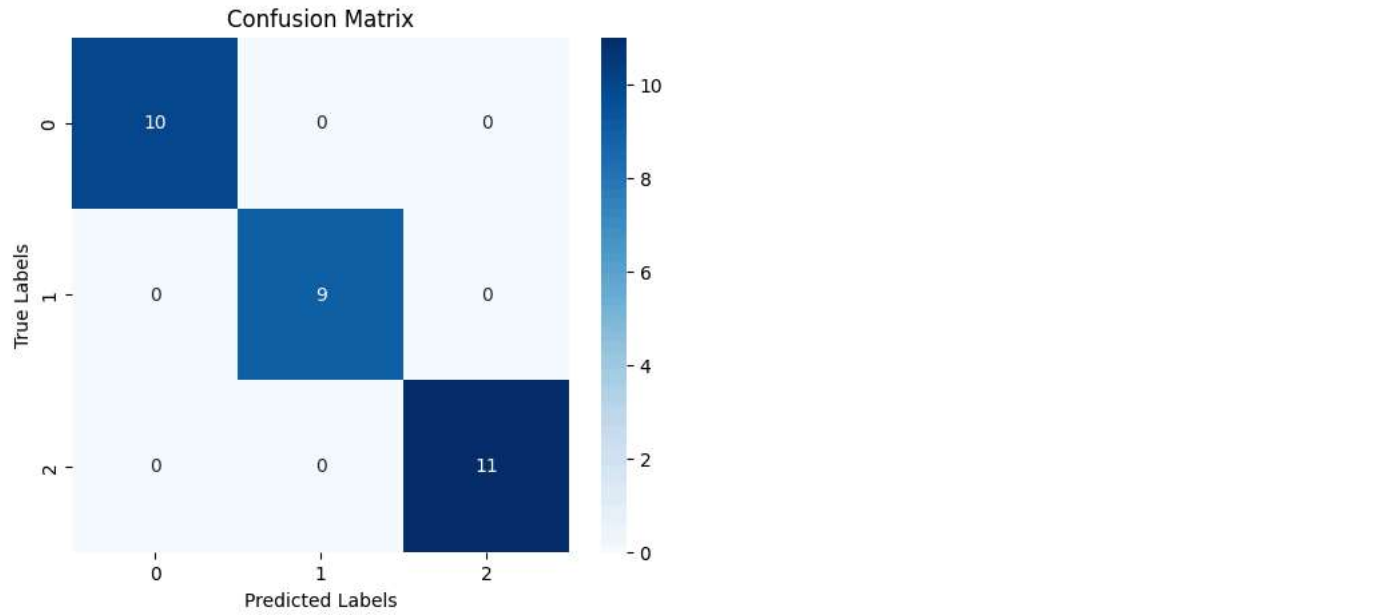
```
Enter the path to your CSV file: diabetes.csv

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
```




```
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.unique(y))
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```

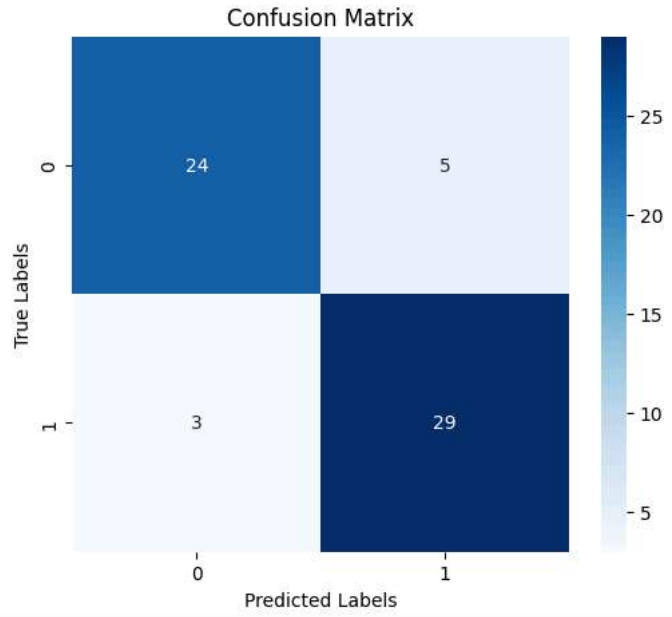
```
Enter the path to your CSV file: iris (2).csv

Accuracy Score: 1.00

Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```




```
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.unique(y))
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```

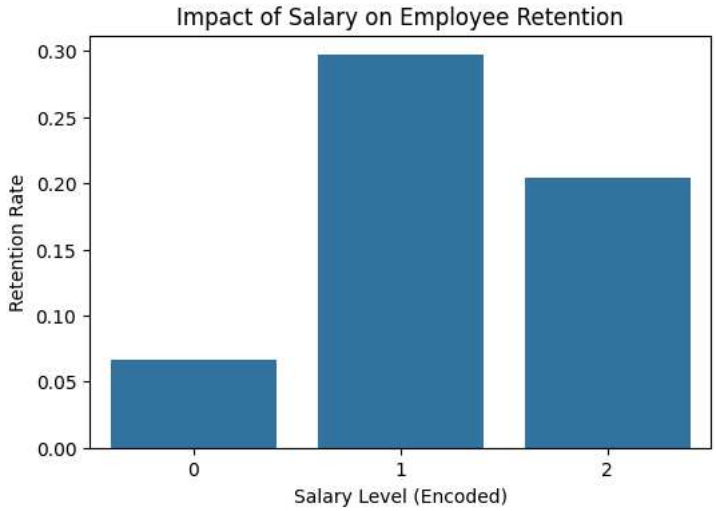
```
Enter the path to your CSV file: heart.csv

Accuracy Score: 0.87

Confusion Matrix:
 [[24  5]
 [ 3 29]]

Classification Report:
               precision    recall  f1-score   support

           0       0.89      0.83      0.86        29
           1       0.85      0.91      0.88        32

    accuracy                           0.87        61
   macro avg       0.87      0.87      0.87        61
weighted avg       0.87      0.87      0.87        61
```



LOGISTIC REGRESSION

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

# Load dataset
df = pd.read_csv("HR_comma_sep.csv")

# Convert categorical variables to numerical
label_enc = LabelEncoder()
df["salary"] = label_enc.fit_transform(df["salary"])
df["Department"] = label_enc.fit_transform(df["Department"])

# Step 1: Exploratory Data Analysis (EDA)
correlation = df.corr()["left"].sort_values(ascending=False)
print("\nFeature Correlation with Employee Retention:\n", correlation)

# Step 2: Impact of Salary on Retention
plt.figure(figsize=(6,4))
sns.barplot(x="salary", y="left", data=df, ci=None)
plt.xlabel("Salary Level (Encoded)")
plt.ylabel("Retention Rate")
plt.title("Impact of Salary on Employee Retention")
plt.show()

# Step 3: Correlation between Department and Retention
plt.figure(figsize=(8,4))
sns.barplot(x="Department", y="left", data=df, ci=None)
plt.xlabel("Department (Encoded)")
plt.ylabel("Retention Rate")
plt.title("Department vs Employee Retention")
plt.show()

# Step 4: Logistic Regression Model
features = ["satisfaction_level", "last_evaluation", "number_project", "average_montly_hours", "time_spend_company", "salary"]
X = df[features]
y = df["left"]

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

model = LogisticRegression()
model.fit(X_train, y_train)

# Step 5: Model Accuracy
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"\nModel Accuracy: {accuracy:.2f}")

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

# Display Classification Report
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Feature Correlation with Employee Retention:
 left                    1.000000
time_spend_company       0.144822
average_montly_hours     0.071287
Department               0.032105
number_project           0.023787
last_evaluation          0.006567
salary                  -0.001294
promotion_last_5years   -0.061788
Work_accident           -0.154622
satisfaction_level      -0.388375
Name: left, dtype: float64
<ipython-input-18-d5b96e519139>:24: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x="salary", y="left", data=df, ci=None)
```




```
<ipython-input-18-d5b96e519139>:32: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x="Department", y="left", data=df, ci=None)
```