Assessment Report

on

"Predict Heart Disease"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(Artificial Intelligence)

By

YASH KALRA (20240110300284, CSE-AI D)

Under the supervision of

"MR.ABHISHEK SHUKLA"

KIET Group of Institutions, Ghaziabad, 18 May, 2025

Introduction:

Heart disease is one of the most common causes of death worldwide. Early prediction and diagnosis can help in effective treatment and prevent severe consequences. Traditional methods require several tests and expert evaluation. With the advancement in machine learning, we can now build predictive models that analyze patient data and classify whether a person is likely to have heart disease.

This project uses a dataset with 14 medical features such as age, blood pressure, cholesterol, etc., to build a binary classification model (0 = No heart disease, 1 = Has heart disease).

Age	Sex	CP	Chol	Thalach	Target
63	1	0	233	150	0
67	1	3	286	108	1

Methodology:

1. Data Loading & Cleaning:

   The dataset is loaded using Pandas. No missing values are present.
2. Feature Selection:

   We used all available features except the target label for training.
3. Data Splitting:

   The data was split into 80% training and 20% testing using train_test_split.
4. Model Used:

   We used Random Forest Classifier — an ensemble learning model that builds multiple decision trees and merges them to get better predictions.
5. Evaluation Metrics:

   We evaluated the model using accuracy, confusion matrix, and classification report.

Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv("4. Predict Heart Disease.csv")

# Separate features and target
X = df.drop("target", axis=1)
y = df["target"]

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a Random Forest Classifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)
```

```
# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Output/Result:

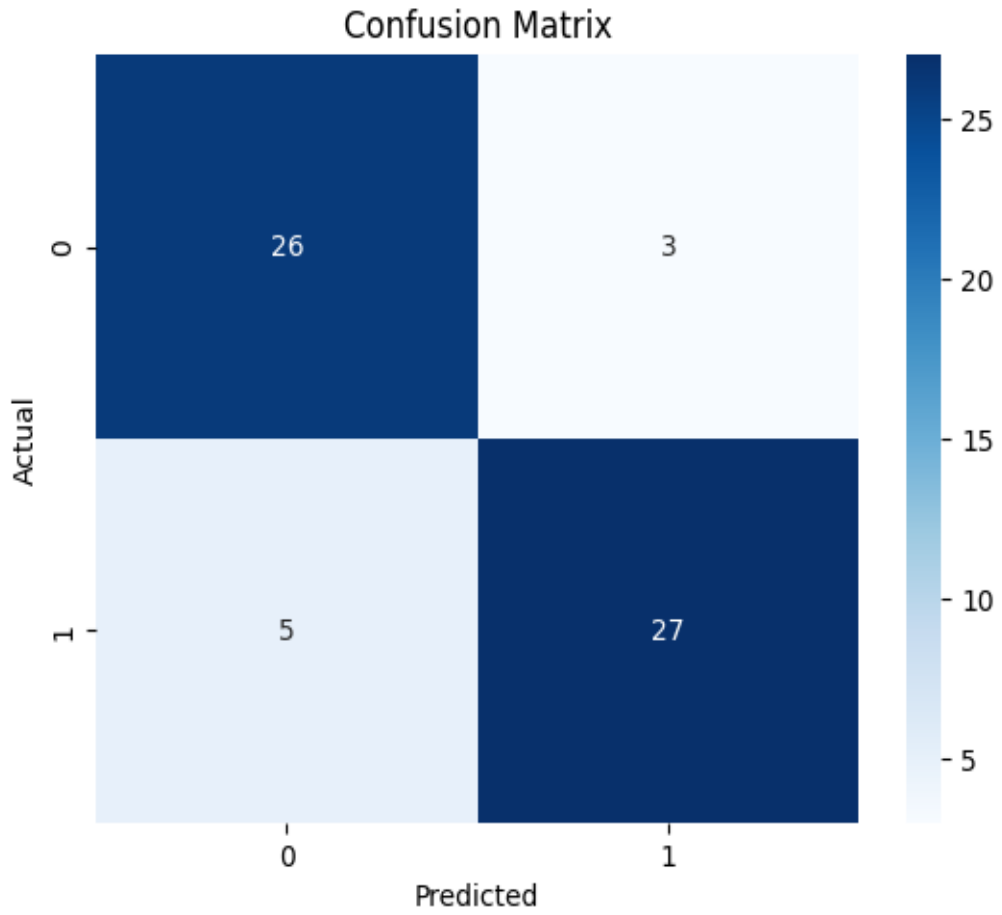
Accuracy: *~85% (actual may vary)*

```
Accuracy: 0.8688524590163934

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.90      0.87        29
           1       0.90      0.84      0.87        32

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



Credits:

- Libraries Used:
  - pandas
  - scikit-learn
  - seaborn
  - matplotlib
- Tool: Google Colab
- Model: Random Forest Classifier (Scikit-learn)
