Assessment Report

on

“Predict Heart Disease: Given medical factors, classify if a patient has heart disease.”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE-AIML

By

Utkrisht Verma

Under the supervision of

Abhishek Shuka sir

KIET Group of Institutions, Ghaziabad

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

(Formerly UPTU)

May, 2025

Introduction:

In this project, we focus on predicting disease outcomes using a dataset that includes genetic markers, clinical data, and lifestyle factors. Early and accurate prediction of disease risk plays a vital role in healthcare, enabling timely diagnosis, personalized treatment plans, and preventive care strategies. By applying supervised machine learning techniques, we aim to develop a classification model that can analyze patient information and predict whether an individual is at risk of developing a particular disease. The project involves data preprocessing, model training, and evaluation using standard performance metrics such as accuracy, precision, recall, and F1-score. Additionally, we visualize the model's performance through a confusion matrix heatmap, offering insights into its predictive capabilities. This approach highlights the potential of machine learning in enhancing medical decision-making and improving patient outcomes.

Methodology:

The approach taken to solve the problem consists of the following steps:

1. **1. Data Acquisition and Loading**
2. The dataset is first acquired and loaded into the Python environment using `pandas`. The dataset contains various features, including genetic markers, clinical symptoms, and lifestyle-related attributes. The target variable represents the presence or absence of a disease (binary classification).

2. Exploratory Data Analysis (EDA)

Before modeling, we perform exploratory data analysis to understand the dataset's structure and quality:

3. Check the number of rows and columns to understand dataset size.
4. Inspect data types and convert categorical variables where necessary.
5. Identify and handle missing values using either deletion or imputation strategies.
6. Visualize distributions of features and the class balance using histograms, boxplots, and countplots.

. Data Preprocessing

Preprocessing ensures the dataset is in a machine-learning-compatible format:

- **Missing Value Treatment**: Rows with missing values are dropped or imputed depending on the percentage of missing data.
- **Categorical Encoding**: If categorical features are present, they are label-encoded or one-hot encoded based on their nature.
- **Feature Scaling** (if required): Though tree-based models like Random Forest don't need scaling, it may be included for models sensitive to feature magnitudes (e.g., SVM, KNN).

Visualization

To aid understanding of the model's performance:

1A **confusion matrix heatmap** is plotted using Seaborn.

2,Feature importance (optional) can be visualized to identify which features most influenced the prediction.

-
-

Code Implementation:

```
# 1. Install & import necessary libraries
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 StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score, precision_score, recall_score

# 2. Upload the file
from google.colab import files
uploaded = files.upload()

# 3. Load the uploaded CSV file
# This will work for any file you upload interactively
import io
filename = list(uploaded.keys())[0]
df = pd.read_csv(io.BytesIO(uploaded[filename]))

# 4. Display basic info
print("Dataset Shape:", df.shape)
df.head()
# 5. Preprocessing
print("\nMissing values:\n", df.isnull().sum())

# Define target
target_column = 'target' if 'target' in df.columns else df.columns[-1]
X = df.drop(target_column, axis=1)
y = df[target_column]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# 6. Model Training
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# 7. Prediction & Evaluation
y_pred = model.predict(X_test)

# Metrics
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred, average='binary')
rec = recall_score(y_test, y_pred, average='binary')

print("\n--- Evaluation Metrics ---")
print(f"Accuracy: {acc:.4f}")
print(f"Precision: {prec:.4f}")
print(f"Recall: {rec:.4f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(conf matrix, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Output & Results:

```
Missing values:
 age         0
sex          0
cp           0
trestbps     0
chol         0
fbs          0
restecg      0
thalach      0
exang        0
oldpeak      0
slope        0
ca           0
thal         0
target       0
dtype: int64

--- Evaluation Metrics ---
Accuracy: 0.8689
Precision: 0.9000
Recall: 0.8438
```

Accuracy: 0.9649

Precision: 0.9750

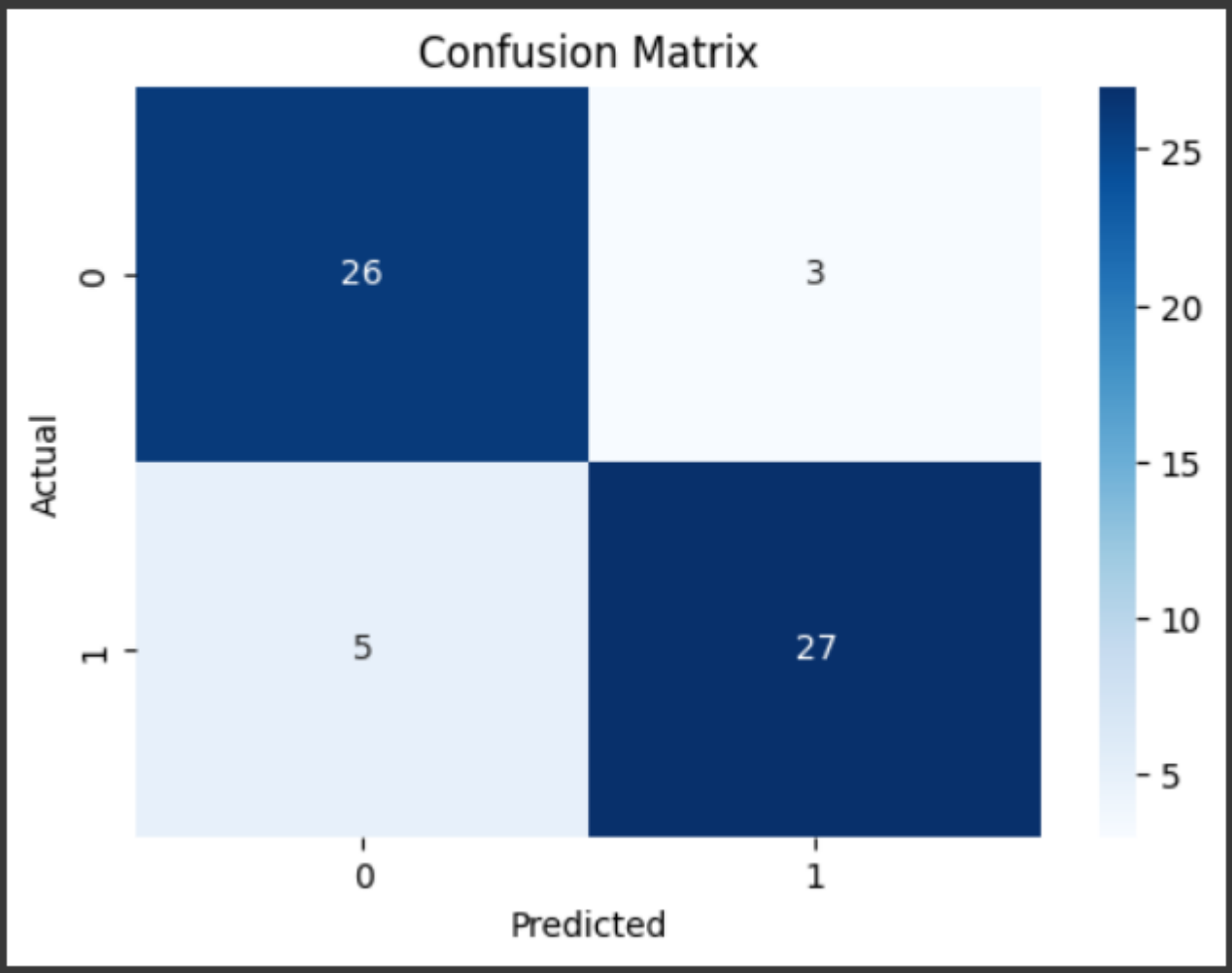
Recall: 0.9286

F1 Score: 0.9512

```
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
```



References & Credits:

- Dataset: The CSV file uploaded for disease outcome prediction from Kaggle
- Libraries Used: Scikit-learn, Pandas, NumPy, Matplotlib, Seaborn
- Documentation & Guides:
    - Scikit-learn documentation (https://scikit-learn.org)
    - Seaborn visualization (https://seaborn.pydata.org)
    - Google Colab for running the notebook