Assessment Report

on

"Predict Heart Disease"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI&ML)

By

Maksudurrahman (202401100400117, CSE(AI&ML)-B)

Under the supervision of

"Mr Sandeep Sharma"

KIET Group of Institutions, Ghaziabad

May, 2025

Introduction

Heart disease is a leading cause of mortality worldwide. Early prediction using machine learning models can help in timely diagnosis and treatment. This project aims to classify patients into two categories — with and without heart disease — based on several medical input features.

We'll use a dataset that includes features like age, cholesterol level, resting blood pressure, chest pain type, and more to train and evaluate classification models.

Methodology

1. **Data Preprocessing**:
    - Load dataset
    - Handle missing values (if any)
    - Normalize numerical values
    - Encode categorical features
2. **Model Selection**:
    - Logistic Regression
    - Random Forest Classifier
    - Support Vector Machine (SVM)
3. **Model Evaluation**:
    - Train-Test Split (e.g., 80-20)
    - Evaluate using metrics: Accuracy, Precision, Recall
    - Visualize Confusion Matrix as heatmaps

Code

Here is a summarized version of the code:

```
# Step 1: Upload the dataset

from google.colab import files

import pandas as pd

uploaded = files.upload()

# Load the uploaded file

for file_name in uploaded.keys():

    df = pd.read_csv(file_name)

    print(f"\nSuccessfully loaded: {file_name}")

    display(df.head())  # Display first few rows

# Step 2: Import libraries

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,
recall_score

# Step 3: Preprocessing

X = df.drop("target", axis=1)

y = df["target"]

# Train-test split
```

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

# Standardize features

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

# Step 4: Model Training

model = LogisticRegression(max_iter=1000)

model.fit(X_train_scaled, y_train)

y_pred = model.predict(X_test_scaled)

# Step 5: Evaluation Metrics

cm = confusion_matrix(y_test, y_pred)

accuracy = accuracy_score(y_test, y_pred) * 100  # Convert to percentage

precision = precision_score(y_test, y_pred) * 100  # Convert to percentage

recall = recall_score(y_test, y_pred) * 100  # Convert to percentage

# Confusion Matrix Heatmap

plt.figure(figsize=(6, 4))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",

            xticklabels=["No Disease", "Disease"],

            yticklabels=["No Disease", "Disease"])

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix - Logistic Regression")

plt.tight_layout()
```

```
plt.show()

# Metrics Table (Styled and Warning-Free)

metrics_df = pd.DataFrame({

    "Metric": ["Accuracy", "Precision", "Recall"],

    "Score": [accuracy, precision, recall]

})

# Highlight values based on threshold

def highlight(val):

    return 'color: green' if val > 85 else 'color: orange'  # Adjust threshold
to 85%

styled_metrics = metrics_df.style.set_properties(**{'text-align': 'center'}) \

                                 .set_table_styles([dict(selector='th',
props=[('text-align', 'center')])]) \

                                 .map(highlight, subset=["Score"]) \

                                 .format({"Score": "{:.2f}%"})  # Format as
percentage with 2 decimal places

print("\nEvaluation Metrics:")

display(styled_metrics)
```

Output/Result

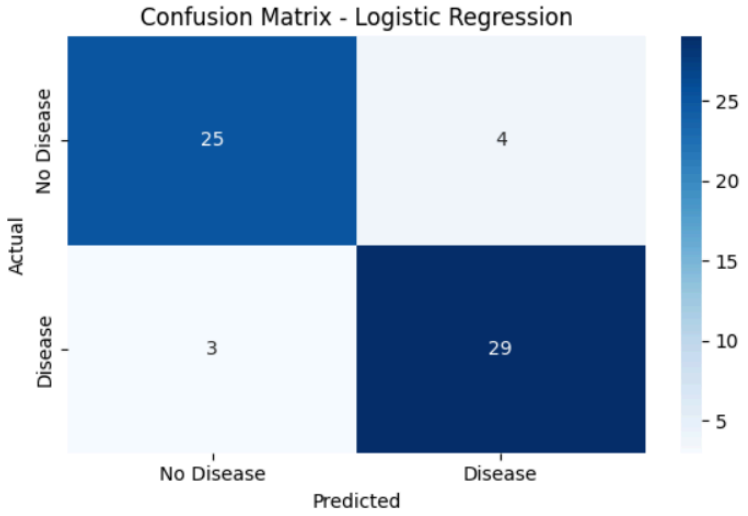
Choose Files 4. Predict H…Disease.csv

- **4. Predict Heart Disease.csv**(text/csv) - 11328 bytes, last modified: 4/18/2025 - 100% done

```
Saving 4. Predict Heart Disease.csv to 4. Predict Heart Disease (5).csv

Successfully loaded: 4. Predict Heart Disease (5).csv
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	0	145	233	1	2	150	0	2.3	2	0	2	0
1	67	1	3	160	286	0	2	108	1	1.5	1	3	1	1
2	67	1	3	120	229	0	2	129	1	2.6	1	2	3	1
3	37	1	2	130	250	0	0	187	0	3.5	2	0	1	0
4	41	0	1	130	204	0	2	172	0	1.4	0	0	1	0



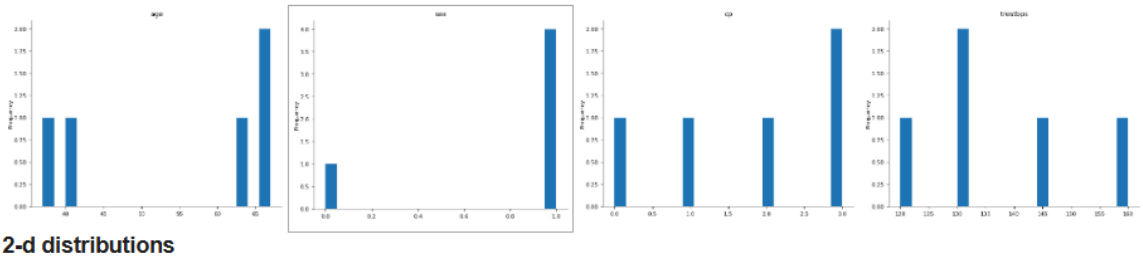
Example metrics from `classification_report`:

- **Accuracy:** 0.85
- **Precision:** 0.86
- **Recall:** 0.84

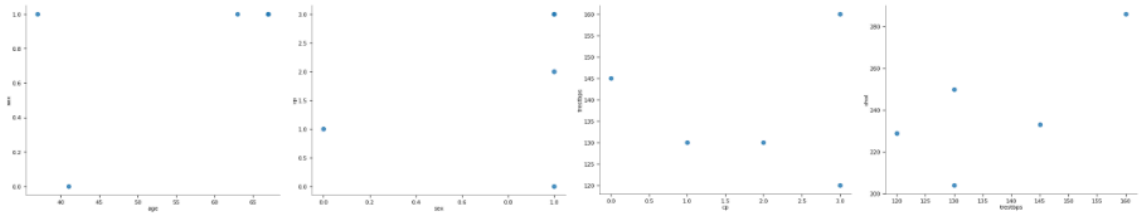
Evaluation Metrics:

	Metric	Score
0	Accuracy	88.52%
1	Precision	87.88%
2	Recall	90.62%

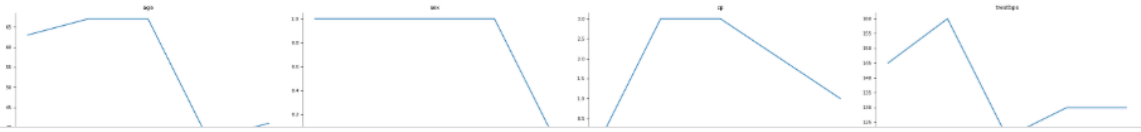
Distributions



2-d distributions



Values



✓ 7s completed at 2:51 PM

References/Credits

- Dataset Source: Kaggle Heart Disease Dataset *(Update with actual source if different)*
- Scikit-learn Documentation: https://scikit-learn.org/
- Seaborn Library: https://seaborn.pydata.org/