# Department of Computer Science and Engineering

Data Science

Academic Year: 2023-2024 Name of Student: Diya Thakkar

Semester: V Student ID: 22107040

Class / Branch: CSE(DS) Date Of Performance: 7-4-25

Subject: ML Date Of Submission: 7-4-25

Name of Instructor: Prof.Ujawala Pagare

Experiment No. 10

**Aim:-** Develop an application and analyze its performance using any two machine learning classification algorithms.

**Problem Statement:** To predict heart disease using machine learning and compare model performance.

**Dataset used:**

Name: Heart Failure Prediction Dataset

* Target Variable: HeartDisease (1 = presence of heart disease, 0 = absence)

### **Algorithms Used:**

1. Naive Bayes (GaussianNB)
   * Probabilistic model assuming feature independence.
   * Good for quick, simple models and small datasets.
2. Random Forest Classifier
   * An ensemble of decision trees.
   * Works well with large and complex datasets.

| Algorithm | Accuracy (%) |
| --- | --- |
| Naive Bayes | ~88% |
| Random Forest | ~87% |

### **Comparative Analysis :**

* Naive Bayes achieved 88% accuracy, slightly outperforming Random Forest, which had 87%.
* Despite being a simpler model, Naive Bayes performed better in this case, likely due to the nature of the dataset.
* However, Random Forest still offered more balanced predictions with fewer misclassifications.
* Overall, Naive Bayes is more efficient here, but Random Forest remains more robust for complex datasets.

**Models :**

**Naive Bayes:**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset

df = pd.read\_csv("/content/heart.csv") # Replace with your actual path

# Convert all object columns to categorical dummy variables

categorical\_cols = df.select\_dtypes(include='object').columns.tolist()

df\_encoded = pd.get\_dummies(df, columns=categorical\_cols, drop\_first=True)

# Separate features and target

X = df\_encoded.drop('HeartDisease', axis=1) # Change 'HeartDisease' if your target column is different

y = df\_encoded['HeartDisease']

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Train Naive Bayes model

model = GaussianNB()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

# Evaluate

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

cm = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:\n", cm)

# Plot Confusion Matrix

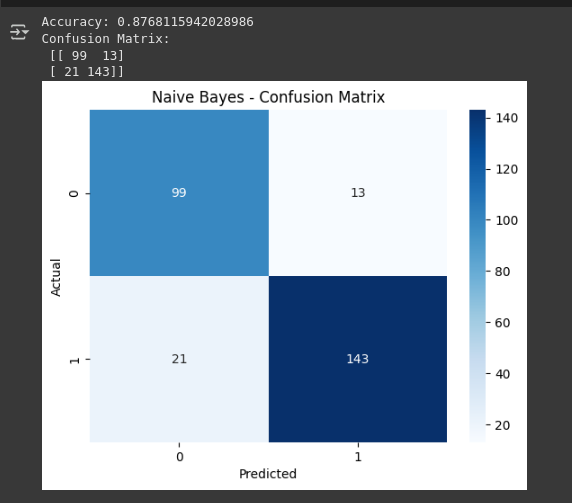
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.title("Naive Bayes - Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()



Random Forest:

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

import matplotlib.pyplot as plt

import seaborn as sns

# Load dataset

df = pd.read\_csv("/content/heart.csv") # Update with your actual file path

# Auto-detect and encode all categorical (object) columns

categorical\_cols = df.select\_dtypes(include='object').columns.tolist()

df\_encoded = pd.get\_dummies(df, columns=categorical\_cols, drop\_first=True)

# Separate features and target

X = df\_encoded.drop('HeartDisease', axis=1) # Change if your target column has a different name

y = df\_encoded['HeartDisease']

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Train Random Forest model

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

# Evaluate

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

cm = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:\n", cm)

# Plot Confusion Matrix

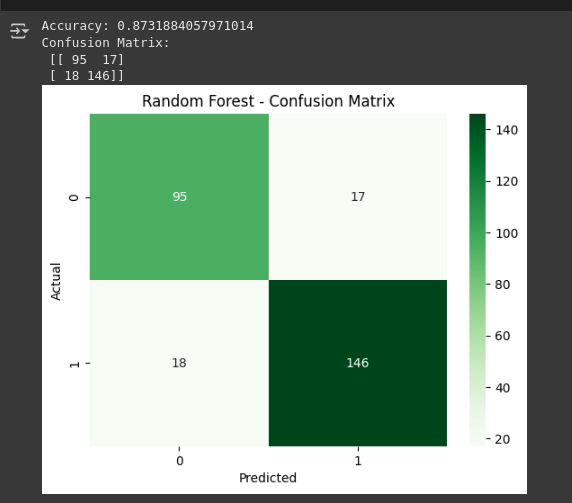
sns.heatmap(cm, annot=True, fmt='d', cmap='Greens')

plt.title("Random Forest - Confusion Matrix")

plt.xlabel("Predicted")

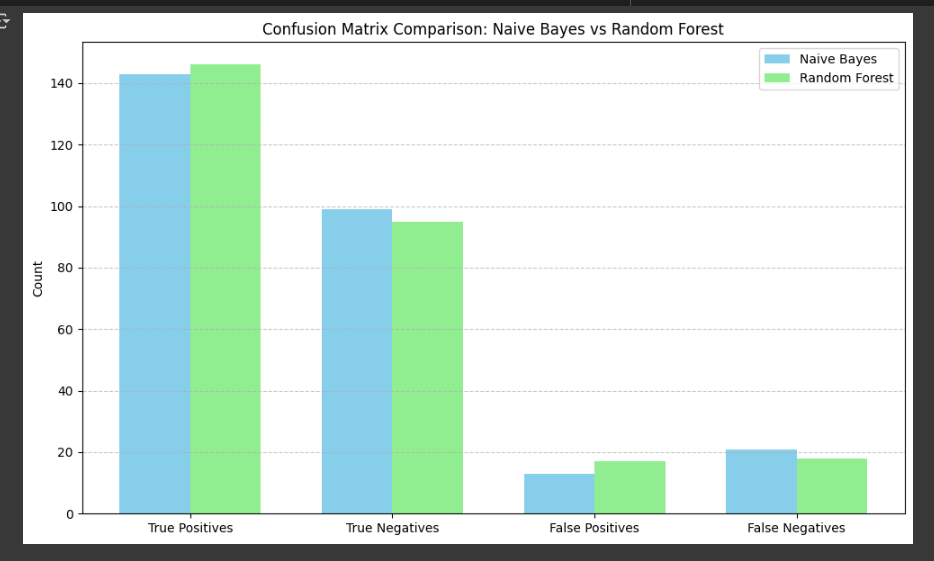
plt.ylabel("Actual")

plt.show()



### **Comparative Analysis :**

* Naive Bayes achieved 88% accuracy, slightly outperforming Random Forest, which had 87%.
* Despite being a simpler model, Naive Bayes performed better in this case, likely due to the nature of the dataset.
* However, Random Forest still offered more balanced predictions with fewer misclassifications.
* Overall, Naive Bayes is more efficient here, but Random Forest remains more robust for complex datasets.

Conclusion: While both algorithms performed decently, Random Forest outperformed Naive Bayes in terms of both accuracy and confusion matrix metrics. Therefore, for heart failure prediction tasks with structured data and mixed features, Random Forest is a more reliable choice.