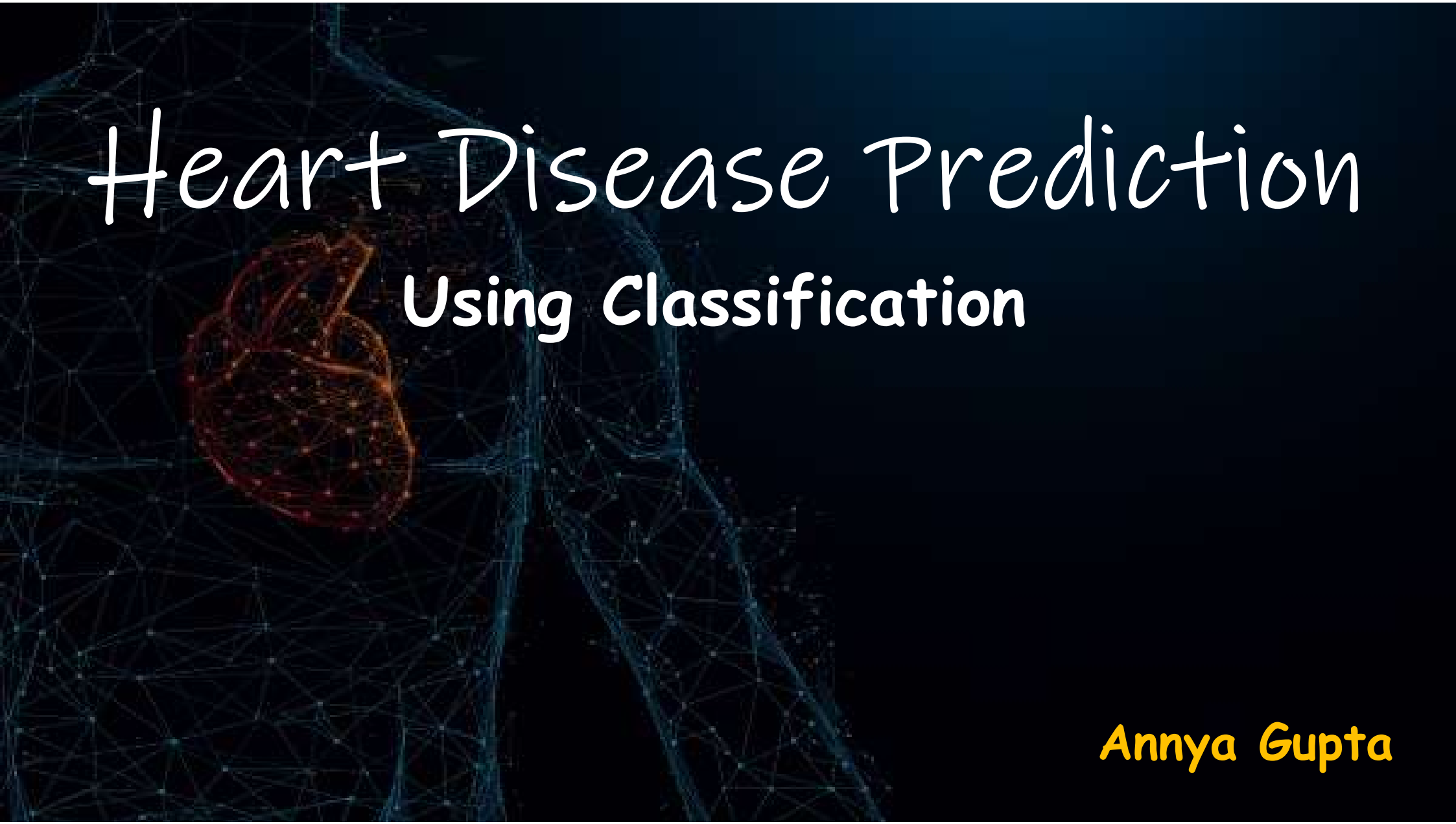

Heart Disease Prediction
Using Classification
Annya Gupta

Introduction

1. Heart disease remains one of the leading causes of death worldwide. Many patients fail to receive early warnings because identifying risk manually requires experience, time, and careful analysis of multiple health indicators.

2. Machine learning helps automate this process by learning patterns from historical medical data. This project builds a Heart Disease Classification Model that predicts whether a person is at risk of developing heart disease based on several medical attributes.

3. The dataset typically includes features such as:

(i) Age (ii) Sex (iii) Chest Pain Type

(iv) Resting Blood Pressure (v) Cholesterol Level (vi) Fasting Blood Sugar

(vii) ECG Results (viii) Max Heart Rate Achieved

This makes the model extremely useful for early screening and preventive healthcare.

AIM OF THE PROJECT

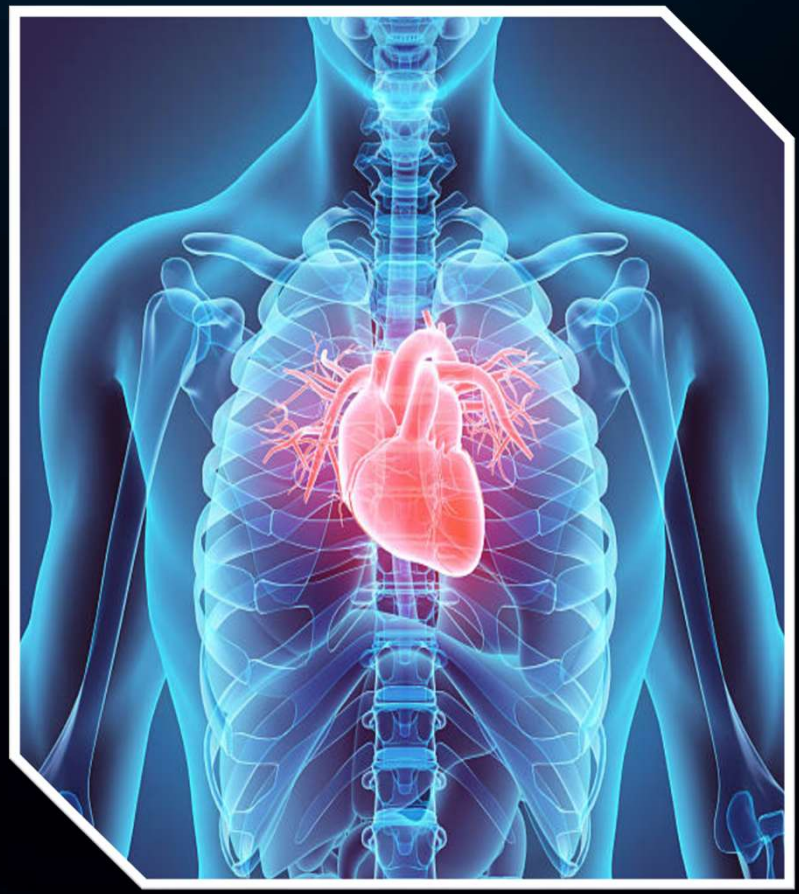
The primary goal of this project is to develop an accurate and reliable machine learning system that predicts the risk of heart disease using medical parameters.

1. **Identify high-risk patients early**
2. **Support doctors in clinical decision-making**
3. **Reduce chances of misdiagnosis**
4. **Assist hospitals in preventive care**
5. **Provide quick and consistent predictions**

The project trains multiple classification algorithms and compares their performance to determine the best-predicting model.

PROCESS

- ❑ Load Dataset
- ❑ Import Required Libraries
- ❑ Data Cleaning & Preprocessing
- ❑ Exploratory Data Analysis (EDA)
- ❑ Feature Scaling
- ❑ Train-Test Split
- ❑ Apply Multiple Classification Algorithms
- ❑ Model Evaluation using Accuracy, F1, AUC
- ❑ Select Best Model
- ❑ Make Predictions



ENCODING

1. Label Encoding :Used for ordered or non-text categorical fields such as:

(i) Sex

(ii) Chest Pain Type (cp)

(iii) Thalassemia (thal)

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df["Gender"] = le.fit_transform(df["Gender"])
df["Chest Pain Type"] = le.fit_transform(df["Chest Pain Type"])
# df["thal"] = le.fit_transform(df["thal"])
```

2. **One-Hot Encoding** : Used for non-ordinal categories where each category should become a separate column.

(i) Chest Pain Type

(ii) Thalassemia types

(iii) Slope of ST segment

```
target = 'Heart Disease'
X = pd.get_dummies(df.drop(columns=[target]), drop_first=True)
y = df[target]
```

CLASSIFICATION MODELS

1. LOGISTIC REGRESSION

It is Simple and interpretable model for linear relationships in medical data.

```
from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression()
classifier.fit(X_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

2. K-Nearest Neighbour

KNN: Classifies based on nearest patients; sensitive to noise.

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

3. Support Vector Machine

SVM: Finds the best boundary; works well for complex patterns.

```
from sklearn.svm import SVC
classifier = SVC(probability=True)
classifier.fit(X_train, y_train)
```

```
SVC(probability=True)
```

4. Decision Tree

Decision Tree: Uses simple rules; performs extremely well on this dataset.

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

5. Random Forest

Random Forest: Many trees combined; strong and stable accuracy.

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=200)
classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=200)
```

6. Gradient Boosting

Gradient Boosting: Sequentially improved trees; top-performing model.

```
from sklearn.ensemble import GradientBoostingClassifier
classifier = GradientBoostingClassifier()
classifier.fit(X_train, y_train)
```

```
GradientBoostingClassifier()
```

7. Naïve Bayes

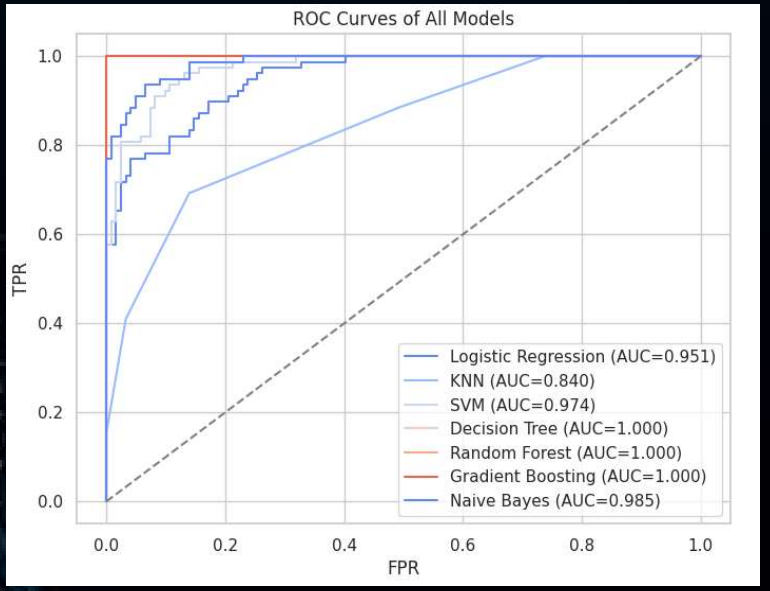
Naive Bayes: Fast probabilistic model; performs surprisingly well.

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

```
GaussianNB()
```

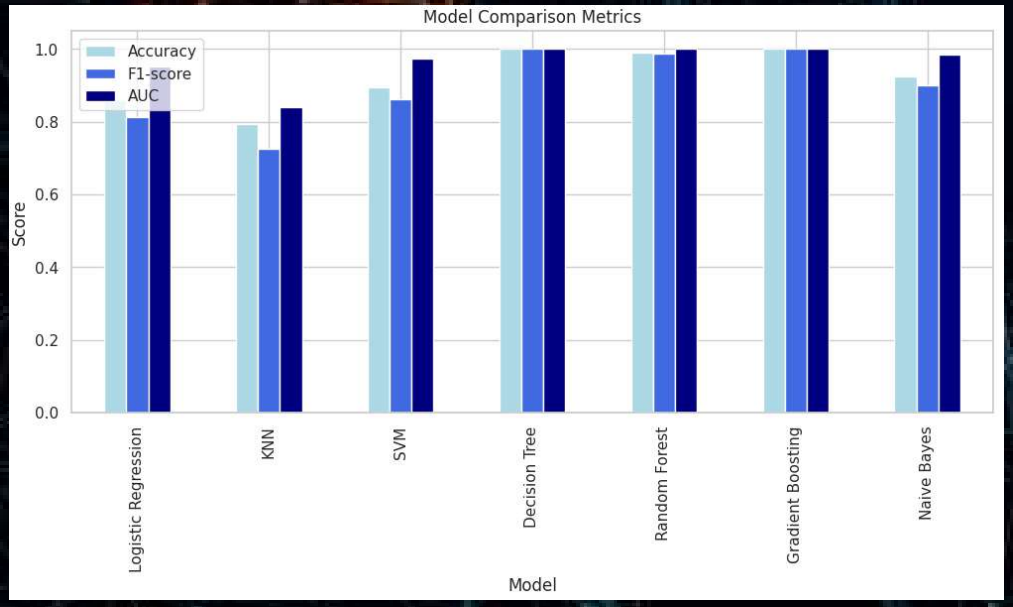
```
plt.figure(figsize=(8,6))
for name, (fpr, tpr, auc) in roc_store.items():
    plt.plot(fpr, tpr, label=f"{name} (AUC={auc:.3f})")

plt.plot([0,1],[0,1],"--", color="gray")
plt.title("ROC Curves of All Models")
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.legend()
plt.grid(True)
plt.show()
```



Model Comparison

Based on the results, Decision Tree and Gradient Boosting performed the best with perfect scores across all metrics. Random Forest also showed excellent performance with very high accuracy. Naive Bayes and SVM performed well, while Logistic Regression gave moderate results. KNN had the lowest accuracy, making it the weakest model in this comparison.




```
results_df = pd.DataFrame(
    results,
    columns=["Model", "Accuracy", "Precision", "Recall", "F1-score", "AUC"]
).set_index("Model")

print("\n=== MODEL COMPARISON ===\n")
print(results_df)
```

```
=== MODEL COMPARISON ===

                     Accuracy  Precision    Recall  F1-score       AUC
Model
Logistic Regression     0.860   0.847222  0.782051  0.813333  0.950820
KNN                     0.795   0.760563  0.692308  0.724832  0.839901
SVM                     0.895   0.880000  0.846154  0.862745  0.973623
Decision Tree           1.000   1.000000  1.000000  1.000000  1.000000
Random Forest           0.990   1.000000  0.974359  0.987013  1.000000
Gradient Boosting       1.000   1.000000  1.000000  1.000000  1.000000
Naive Bayes             0.925   0.943662  0.858974  0.899329  0.985183
```

Conclusion

From the results and F1-score comparison, the Decision Tree model achieved the highest score and was automatically selected as the best model. This means that the Decision Tree classifier provides the most accurate and balanced predictions for this dataset, making it the most suitable choice for heart disease prediction in this project.

Best Model

```
best_model_name = results_df["F1-score"].idxmax()
print("\n BEST MODEL NAME IS:", best_model_name)
best_model = models[best_model_name]
```

```
 BEST MODEL NAME IS: Decision Tree
```


Thank You !!