UNIVERSITI TEKNOLOGI MALAYSIA

SECP2753

DATA MINING

PROJECT 1

Title: Heart Failure Prediction Using Supervised Learning

PROF. MADYA DR. ROLIANA BINTI IBRAHIM

NO	NAME	MATRIC NO
1.	CHUA JIA LIN	A23CS0069
2.	EVELYN GOH YUAN QI	A23CS0222
3.	LAU YEE WEN	A23CS0099
4.	POH LOK YEE	A23CS0262

Table of Contents

1.0 Introduction

Cardiovascular disease continues to be a major global health concern, accounting for a significant portion of mortality rates worldwide. Early detection and prevention are crucial in managing the risks associated with heart disease. In this context, data mining techniques play a vital role in uncovering patterns and building predictive models using historical patient data.

This project focuses on applying supervised learning methods to classify the presence of heart disease based on a structured dataset. Two classification models are employed which are K-Nearest Neighbors (KNN) and Naive Bayes. KNN is a distance-based algorithm that classifies new data points based on the majority class of their nearest neighbors, while Naive Bayes applies probability theory with the assumption of feature independence, making it efficient for high-dimensional data.

By comparing the performance of these models on the heart disease dataset, the project aims to identify the most effective method for medical diagnosis. The findings are expected to support data-driven decision-making in healthcare, especially in the early prediction of heart-related conditions.

2.0 Business and Data Understanding

2.1 Problem Formulation

Heart disease is a leading cause of death globally, and timely diagnosis can significantly improve patient outcomes. However, relying solely on traditional diagnostic procedures can be time-consuming and expensive. This project aims to develop a predictive model that can determine whether a patient is likely to have heart disease based on various health indicators.

The primary objective is to use supervised machine learning techniques, specifically K-Nearest Neighbors (KNN) and Naive Bayes to classify individuals as either having heart disease or not. By analyzing patterns in patient data, these models can assist healthcare professionals in making early and informed decisions.

2.2 Dataset Overview

The dataset used in this project is a heart disease dataset (heart.csv) containing medical attributes of patients, which include both categorical and numerical variables. The key details of the dataset are as follows:

Number of Instances: 918

Number of Attributes: 12

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

Attribute Summary:

Attribute Name	Description	Type
Age	Age of the patient	Numerical
Sex	Gender (M=Male, F=Female)	Categorical
ChestPainType	Type of chest pain (ATA, NAP, ASY, TA)	Categorical
RestingBP	Resting blood pressure (mm Hg)	Numerical

Cholesterol	Serum cholesterol (mg/dl)	Numerical
FastingBS	Fasting blood sugar > 120 mg/dl (1 = True)	Categorical
RestingECG	Resting electrocardiogram results (Normal, etc.)	Categorical
MaxHR	Maximum heart rate achieved	Numerical
ExerciseAngina	Exercise-induced angina (Y = Yes, N = No)	Categorical
Oldpeak	ST depression induced by exercise	Numerical
ST_Slope	Slope of the peak exercise ST segment (Up, Flat, Down)	Categorical
HeartDisease	Target (1 = Yes, 0 = No)	Categorical

3.0 Data Preparation and Processing

Before moving to the model development stage, the Heart Failure Prediction dataset needs to be preprocessed to ensure that the dataset is clean, well-structured, and suitable for input into machine learning algorithms. Proper preprocessing helps the models generalize better and is able to produce more reliable predictions.

Data Cleaning and Preprocessing Steps

1. Upload the Heart Failure Prediction dataset

```
from google.colab import files
uploaded = files.upload()

df=pd.read_csv("heart.csv")
```

```
Choose Files heart.csv
• heart.csv(text/csv) - 35921 bytes, last modified: 24/05/2025 - 100% done
Saving heart.csv to heart (1).csv
```

2. Import required libraries

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

3. Checking any missing values in Heart Failure Prediction dataset

```
print("Missing values:\n", df.isnull().sum())
```

```
Missing values:
 Age               0
Sex                0
ChestPainType      0
RestingBP          0
Cholesterol        0
FastingBS          0
RestingECG         0
MaxHR              0
ExerciseAngina     0
Oldpeak            0
ST_Slope           0
HeartDisease       0
dtype: int64
```

4. Encode categorical columns in Heart Failure Prediction dataset

```
label_encoders = {}
for col in df.select_dtypes(include='object').columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

5. Split feature and target

```
X = df.drop('HeartDisease', axis=1)
y = df['HeartDisease']
```

6. Normalize numerical features for KNN

```
scaler = StandardScaler()
X_scaled = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
```

7. Split Heart Failure Prediction dataset into 70% training set and 30% testing set

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
```

```
print("Training data shape:", X_train.shape)
print("Testing data shape:", X_test.shape)
```

```
Training data shape: (642, 11)
Testing data shape: (276, 11)
```

4.0 Model Development (Supervised Learning)

During this phase, the K-Nearest Neighbours (KNN) and the Naive Bayers methods were selected for the model development process.

The KNN method aims to classify whether the patients are having heart disease (1) or not having heart disease (0) whereas the Naive Bayers method wants to estimate the probability of heart disease and classify the patients into Low Risk (probability < 0.5), Medium Risk (prob ability >= 0.5 && prob ability <= 0.8), and High Risk (prob ability > 0.8).

The KNN model was selected because it focusses on binary classification for diagnosis, while the Naive Bayers model is able to provide probabilistic outputs that are useful for risk-based decision-making.

4.1 K-Nearest Neighbors (KNN)

1. Import required libraries

```
from sklearn.metrics import precision_score, recall_score, f1_score, confusion_matrix, accuracy_score
from sklearn.neighbors import KNeighborsClassifier
```

2. Initialize the K-Nearest Neighbors classifier
   Set k = 5 (number of neighbors to consider)

```
knn = KNeighborsClassifier(n_neighbors=5)
```

3. Train the KNN model using the training data

```
knn.fit(X_train, y_train)
```

4. Predict class labels for the test set

```
y_pred_knn = knn.predict(X_test)
```

4.2 Naive Bayers

1. Import required libraries

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import numpy as np
```

2. Train Naive Bayers model using the training set

```
nb = GaussianNB()
nb.fit(X_train, y_train)
```

3. Predict class labels (0 or 1)

```
y_pred_nb = nb.predict(X_test)
```

4. Predict probabilities of getting heart disease

```
y_proba_nb = nb.predict_proba(X_test)[:, 1]  # probability of HeartDisease = 1
```

5.0 Model Performance and Evaluation

K-Nearest Neighbors (KNN)

Evaluation Metrics

```
knn_accuracy = accuracy_score(y_test, y_pred_knn)
knn_precision = precision_score(y_test, y_pred_knn)
knn_recall = recall_score(y_test, y_pred_knn)
knn_f1 = f1_score(y_test, y_pred_knn)

print("\n--- KNN Evaluation ---")
print(f"Accuracy: {knn_accuracy * 100:.2f}%")
print(f"Precision: {knn_precision * 100:.2f}%")
print(f"Recall: {knn_recall * 100:.2f}%")
print(f"F1-Score: {knn_f1 * 100:.2f}%")
```

```
--- KNN Evaluation ---
Accuracy: 86.96%
Precision: 91.03%
Recall: 86.59%
F1-Score: 88.75%
```

The KNN model achieved an accuracy of 86.96%, indicating that it correctly classified a high proportion of test instances. Its precision of 91.03% shows that the model is highly reliable when it predicts the presence of heart disease, which most of those predictions are correct. The recall of 86.59% means it successfully detected a large portion of actual heart disease cases. Finally, the F1-score of 88.75% reflects a strong balance between precision and recall, confirming that the model is both accurate and consistent in identifying patients at risk. These results highlight KNN's effectiveness in medical classification tasks, particularly in reducing false positives and ensuring that true cases of heart disease are identified early.

Confusion Matrix

```
cm = confusion_matrix(y_test, y_pred_knn)

print("\nConfusion Matrix (KNN):")
print(cm)
```

```
Confusion Matrix (KNN):
[[ 98  14]
 [ 22 142]]
```

The KNN model correctly predicted 98 patients without heart disease (true negatives) and 142 patients with heart disease (true positives). However, it also misclassified 14 healthy patients as having heart disease (false positives) and failed to detect heart disease in 22 actual cases (false negatives). These results indicate that the model performs well in identifying both positive and negative cases, with relatively low misclassification rates. The high number of true positives and

true negatives contributes to the model's strong precision and recall scores, making it a reliable tool for heart disease prediction.

Naive Bayers

Evaluation Metrics

```
accuracy = accuracy_score(y_test, y_pred_nb)
precision = precision_score(y_test, y_pred_nb)
recall = recall_score(y_test, y_pred_nb)
f1 = f1_score(y_test, y_pred_nb)

print("\n--- Naive Bayes Evaluation ---")
print(f"Accuracy: {accuracy * 100:.2f}%")
print(f"Precision: {precision * 100:.2f}%")
print(f"Recall: {recall * 100:.2f}%")
print(f"F1-Score: {f1 * 100:.2f}%")
```

```
--- Naive Bayes Evaluation ---
Accuracy: 87.32%
Precision: 91.08%
Recall: 87.20%
F1-Score: 89.10%
```

The results of Accuracy (87.32%), Precision (91.08%), Recall (87.20%), and F1-Score (89.10%) metrics indicate that the model is reliable and consistent in identifying patients with heart disease.

Confusion Matrix

```
cm = confusion_matrix(y_test, y_pred_nb)

print("\nConfusion Matrix (Naive Bayes):")
print(cm)
```

```
Confusion Matrix (Naive Bayes):
[[ 98  14]
 [ 21 143]]
```

The Confusion Matrix tells the values of TP (143), TN (98), FT (14), and FN (21). It shows that the model correctly identifies most heart disease cases with relatively few misclassifications.

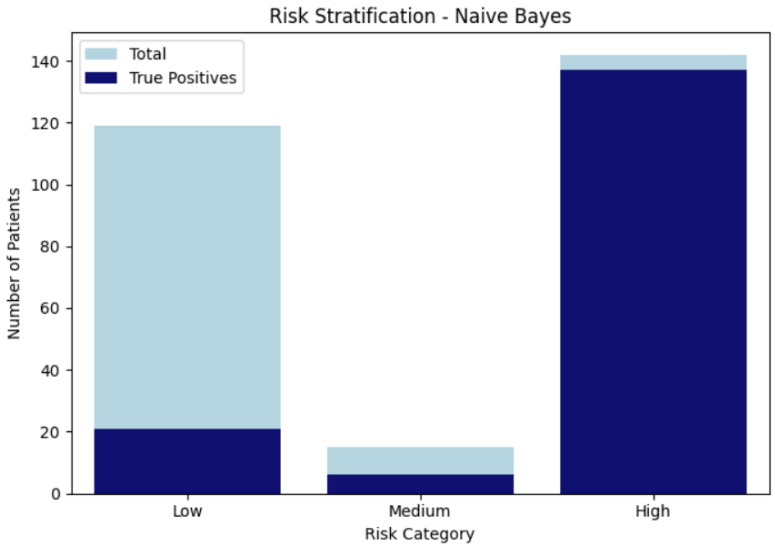
Risk Stratification

```python
# Risk Stratification based on probabilities
risk_labels = []
for prob in y_proba_nb:
    if prob < 0.5:
        risk_labels.append('Low')
    elif 0.5<= prob <= 0.8:
        risk_labels.append('Medium')
    else:
        risk_labels.append('High')

# Create dataframe to evaluate risk stratification
risk_df = pd.DataFrame({
    'Actual': y_test.values,
    'Predicted_Prob': y_proba_nb,
    'Risk_Category': risk_labels
})
```

```python
# Compute risk summary
risk_summary = []
for category in ['Low', 'Medium', 'High']:
    group = risk_df[risk_df['Risk_Category'] == category]
    total = len(group)
    true_positives = group['Actual'].sum()
    precision = (true_positives / total * 100) if total > 0 else 0
    risk_summary.append([category, total, true_positives, f"{precision:.1f}%"])

# Display risk summary as a DataFrame
risk_summary_df = pd.DataFrame(risk_summary, columns=["Risk Category", "Number of Patients", "True Positives", "Precision (%)"])

print("\n--- Risk Stratification Summary (Naive Bayes) ---")
print(risk_summary_df.to_string(index=False))
```

```
--- Risk Stratification Summary (Naive Bayes) ---
Risk Category  Number of Patients  True Positives Precision (%)
          Low                 119              21         17.6%
       Medium                  15               6         40.0%
         High                 142             137         96.5%
```

```python
# 2. Risk Stratification Bar Chart
risk_summary_df_plot = risk_summary_df.copy()
risk_summary_df_plot["True Positives"] = risk_summary_df_plot["True Positives"].astype(int)

plt.figure(figsize=(7, 5))
sns.barplot(data=risk_summary_df_plot, x="Risk Category", y="Number of Patients", color="lightblue", label="Total")
sns.barplot(data=risk_summary_df_plot, x="Risk Category", y="True Positives", color="navy", label="True Positives")
plt.title("Risk Stratification - Naive Bayes")
plt.ylabel("Number of Patients")
plt.legend()
plt.tight_layout()
plt.show()
```



The Risk Stratification results highlight the effectiveness in identifying patients based on their likelihood of having heart disease. The High-Risk group had very high precision (96.5%), showing the model's strength in correctly identifying patients at serious risk. On the other hand, the Low-Risk group showed low precision at 17.6%, suggesting a significant number of patients who were predicted to be at low risk but had heart disease. The Medium Risk group shows moderate performance with a precision of 40.0%, indicating some true positives but also including several false positives.

6.0 Conclusion

Both K-Nearest Neighbors (KNN) and Naive Bayers models provide useful insights and help to make effective predictions from the Heart Failure Prediction dataset.

For this Heart Failure Prediction dataset, the KNN model was good at classifying whether a patient has heart disease or not. It works well in making accurate decisions about the occurrence of heart disease based on the patients' health data.

On the other hand, the Naive Bayers model was helpful in dividing patients into different risk levels of heart disease by identifying which patients had a higher probability of getting heart disease. This could ease the healthcare professionals' burden as they only need to focus on high-risk heart disease patients.

In conclusion, the application of the KNN model helps in detecting heart disease accurately, while the usage of Naive Bayers model helps in prioritizing care based on patients' heart disease risk levels. The combination of both approaches can make the healthcare system more organized and efficient in treating patients.