Case Study: Heart Disease Prediction Using K-Nearest Neighbors (K-NN)

Aim

The aim of this experiment is to use the **K-Nearest Neighbors (K-NN)** algorithm to predict whether a person has heart disease by analyzing real patient health data.

Introduction

Heart disease is a serious health issue and is one of the main causes of death across the world. Doctors usually depend on medical tests and experience to diagnose heart disease. With the help of machine learning, patient data can be analyzed more effectively and used to support medical decisions.

In this case study, the K-Nearest Neighbors algorithm is used to classify patients as either having heart disease or not, based on different medical parameters.

Problem Statement

To build a machine learning model using the **K-Nearest Neighbors algorithm** that can predict the presence of heart disease using patient health attributes.

Dataset Used

The **Heart Disease Dataset** is used for this experiment.

- This dataset is taken from the **UCI Machine Learning Repository / Kaggle**
- It contains real-world hospital patient data
- The dataset has around **300 patient records**

Input Attributes

Age, gender, chest pain type, blood pressure, cholesterol level, ECG results, maximum heart rate, exercise-induced angina, and other related health parameters.

Target Attribute

- 0 → Patient does not have heart disease
- 1 → Patient has heart disease

Algorithm Used: K-Nearest Neighbors (K-NN)

K-NN is a supervised learning algorithm that works by comparing a new data point with existing data points.

The algorithm:

- Measures the distance between data points
- Finds the **K closest neighbors**
- Predicts the class based on majority voting

Euclidean distance is used to measure similarity between data points.

Steps Performed

1. The dataset was loaded into Google Colab
2. The data was checked to understand its structure
3. The dataset was split into training and testing sets
4. Feature scaling was applied because K-NN is distance-based
5. The K-NN model was trained using training data
6. Predictions were made on test data
7. Model performance was evaluated
8. The value of K was optimized to improve accuracy

Implementation

The experiment was performed using **Python in Google Colab**.
Pandas and NumPy were used for data handling, while Scikit-learn was used to build and evaluate the model.

Initially, the value of K was set to 5. Later, **hyperparameter tuning** was done using GridSearch to find the best value of K, which improved the model's performance.

Python program :-

```
import pandas as pd

import numpy as np


from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report


df = pd.read_csv('heart.csv')


print(df.head())

print(df.info())

print(df.describe())
```

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

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

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

print(accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

param_grid = {
    'n_neighbors': np.arange(1, 21)
}

grid = GridSearchCV(
    KNeighborsClassifier(),
    param_grid,
    cv=5,
    scoring='accuracy'
)
```

```
grid.fit(X_train, y_train)


print(grid.best_params_)
print(grid.best_score_)


best_knn = grid.best_estimator_
y_final_pred = best_knn.predict(X_test)


print(accuracy_score(y_test, y_final_pred))
print(confusion_matrix(y_test, y_final_pred))
print(classification_report(y_test, y_final_pred))
```

Model Evaluation

The model was evaluated using:

- Accuracy score
- Confusion matrix
- Classification report

These metrics helped in understanding how well the model predicted heart disease.

Result

```
df = pd.read_csv('heart.csv')
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

← **TRAING KNN Model**

```
df['target'].value_counts()
```

```
            count
target
1             526
0             499
dtype: int64
```

← **Target value**

```
Accuracy: 0.8341463414634146
[[79 23]
 [11 92]]
              precision    recall  f1-score   support

           0       0.88      0.77      0.82       102
           1       0.80      0.89      0.84       103

    accuracy                           0.83       205
   macro avg       0.84      0.83      0.83       205
weighted avg       0.84      0.83      0.83       205
```

← **Model evaluation**

- **Hyperparameter Tuning**

```
Best K: {'n_neighbors': np.int64(1)}
Best Accuracy: 0.978048780487805
```

```
Final accuracy
```

```
Final Accuracy: 0.9853658536585366
```

The final K-NN model was able to predict whether a patient has heart disease with `0.9853658536585366` **accuracy**.
After tuning the value of K, the prediction results improved further.

Conclusion

This case study shows that the **K-Nearest Neighbors algorithm** can be effectively used for predicting heart disease using medical data. Proper preprocessing and selection of the best K value play an important role in achieving accurate results. The model can be useful as a support system for doctors in early diagnosis.

Applications

- Medical diagnosis
- Healthcare analytics
- Clinical decision support systems