# **Heart Disease Risk Prediction Using K-Nearest Neighbors**

## **Introduction**

The goal of this project is to predict the risk of heart disease within the next 10 years using the Framingham Heart Study dataset. This dataset contains several health-related factors such as age, smoking status, cholesterol levels, blood pressure, body mass index (BMI), and more, which are potential indicators of heart disease. The target variable, TenYearCHD, is binary, indicating whether or not the individual developed heart disease within the next decade (1 = heart disease, 0 = no heart disease).

For this classification problem, we employed the K-Nearest Neighbors (K-NN) algorithm. This algorithm is widely used for classification tasks and works by finding the most similar data points (neighbors) to a given point and predicting the class based on the majority class of those neighbors. We trained the K-NN model using different values of K (3, 5, and 7) and evaluated their performance.

## **Methodology**

**Data Preprocessing:**

1. **Missing Value Handling**: The dataset contained missing values, which were handled by filling missing categorical data (such as education) with the mode (most frequent value) and numerical data (such as cigsPerDay, totChol, etc.) with the median. This approach was chosen to preserve the data distribution while ensuring all records could be used for training.

# Handle missing values

df['education'] = df['education'].fillna(df['education'].mode()[0])

df['cigsPerDay'] = df['cigsPerDay'].fillna(df['cigsPerDay'].median())

df['BPMeds'] = df['BPMeds'].fillna(df['BPMeds'].median())

df['totChol'] = df['totChol'].fillna(df['totChol'].median())

df['BMI'] = df['BMI'].fillna(df['BMI'].median())

df['heartRate'] = df['heartRate'].fillna(df['heartRate'].median())

df['glucose'] = df['glucose'].fillna(df['glucose'].median())

1. **Feature Normalization**: Since K-NN is sensitive to the scale of features, all numerical features were normalized using StandardScaler to ensure they are on the same scale. This improves the model's performance by making all features contribute equally to distance calculations.

# Normalize the numerical features

X = df.drop(columns=['TenYearCHD'])

y = df['TenYearCHD']

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

1. **Train-Test Split**: The dataset was split into training and testing sets using an 80/20 split, with 80% of the data used for training the model and 20% used for testing. This allowed us to validate the model on unseen data.

# Split the dataset into training and testing sets (80/20 split)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

**Training Phase:**

For the training phase, we implemented the K-NN algorithm with three different values of K (3, 5, and 7). The three main steps in the K-NN algorithm are:

1. **Calculate the Distance**: For each test point, the algorithm calculates the distance to all training points. In scikit-learn's KNeighborsClassifier, Euclidean distance is used by default (the "minkowski" metric with p=2).
2. **Find the Nearest Neighbors**: The algorithm then selects the K training points with the smallest distances to the test point. These are the "nearest neighbors" of the test point.
3. **Making Predictions**: Finally, the algorithm predicts the class of the test point based on the majority class among its K nearest neighbors.

In our implementation, we used scikit-learn's KNeighborsClassifier which handles these steps internally:

# Try different values of K (e.g., K=3, 5, 7)

k\_values = [3, 5, 7]

results = {}

# Train and evaluate the models

for k in k\_values:

    knn = KNeighborsClassifier(n\_neighbors=k)

    knn.fit(X\_train, y\_train)

    y\_pred = knn.predict(X\_test)

    accuracy = accuracy\_score(y\_test, y\_pred)

    results[k] = {

        'accuracy': accuracy,

        'confusion\_matrix': confusion\_matrix(y\_test, y\_pred),

        'classification\_report': classification\_report(y\_test, y\_pred)

    }

**Testing Phase:**

In the testing phase, we applied our trained models to the test dataset to predict heart disease risk. We compared the accuracy across all three models to evaluate their performance on unseen data.

     # Print results for each K model

    print(f"Results for K={k}:")

    print(f"Accuracy: {results[k]['accuracy']}")

    print("Confusion Matrix:")

    print(results[k]['confusion\_matrix'])

    print("Classification Report:")

    print(results[k]['classification\_report'])

    print("="\*50)

The results from the testing phase showed varying performance across the different K values:

* K=3:

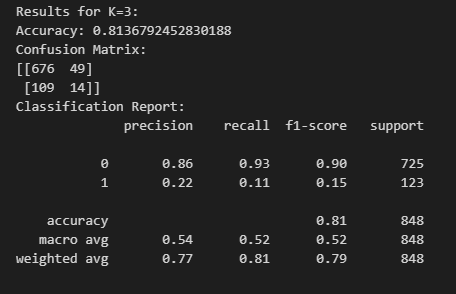


Figure : Results for K=3

* K=5:

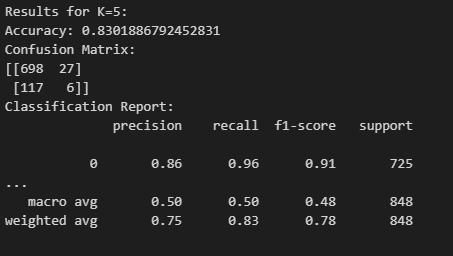


Figure : Results for K=5

* K=7:

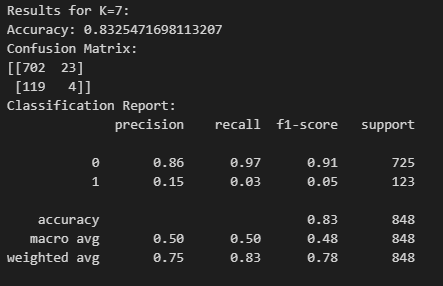


Figure : Results K=7

These results indicate that as K increases from 3 to 7, the model's overall accuracy on the test data improves slightly, suggesting that considering more neighbors helps stabilize predictions for this dataset.

**Evaluation Phase:**

For the evaluation phase, we analyzed the models' performances using confusion matrices, recall scores, and precision scores for all three K values.

For K=3:

* Accuracy: 81.37%
* Confusion Matrix:
* [[676 49] [109 14]]
* Classification Report:
  + Precision (Class 0): 0.86
  + Recall (Class 0): 0.93
  + F1-score (Class 0): 0.90
  + Precision (Class 1): 0.22
  + Recall (Class 1): 0.11
  + F1-score (Class 1): 0.15

For K=5:

* Accuracy: 83.02%
* Confusion Matrix:
* [[698 27] [117 6]]
* Classification Report:
  + Precision (Class 0): 0.86
  + Recall (Class 0): 0.96
  + F1-score (Class 0): 0.91
  + Precision (Class 1): 0.18
  + Recall (Class 1): 0.05
  + F1-score (Class 1): 0.08

For K=7:

* Accuracy: 83.25%
* Confusion Matrix:
* [[702 23] [119 4]]
* Classification Report:
  + Precision (Class 0): 0.85
  + Recall (Class 0): 0.97
  + F1-score (Class 0): 0.91
  + Precision (Class 1): 0.15
  + Recall (Class 1): 0.03
  + F1-score (Class 1): 0.05

Understanding these metrics:

* **Accuracy**: Represents the proportion of correct predictions out of all predictions
* **Confusion Matrix**: Shows true positives, false positives, true negatives, and false negatives
* **Recall**: Measures the ability to find all positive instances (TP/(TP+FN))
* **Precision**: Measures the accuracy of positive predictions (TP/(TP+FP))

The misclassification error analysis shows that all models struggle with correctly identifying positive cases (class 1 - patients with heart disease), with particularly low recall values. This indicates that many patients with heart disease risk are being classified as healthy.

## **Results and Discussion**

**Correlation Analysis:**

We used a correlation heatmap to visualize relationships between variables in the dataset.

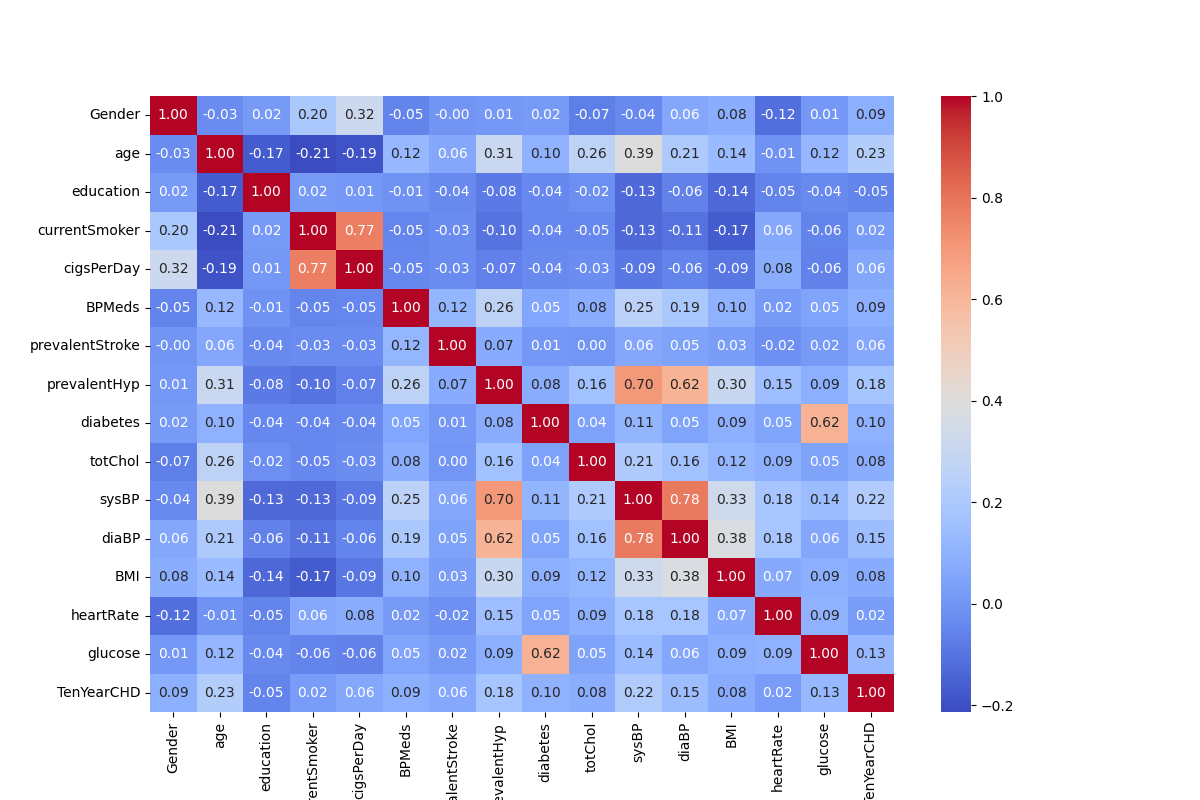


Figure 4: Correlation Heatmap showing relationships between features in the Framingham Heart Study dataset.

The correlation heatmap illustrates important relationships between numerical variables in the dataset. Some key observations include:

* Cigarettes per day (cigsPerDay) and current smoking status (currentSmoker) are highly correlated (0.77), as expected.
* Systolic blood pressure (sysBP) and diastolic blood pressure (diaBP) have a strong correlation (0.78), which is consistent with medical knowledge.
* Prevalent hypertension (prevalentHyp) and systolic blood pressure (sysBP) show a strong correlation (0.70), indicating that individuals with hypertension typically have higher systolic blood pressure.
* Diabetes and glucose levels are strongly correlated (0.62), which aligns with the medical understanding that diabetes is characterized by elevated blood glucose levels.
* Body mass index (BMI) shows moderate correlations with systolic blood pressure (0.33) and diastolic blood pressure (0.38), suggesting that higher BMI is associated with increased blood pressure.
* Ten-year risk of heart disease (TenYearCHD) shows moderate correlations with several health indicators, such as age (0.23) and systolic blood pressure (0.22), suggesting that these factors influence heart disease risk.

**Best Model Selection:**

We automatically selected the best model based on the highest accuracy. After evaluating all three models, we found that:

* The model with K=7 achieved the highest accuracy (83.25%), making it our best model according to this metric.
* However, when looking specifically at detecting heart disease cases (class 1), all models performed poorly, with the K=3 model having the highest recall (0.11) for class 1.
* The lower recall scores for class 1 indicate that our models are missing many heart disease cases, which is particularly concerning in a medical context where false negatives can have serious consequences.
* The trade-off between overall accuracy and class 1 recall suggests that while K=7 is technically the "best" model by accuracy, the K=3 model might be preferable if detecting heart disease cases is the priority.

## **Conclusion**

In this project, we employed the K-Nearest Neighbors (K-NN) algorithm to classify the risk of heart disease using the Framingham Heart Study dataset. Our results showed that the model with K=7 achieved the highest overall accuracy of 83.25%. However, all models performed poorly at identifying patients with heart disease (class 1), as evidenced by the low recall scores.

The following are recommendations to enhance the performance of the model:

1. **Address Class Imbalance**: The dataset shows significant imbalance between healthy patients and those with heart disease. Techniques to address this problem include oversampling the minority class (using methods like SMOTE), undersampling the majority class, or using class weights in the model.
2. **Try Other Algorithms**: Other machine learning models such as Random Forest, Support Vector Machines, or Logistic Regression might perform better, especially in terms of recall for the minority class. Ensemble methods could also be explored to improve performance.
3. **Feature Engineering**: Developing new features or selecting the best features from the available ones could enhance the model's accuracy of diagnosing heart disease. This could include creating interaction terms between correlated features or transforming existing features.
4. **Hyperparameter Tuning**: We could perform a more exhaustive search for the optimal value of K using cross-validation techniques, or explore different distance metrics beyond the default Euclidean distance used in our implementation.

In conclusion, while the K-NN model provides a reasonable baseline for predicting heart disease risk with an accuracy of over 83%, there is significant room for improvement, particularly in correctly identifying patients at risk of heart disease. Future work should focus on addressing the class imbalance problem and exploring more sophisticated modeling approaches.