Author: Vishal Singh Rathore

Predicting Heart Disease Using Machine Learning

Dataset

The dataset I have used is the Cleveland heart Disease dataset from UCI Machine Learning Repository. It has a total of 303 observations and 14 attributes, which has both quantitative (Age, Resting blood pressure, Serum cholesterol etc.) and qualitative (sex, Chest pain type, Resting ECG results etc.) features relevant to cardiovascular health. The target variable indicates the presence of heart disease and was converted to a binary format where 0 being no heart disease and 1 being heart disease present.

I believe the dataset is well suited for a supervised classification task and has been widely used in machine learning research focused on clinical prediction. It is important because it uses basic and easily accessible medical features which makes it more applicable to real world checkups.

Problem Formulation and Significance

Heart disease is one of the most common causes of death all around the world. This project will help me learn if machine learning can help predicting who is at a risk of having a heart disease using health data that doctors collected during regular checkups. I have used logistic regression, decision trees, and K-nearest neighbors (k-NN) to predict the presence of heart disease.

Detecting early symptoms of heart disease can save lives and reduce healthcare spending. If a model like this performs well, it could allow doctors to catch problems before they become serious and allow patients to receive care sooner. Since it uses information that is already collected in routine healthcare visits, it could be included in current procedures to improve existing treatments.

Exploratory Data Analysis

Data Exploration

To begin my analysis, I looked at descriptive statistics for all features in the dataset. This included the mean, standard deviation, minimum, and maximum values. These basic statistics helped me understand how the values were spread and what typical measurements looked like for each feature. For instance, the age of patients in the dataset ranges from 29 to 77, and cholesterol values showed a wide spread across different individuals. Looking at these numbers helped me recognize which features varied the most and which ones might be useful in predicting heart disease.

While reviewing the features, I noticed that certain ones showed noticeable differences between people who had heart disease and those who didn't. For example, chest pain type and ST depression stood out early on. People with heart disease were more likely to have certain types of chest pain and higher ST depression values. These early findings helped me choose which features to focus on when I built the models later in the project.

I also explored how the target variable — whether or not a person had heart disease — related to other features. This included comparing averages between the two groups and checking how the target variable changed across categories. In addition, I looked at the overall distributions of both categorical and numerical variables. This gave me a clearer view of the dataset's structure and balance. For example, I could see how evenly patients were spread across different chest pain types or how blood pressure levels were distributed. All of this helped me better prepare the data and understand the patterns I might expect the models to learn.

Handling of Missing/Imbalanced Data

While preparing the Cleveland Heart Disease dataset, I noticed missing values in two specific columns: thal and ca. These missing values were marked with a "?" symbol instead of standard null entries. To fix this, I first replaced the "?" symbols with actual missing value indicators using pandas. After that, I removed all rows that contained missing values. This was a straightforward and safe approach because only a small number of rows had missing data. Removing them did not significantly change the size of the dataset or its distribution, and it helped ensure that the models would not be confused by incomplete entries.

After cleaning the missing values, I worked on normalizing the numeric features. Some columns, like cholesterol, resting blood pressure , and ST depression, had very different ranges and units. I used **standard scaling**, which transforms each feature so that it has a mean of 0 and a standard deviation of 1. This was an important step because many machine learning models, especially logistic regression and K-nearest neighbors, are sensitive to differences in scale. Scaling made sure that each feature contributed equally during model training and that no feature was unfairly weighted just because of its numerical size.

I also checked whether the dataset was imbalanced in terms of the target variable. There were slightly more patients who had heart disease than those who didn't. However, the difference was not very large, so I decided not to apply techniques like resampling, class weighting, or synthetic data generation. Instead, I evaluated the model performance carefully using multiple metrics — not just accuracy, but also **recall** and **AUC** — to ensure that the models were performing well for both positive and negative cases. This helped give a more complete picture of how well each model handled the slight imbalance without needing extra steps to adjust the data.

Data Visualization

To better understand the data, I created several charts that helped reveal patterns and trends. First, I made a correlation heatmap to see how strongly each feature was related to the target variable, which is the presence of heart disease. This chart showed that features like chest pain type, maximum heart rate, and ST depression had strong relationships with the outcome. These features stood out and gave me an early idea of which ones might be the most useful during modeling.

Next, I created histograms for key features like age and cholesterol. These helped me understand the overall distribution of the data. For example, most patients in the dataset were between 50 and 60 years old, and cholesterol values varied a lot across individuals. Seeing these distributions gave me a better idea of how the data was spread and whether it might affect model training.

I also made a boxplot comparing ST depression levels between patients with and without heart disease. The chart clearly showed that people with heart disease often had higher ST depression, which supported what I found in the correlation heatmap. This type of visual comparison made it easier to see which variables could be strong indicators of heart problems.

To see how well each model predicted the results, I created confusion matrix plots. A confusion matrix shows how many times the model was right or wrong. It tells us how many people with heart disease were correctly identified, and how many were missed. It also shows if the model mistakenly predicted someone had heart disease when they didn't. These plots helped me understand not just how often the model was right, but also what kinds of mistakes it made. By looking at all three models this way, I could compare which ones were more accurate and which ones made more errors.

Overall, these visualizations played an important role in helping me understand the dataset. They made it easier to choose important features for modeling and supported the results I found later during evaluation. Using visual tools made the data more accessible and guided better decisions during the analysis process.

Model Selection, Application, and Evaluation

For this project, I chose to use three different models: logistic regression, decision tree, and K-nearest neighbors (KNN). These models were selected because they represent different types of machine learning methods that we covered in class — linear models, tree-based models, and distance-based models. More importantly, each model has strengths that match the characteristics of the dataset and the nature of the problem I'm trying to solve, which is predicting whether a person has heart disease based on medical data.

Logistic regression is a widely used linear model for binary classification problems. It works by finding the relationship between the input features and the target variable using a logistic function. I chose this model because it is simple, easy to train, and provides clear, interpretable results. Each feature is assigned a coefficient, which shows how it affects the final prediction. This is useful in medical problems because it helps identify which health indicators have the strongest influence on heart disease. Logistic regression also performs well when the features have a linear relationship with the outcome, and since some features in the dataset like ST depression and chest pain type appeared to show this, logistic regression was a good starting point.

The decision tree model was chosen because it can capture more complex, non-linear relationships between the features and the target variable. Unlike logistic regression, which assumes a straight-line relationship, decision trees split the data into smaller groups based on decision rules. This makes them more flexible and capable of handling a mix of different types of features. Another advantage of decision trees is that they are very interpretable. Each path from the root to a leaf in the tree represents a decision rule, which can be useful for explaining why a certain prediction was made. This makes decision trees especially helpful in healthcare, where transparency is important.

Lastly, I included K-nearest neighbors (KNN) because it is a non-parametric, distance-based model. KNN works by comparing a new data point to its closest neighbors in the training set and assigning the most common class among those neighbors. KNN is useful for problems where similar examples are expected to have similar outcomes, which fits with the idea that patients with similar health measurements may have similar risk levels. This model doesn't make any assumptions about the underlying distribution of the data, which can be helpful when working with real-world datasets that might not follow a clear pattern. Since the Cleveland dataset is not too large, KNN can run efficiently without causing long delays.

Overall, I selected these three models because they are different in how they make predictions, which gave me the chance to compare a variety of approaches. Logistic regression gave me a strong baseline with interpretability, decision trees allowed me to explore more complex rules in the data, and KNN helped me understand how data clustering affects classification. All three models were learned and practiced in class, and applying them to this real-world dataset helped reinforce those concepts while allowing me to explore the strengths and weaknesses of each one.

Model Application

After selecting the models, I followed a series of steps to prepare the data and apply each model fairly and consistently. First, I cleaned the dataset by handling the missing values. In the Cleveland Heart Disease dataset, the ca and thal columns had missing values represented by question marks. I converted these to actual missing values using pandas, and then removed the rows containing them. This approach was chosen because only a small number of rows were missing, so dropping them did not harm the dataset significantly. Once the data was clean, I split it into training and testing sets. I used an 80/20 split, which means 80% of the data was used to train the models, and 20% was held out to test how well the models could perform on new, unseen data.

Next, I scaled all the numeric features using standard scaling. This step was important because some of the models, especially K-nearest neighbors and logistic regression, can be affected by the scale of the features. For example, cholesterol values are much larger than binary features like sex or fasting blood sugar. Without scaling, the model could give too much weight to features just because they have larger numbers. I used the StandardScaler from scikit-learn, which standardizes the features to have a mean of zero and a standard deviation of one.

After preprocessing, I trained each of the three models using the same training data. For logistic regression, I used the default settings provided in scikit-learn, which uses L2 regularization. This worked well with the data and helped avoid overfitting. For the decision tree, I also used default settings but made sure to set a random seed for reproducibility. This model builds a tree by selecting the best features to split the data at each level, based on how well the split separates the classes. For K-nearest neighbors, I set the value of `k` to 5. This means the model looks at the five closest points in the training set and predicts the class that appears most often among those neighbors. I picked k=5 because it's a common default value and balances smoothness with responsiveness to local patterns.

Once all the models were trained, I used them to make predictions on the test set. I collected results like accuracy, confusion matrices, and AUC scores to evaluate each model. These steps were done carefully so I could make a fair comparison between the models and understand how each one worked with the dataset. Throughout the process, I kept the data pipeline consistent — all models were trained and tested on the same scaled and cleaned dataset — so that the results would reflect differences in the models themselves rather than differences in the data preparation.

Model Evaluation

To understand how well each model performed, I used several evaluation metrics. These included accuracy, confusion matrix, and AUC (Area Under the Receiver Operating Characteristic Curve). Accuracy shows the overall percentage of correct predictions. The confusion matrix shows how many times the model was right or wrong when predicting whether a patient had heart disease. It also breaks down the results into four categories: true positives, true negatives, false positives, and false negatives. AUC measures how well the model separates the two classes — in this case, people with heart disease and people without it. A higher AUC score means the model is better at telling the difference between the two groups.

The logistic regression model gave the best overall results. It achieved an accuracy of about 87%, and the AUC score was 0.94. This means it correctly predicted most test cases and was also good at ranking patients by their risk of having heart disease. The confusion matrix showed that it made very few false positives and false negatives, which is important in medical settings where both kinds of mistakes can lead to problems. False positives could mean unnecessary tests or treatment, while false negatives could cause a delay in care.

The K-nearest neighbors (KNN) model also performed well. It had an accuracy of 83% and an AUC score of 0.93. While not as high as logistic regression, the results were still strong. The confusion matrix for KNN showed slightly more false negatives than logistic regression, but the overall balance between classes was still good. This model worked well especially after scaling the features, which is important for KNN since it relies on distance calculations.

The decision tree model had the lowest performance among the three. It reached an accuracy of 80% and had an AUC score of 0.81. It made more false positives than the other models, which reduced its precision. However, it still performed reasonably well, and one benefit of decision trees is their interpretability. I could look at the structure of the tree and see which questions the model was asking to make decisions, such as whether chest pain type or ST depression was above or below a certain value.

In all models, certain features stood out as the most useful for prediction. These included chest pain type, ST depression, and maximum heart rate. These features were consistently important across the different models and had strong effects on the model outputs. This supported what I saw earlier during data exploration and helped confirm that the models were learning patterns that made sense medically.

Overall, logistic regression gave the best balance of performance and simplicity, but all three models helped show different aspects of the data. Comparing them side by side allowed me to understand how each model works and where they succeed or struggle. This evaluation step was important to choose the best model and to build confidence in its predictions.

Results

The results from the three models showed clear differences in performance. Logistic regression had the best outcome with an accuracy of 87% and an AUC score of 0.94. This means it made the most correct predictions and was very good at separating people with and without heart disease. The confusion matrix showed that it had very few false positives and false negatives. K-nearest neighbors also performed well, with an accuracy of 83% and an AUC of 0.93. Decision tree had slightly lower performance, with an accuracy of 80% and an AUC of 0.81. It made more mistakes than the other two models, especially false positives.

I also looked at which features were most useful across all models. Chest pain type, ST depression , and maximum heart rate were the most important predictors. These features made a big difference in whether or not a patient was classified as having heart disease. These results match what I noticed during exploratory data analysis, which supports the idea that the models were learning meaningful patterns. All three models were trained on the same cleaned and scaled data, and their results were compared using the same test set, making the evaluation fair and consistent.

Conclusions

From this project, I learned that even simple machine learning models can help predict heart disease using basic health data. Logistic regression was the strongest model overall because it was accurate, fast, and easy to interpret. KNN also worked well, especially after scaling the features. While the decision tree didn't perform quite as well, it was helpful for understanding how different features split the data and lead to decisions.

One limitation of the project was the size of the dataset. With only 303 rows, the model may not capture all the patterns that might appear in a larger or more diverse population. Also, I dropped missing values instead of trying to fill them in, which may have removed useful data. Even though the class imbalance wasn't large, it could still affect results in real-world settings. If I were to continue this project, I would try different balancing techniques and test more advanced models like random forests or gradient boosting.

Workforce/Graduate School Preparation

This project helped me improve several skills that are important for both graduate school and the workplace. I got better at working with real-world data — including cleaning it, exploring patterns, and preparing it for machine learning. I practiced building and comparing different types of models, which helped me understand how to choose the right approach depending on the problem. I also improved my ability to explain results in a simple way, both through charts and written analysis.

From a technical point of view, I learned more about how to use Python libraries like pandas, matplotlib, and scikit-learn. These are tools that are widely used in both research and industry. I also became more confident in using evaluation metrics like AUC, confusion matrix, and recall to judge model performance, especially in health-related data where mistakes can have real consequences. This project gave me a full experience of the machine learning workflow — from asking a question to drawing a conclusion — and that will definitely help in my future projects and career.