Knowledge Discovery and Data Mining
CIS 635 03
Predicting Heart Disease Using Machine Learning

by
Yaswitha Kalapala
Sujish Kumar Indrajith
Rebecca Dara
Lakshmi Wajja

Contents

1. Introduction

Heart disease remains one of the leading causes of death worldwide, posing a critical challenge to global public health. Early detection and diagnosis are essential for effective treatment and improving patient outcomes. In recent years, the integration of machine learning (ML) techniques into healthcare has demonstrated promising potential in identifying risk factors and predicting disease occurrence.

Motivation

This project aims to build a machine learning-based predictive model to assist healthcare professionals in identifying patients at risk of heart disease. By leveraging the Cleveland Heart Disease dataset, we strive to create a data-driven diagnostic aid that could ultimately contribute to saving lives through timely interventions.

Overview of Approach

Our approach includes thorough data preprocessing, outlier handling, normalization, encoding of categorical variables, and strategic feature selection. We trained and evaluated multiple classification models, including logistic regression, KNN, SVC, decision trees, random forest, and Naive Bayes. The Support Vector Classifier (SVC) demonstrated the highest accuracy of 86.6%, indicating strong potential for clinical application.

2. Related Work

Significant work has been conducted in applying machine learning to heart disease prediction. Previous studies have used various algorithms such as logistic regression, support vector machines (SVMs), and ensemble methods like random forests to detect cardiovascular conditions. Notable references include:

- Framingham Heart Study (2023): A landmark longitudinal study identifying key cardiovascular risk factors, widely used for risk prediction models.
- Detrano et al. (1989): Developed a probabilistic algorithm for diagnosing coronary artery disease, which laid the groundwork for many modern ML approaches.
- CNN-based deep learning methods have also shown success in processing ECG data for cardiac condition classification.

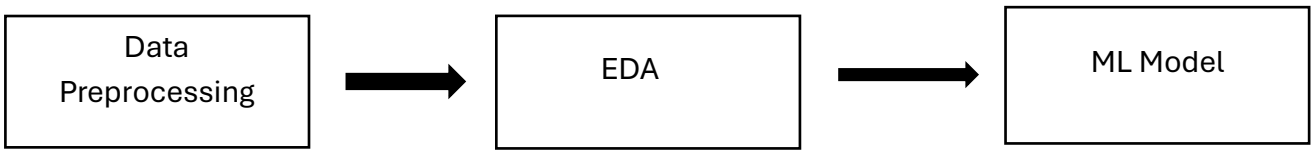
Unlike many prior efforts, our project places a strong emphasis on structured preprocessing, recursive feature elimination, and hyperparameter tuning across multiple models. We also consider practical deployment aspects such as model explainability and computational efficiency.

3. Methods

3.1 Dataset Description

- Source: UCI Machine Learning Repository – Cleveland Heart Disease Dataset
- Records: 303 patient samples
- Attributes: 13 clinical and personal features (e.g., age, sex, chest pain type, resting blood pressure) plus 1 binary target variable (target: 0 = no disease, 1 = presence of heart disease)

Methodology



3.2 Data Preprocessing Pipeline

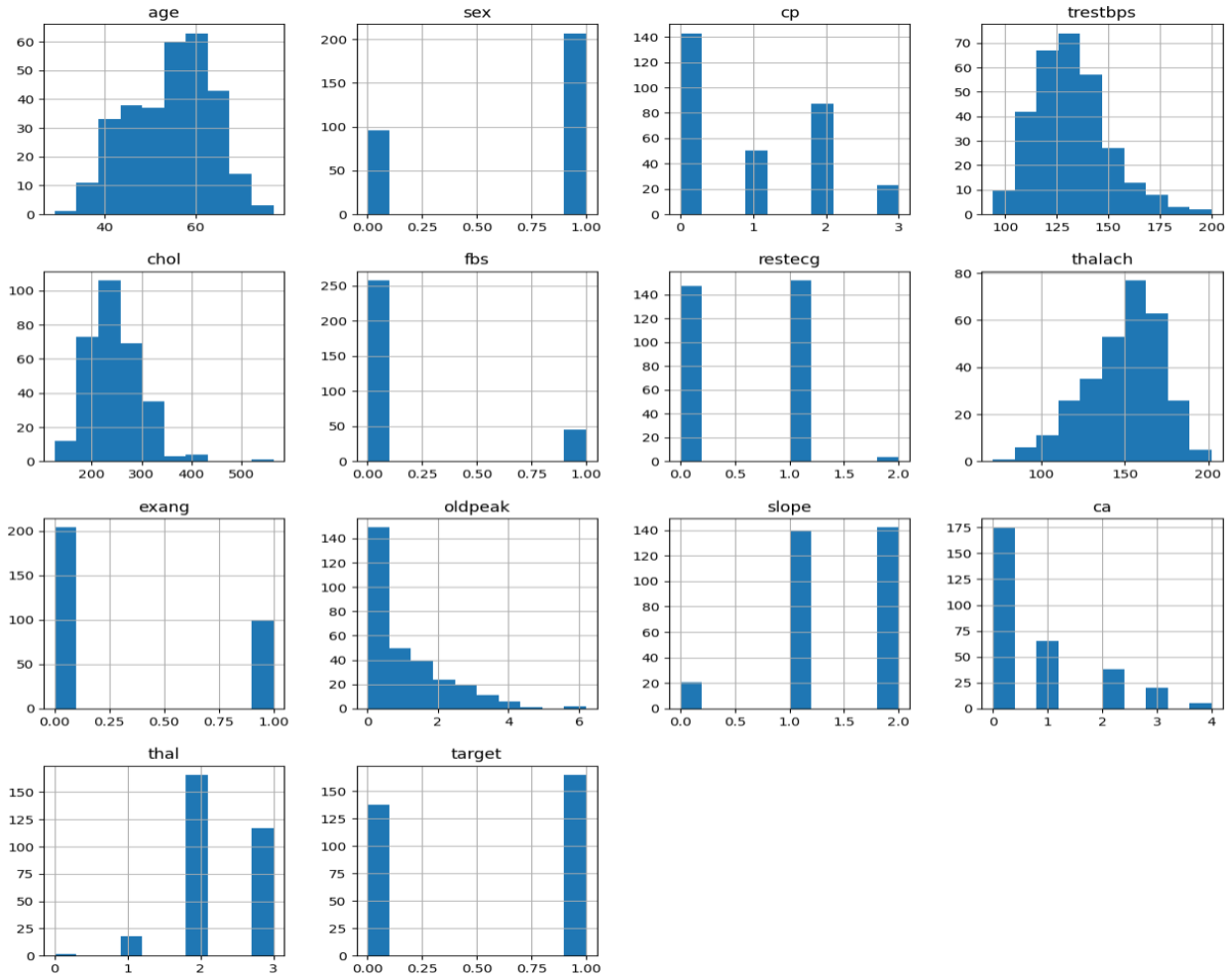
To prepare the dataset for model training, the following steps were performed:

- Duplicate Removal: A single duplicate row was identified and removed to ensure data integrity.
- Missing Value Treatment: The dataset contained no missing values, so imputation was not necessary.
- Outlier Handling: Outliers in cholesterol and age were capped using the Interquartile Range (IQR) method to minimize skew.
- Feature Scaling: Numerical features were scaled using Min-Max normalization, bringing values to the [0,1] range to improve model convergence and comparability.
- Categorical Encoding:
    - One-Hot Encoding for nominal features (cp, thal)
    - Binary Encoding for binary features (sex, fbs)

3.3 Exploratory Data Analysis (EDA)

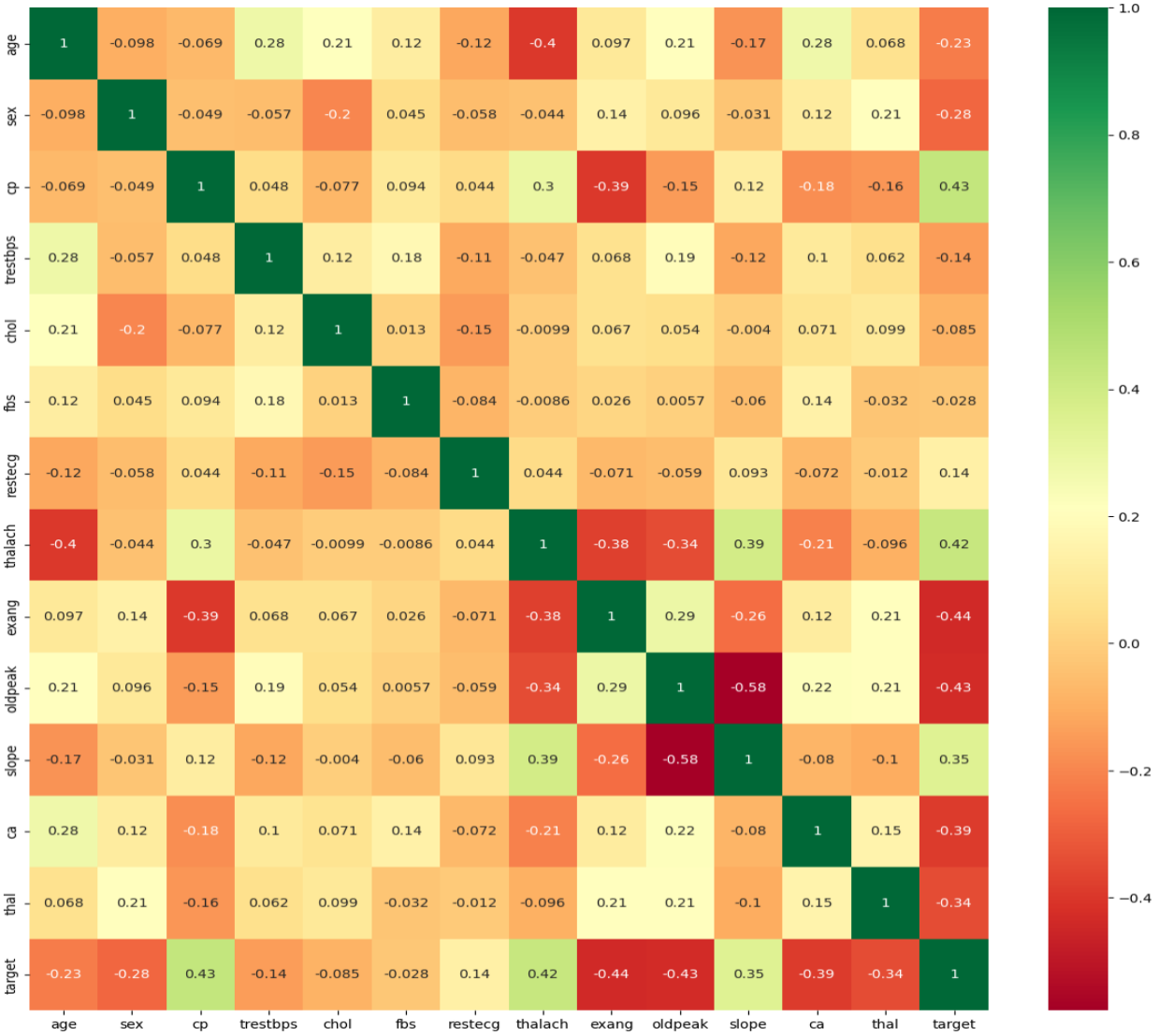
- Histogram

Using histogram, we can check numerical features in our dataset. The histograms represent the distribution of various features in the dataset. age shows the spread of ages, while sex and cp display the gender and chest pain type distributions. trestbps indicates resting blood pressure levels, and chol represents cholesterol levels. fbs is binary for fasting blood sugar, and restecg shows ECG results. thalach illustrates maximum heart rate, and exang represents the presence of exercise-induced angina. Other features, including oldpeak, slope, ca, and thal, are for additional heart conditions. Finally, target represents the distribution of presence/absence of heart disease.



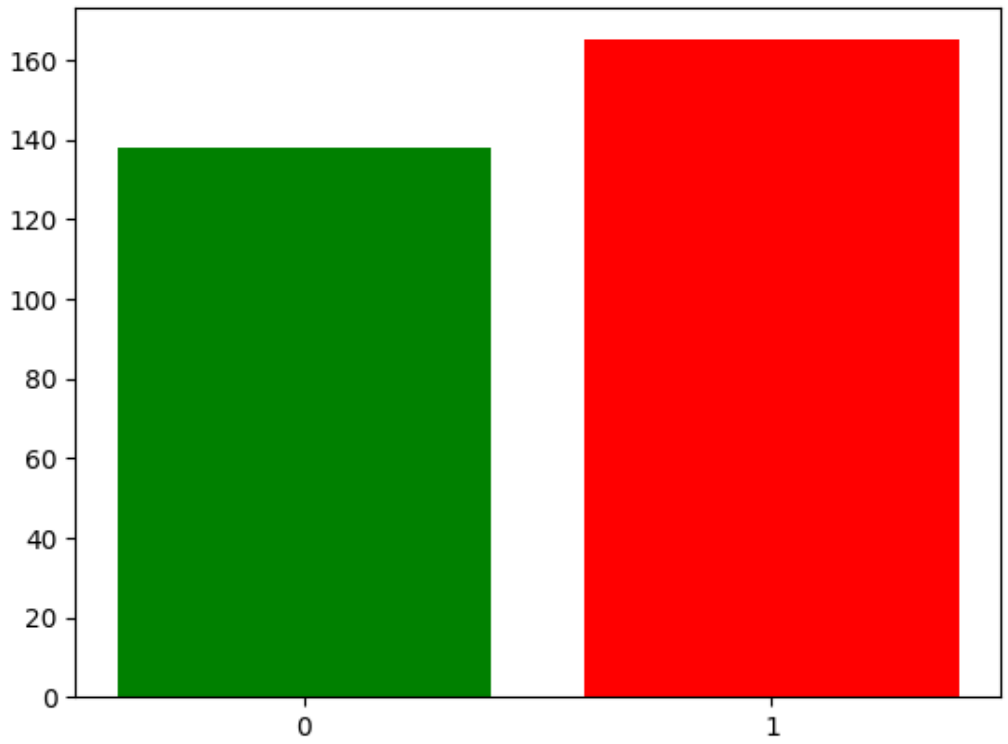
- Correlation Matrix

Using correlation matrix plot we can get how the data in data set is related to each other. We can observe high strong correlation between columns from the below image.



- Target/class distribution bar plot

The target variable represents the presence (1) or absence (0) of heart disease. The bar plot visually displays the distribution of these two classes within the dataset. This plot is essential for understanding whether the dataset suffers from class imbalance, which can significantly affect model performance and evaluation. In our case, the distribution between the two classes is relatively balanced, which reduces the risk of biased predictions toward the majority class and makes advanced balancing techniques (e.g., SMOTE) unnecessary. By inspecting this visualization early, we ensure that our models will have a fair opportunity to learn from both positive and negative cases during training.



3.4 Class Imbalance Analysis

The target variable showed a relatively balanced class distribution. Therefore, synthetic data generation techniques such as SMOTE were deemed unnecessary, avoiding the introduction of synthetic noise.

3.5 Feature Selection

We applied Recursive Feature Elimination (RFE) to identify the most predictive subset of features, resulting in the selection of:

- ca – Number of major vessels colored by fluoroscopy
- oldpeak – ST depression induced by exercise
- thalach – Maximum heart rate achieved

As RFE is model-dependent, we also considered the IAMB (Incremental Association Markov Blanket) algorithm (from previous coursework) as a model-agnostic alternative for future robustness.

3.6 Training and Testing Data

To determine how the model will generalize to new data, it applies the train_test_split function on the dataset training set and testing set. A common function used to divide the data into two subsets is train_test_split () from sklearn.model_selection. The feature variables X and target variable y are used as the input for this function, which divides them into training and testing sets. Key parameters are the key parameters. The test size determines the proportion of the data set to be tested. 33% of the data will be

used for testing if test_size is 0.33.A fixed seed for the random shuffling ensures reproducibility of the split. The training set used to train the model is represented by x_train and y_train. The testing set X_test and y_test represent the testing set used to evaluate the model. By splitting the data, the model will be trained on one subset and evaluated on a completely different, unseen subset. This helps the model generalize well on new data.

3.7 Principal Component Analysis (PCA)
Although not used for feature reduction (due to prior selection), PCA was used for visualization purposes. PCA helped visualize data distribution, separability, and clusters in 2D space—providing qualitative insights into feature behavior and class separability.

3.8 Software and Tools
- Programming Language: Python 3.x
- Development Environment: Jupyter Notebook / Google Colab
- Libraries:
  - Data Handling: pandas, numpy
  - Modeling: scikit-learn
  - Visualization: matplotlib, seaborn
  - Optional Feature Selection: Custom IAMB implementation (referenced from Homework 2)

3.9 Reproducibility
To ensure a streamlined and error-free modeling process, all data preprocessing and modeling steps were encapsulated using scikit-learn Pipelines. This approach guaranteed:
- Consistent application of transformations across all models
- Simplified hyperparameter tuning and cross-validation
- Reduced manual intervention and minimized human error
- Enhanced reproducibility, allowing others to replicate results with minimal setup

4. Experiments, Results, and Discussion
Model Evaluation Setup
All models were evaluated using 5-fold cross-validation to ensure robustness. Evaluation metrics primarily included accuracy and precision.

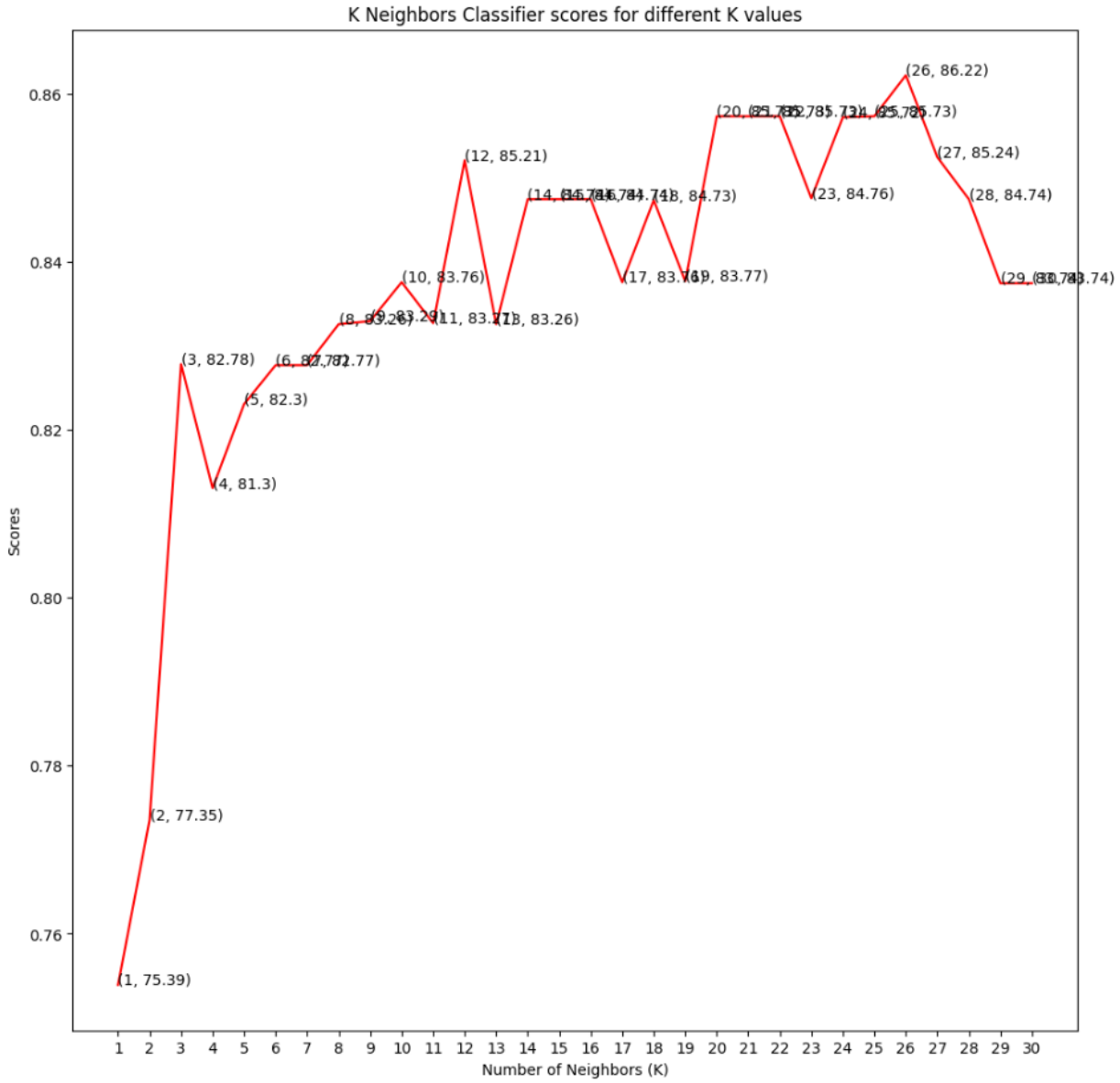
Models and Key Insights:

4.1 Logistic Regression
1. Logistic regression is used to predict heart disease using the Logistic Regression class from scikit-learn. Logistic regression is implemented to predict heart disease using the Logistic Regression class from scikit-learn.
2. The lbfgs solver was picked because it's great for binary classification tasks on small data sets.

3. Data preprocessing and model training are integrated into a pipeline that makes a streamlined workflow.
4. To make sure the model is robust and avoids overfitting, 5-fold cross-validation is used.
5. The evaluation tool is precision, with an average score of 86.22%
6. Logistic regression is a linear model that is characterized by simplicity, computational efficiency, and interpretability.
7. It provides a good starting point for binary classification problems such as predicting heart disease.
8. Pipelining reduces errors and keeps preprocessing steps consistent by keeping preprocessing steps consistent.
9. The high accuracy demonstrates the effectiveness of the model in distinguishing between the target classes.

```
0.8621951219512194
```

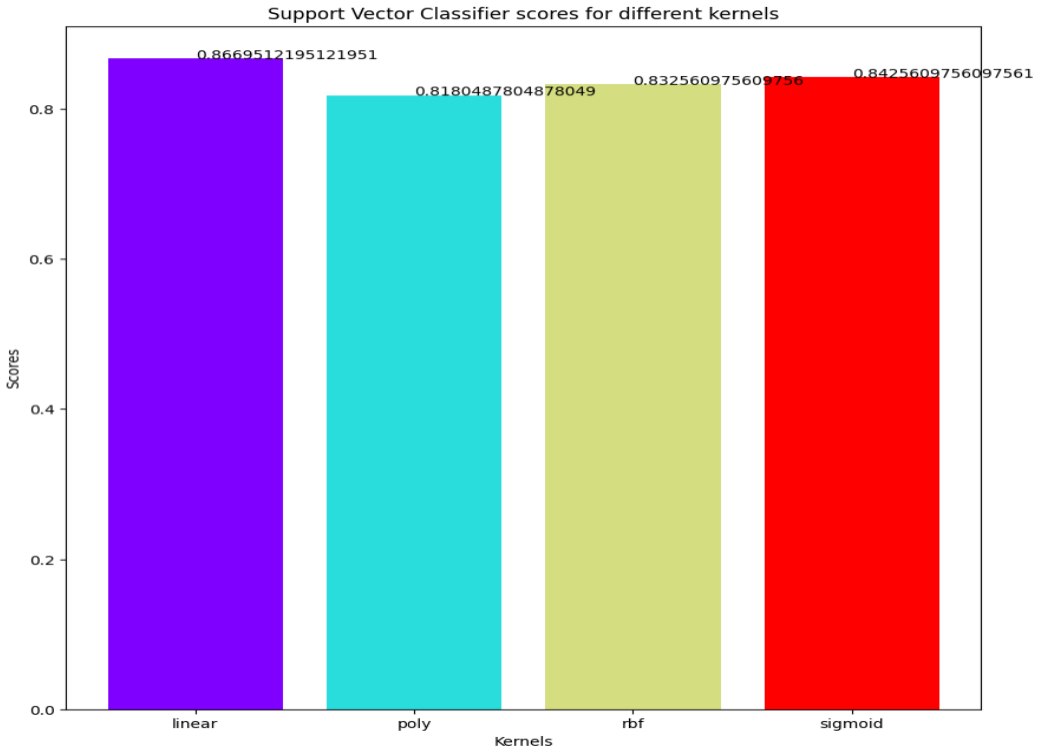
4.2 K Nearest Neighbors
1. Heart illness is classified by instance similarity using the K-Nearest Neighbors algorithm.
2. To find the optimal number of neighbors for classification, KNN is evaluated with k values ranging from 1 to 30.
3. A pipeline (make_pipeline) integrates data preprocessing (column_trans) and the KNN model into one consistent workflow. A pipeline (make_pipeline) integrates data preprocessing (column_trans) and the KNN
4. The model will be evaluated using 5-fold cross-validation, with accuracy being the scoring metric.
5. The effect of k on model performance can be seen by running cross-validation for each different value of k.
6. Look at the outcomes with a graph that displays precision scores for various k values, as well as any alterations in performance.
7. This graph has annotated values and corresponding precision in great detail.
8. KNN utilizes the local relationship of data, which makes it efficient in case of non-linear decision boundaries.
9. This plot shows the explicit tuning of a hyperparameter to reach the best classification performance.



4.3 Support Vector Classifier (SVC)

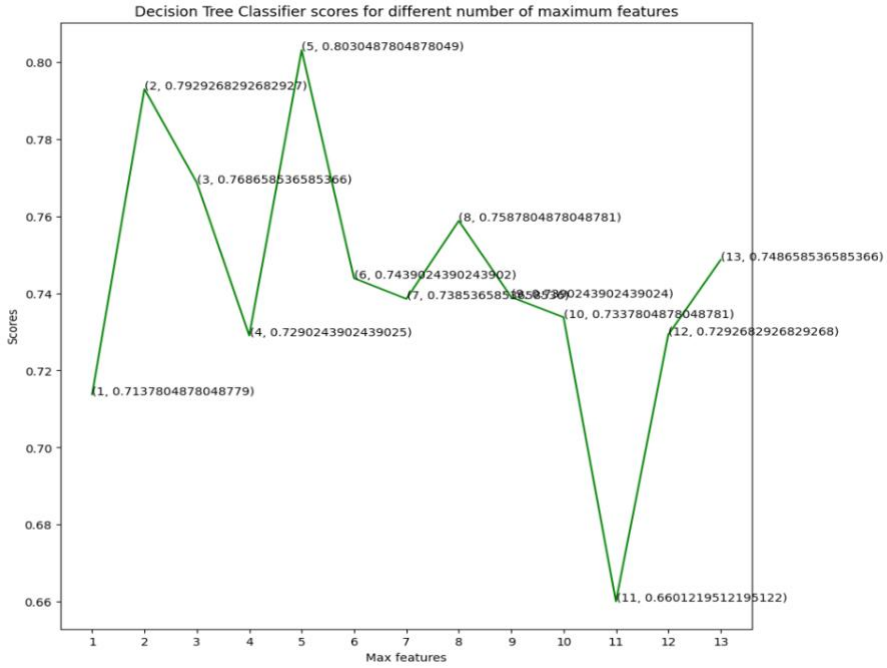
1. Different kernel functions are used to perform SVC to forecast heart disease.
2. To evaluate the best decision boundary that can be used for classification, four different kernels are used.
3. Make_pipeline combines preprocessing (column_trans) and the model, SVC, to ensure a seamless workflow.
4. Cross-validation with accuracy is used for model evaluation.
5. Cross-validation scores are computed for each kernel to compare their effectiveness in classification.
6. A bar chart shows the outcomes, with each kernel's precision score displayed above the bars for clarity.

7. The model's ability to handle linear and nonlinear data patterns is significantly affected by the choice of kernel.
8. The SVC with rbf and linear kernels performs better, demonstrating their suitability for this dataset.
9. This analysis highlights the importance of kernel selection in optimizing SVC performance for heart disease prediction.

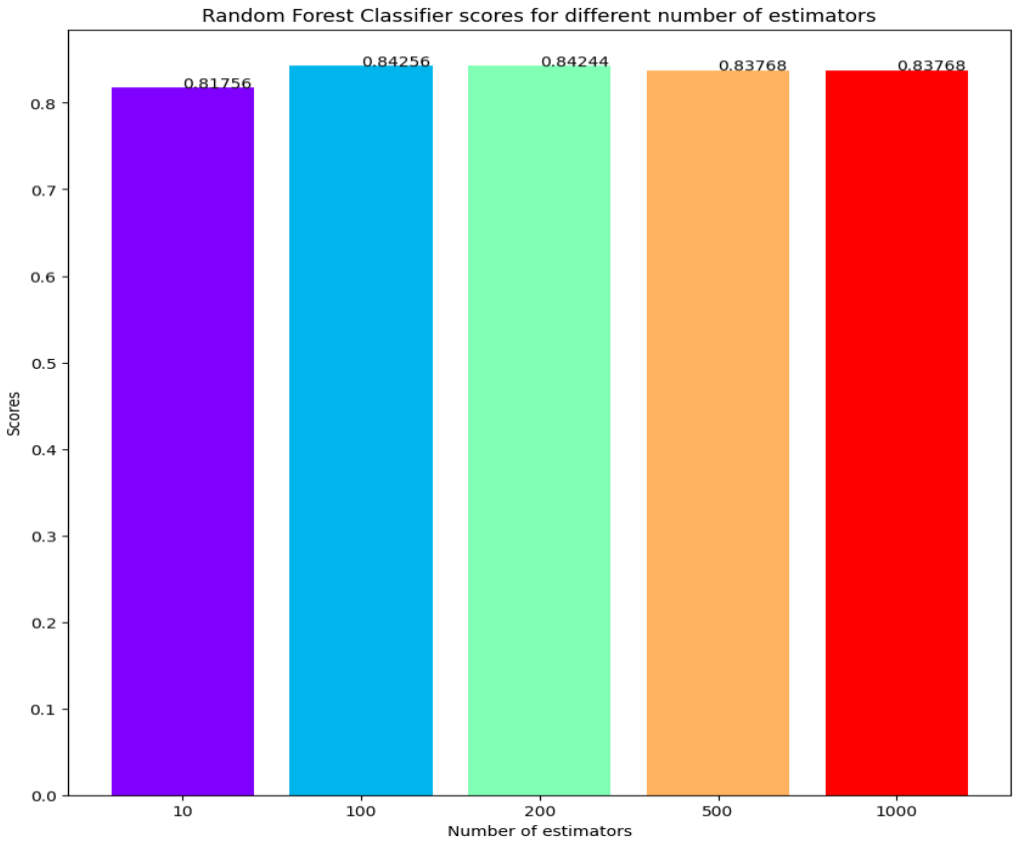


4.4 Decision Tree Classifier
1. To predict heart disease, a Decision Tree Classifier is implemented to explore different numbers of maximum features.
2. To examine its impact, the max_features parameter is set to 1 based on the total number of features in the dataset.
3. Make_pipeline links preprocessing (column_trans) and the Decision Tree model, ensuring consistency.
4. Evaluate model performance using 5-fold cross-validation, with accuracy as your scoring metric, to evaluate model performance.
5. Record cross-validation scores for each value in max_features to see the impact of feature count on model performance.
6. You can visualize results with a line plot, showing how accuracy changes based on the number of max features.

7. Annotations on the graph display the scores of each value of max_features, providing detailed insights about the performances. 8. Decision trees are good with nonlinear relationships and interactions between features, which makes them versatile for classification tasks.
9. The given analysis shows the main importance of tuning hyperparameters such as max_features for Decision Trees to be more accurate.



4.5 Random Forest Classifier
1. Random Forest classifier is applied to predict heart disease using ensemble learning.
2. There are different numbers of estimators representing the number of decision trees.
3. A pipeline combines column transformation and the Random Forest model, ensuring a well-organized process.
4. Perform five times cross-validation on the model, evaluating its performance using precision as the evaluation criterion.
5. Cross-validation scores for each value can be calculated with the help of the n_estimators parameter to demonstrate variation in performance as ensembles get larger.
6. In a bar chart, each bar indicates an accuracy score for the corresponding number of estimators.
7. The scores in the graph are annotated, showing the improvement in performance as more trees are added.
8. Random Forest handles complicated datasets exceptionally well because it reduces overfitting and contains feature interactions.



4.6 Gaussian NB

1. The Gaussian Naive Bayes for heart disease prediction are implemented with the GaussianNB class from scikit-learn.
2. A seamless workflow is achieved by integrating the model into a pipeline with preprocessing (column_trans)
3. 5-fold cross-validation with precision is used to evaluate the model.
4. It achieves an average cross-validation accuracy of 80.83%, which is good performance for this classification task.
5. Gaussian Naive Bayes is computationally efficient and good to go with those datasets where features are supposed to follow a Gaussian distribution.

```
0.8425609756097561
```

Performance (Accuracy):

Model	Accuracy
Logistic Regression	86.2%

Model	Accuracy
KNN	86.2%
SVC (Best)	86.6%
Decision Tree	80.3%
Random Forest	84.2%
Naive Bayes	80.8%

Model Tuning
- SVC: Tuned C parameter
- KNN: Grid Search → best k = 26
- Random Forest: Random Search for n_estimators and max_depth

Insights
- SVC showed the best generalization.
- RFE-selected features had strong predictive power.
- Class balance was adequate—no need for SMOTE.
- PCA was explored for feature visualization, not used for dimensionality reduction due to prior feature selection.

5. Conclusion

We successfully built and evaluated ML models for heart disease prediction. The Support Vector Classifier achieved the highest accuracy (86.6%). Strong features identified included thalach, ca, and oldpeak.

Limitations:
- Small dataset size (303 records)
- Risk of overfitting in some models (e.g., Decision Tree)

Future Work:
- Use of advanced models (XGBoost, neural networks)
- External validation with additional patient data
- Deployment in clinical settings with feedback from healthcare professionals
- Model explainability using SHAP and LIME
- Build an automated ML pipeline with cloud deployment

6. Data and Software Availability

Dataset: UCI ML Repository – Heart Disease

Code Repository:

7. References

1. Detrano, R., et al. (1989). International application of a new probability algorithm for the diagnosis of coronary artery disease. American Journal of Cardiology. DOI: 10.1016/0002-9149(89)90524-2
2. Framingham Heart Study. (2023). Retrieved from https://www.framinghamheartstudy.org/
3. Dua, D., & Graff, C. (2019). UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences. https://archive.ics.uci.edu/ml/datasets/heart+disease

8. Contributions

Lakshmi Wajja

- Wrote and edited sections on Experiments, Results, and Discussion, focusing on model performance comparison.
- Implemented SVC and Gaussian Naive Bayes models and performed hyperparameter tuning.
- Helped maintain a clean and reproducible scikit-learn pipeline structure.
- Contributed to summarizing insights and designing final charts for the presentation.
- Coordinated integration of all components into a polished final report.

Rebecca Dara

- Worked on the Methods and Model Evaluation sections of the final report.
- Took lead on implementing and analyzing K-Nearest Neighbors and Random Forest models in the .ipynb file.
- Contributed to the EDA visualizations (histograms, correlation matrix, target distribution).
- Helped with organizing and sequencing content in the presentation slides.
- Reviewed and validated code outputs and visualizations for accuracy.

Yaswitha Kalapala

- Contributed to writing and editing the Introduction and Related Work sections of the report.
- Participated in data preprocessing and exploratory data analysis in the Jupyter notebook.
- Assisted in tuning and evaluating models like Logistic Regression and Decision Tree.
- Collaborated on designing presentation slides, contributing visuals and bullet points.
- Took part in group discussions and helped finalize the project conclusions.

Sujish Kumar Indrajith

- Contributed to the Conclusion, Feature Selection, and Reproducibility sections of the report.
- Helped prepare and format the final Jupyter notebook, including annotations and cleanup.

- Assisted in training/testing split setup, PCA visualization, and fine-tuning pipeline integration.
- Took the lead in delivering the final presentation on behalf of the team.
- Supported the team in finalizing slides and organizing presentation flow.

Although one team member delivered the final presentation, all team members contributed equally to its content and structure. All aspects of the project technical implementation, report writing, slide creation, and analysis were carried out collaboratively.