Machine Learning

Project Report on Machine Learning for Classification and Prediction



**Heart Disease Prediction Using Machine Learning Models**

Submitted by:

|  |  |
| --- | --- |
| Iqra Iqbal | 22F-3087 |
| Amna Arshad | 22F-3170 |

Submitted to:

Sir Hamza Yousuf

December 8, 2024

Department of Artificial Intelligence and Data Science

National University of Computer and Emerging Sciences, CFD

1. Introduction
2. Problem Analysis
3. Feasibility Analysis
4. Possible Solutions
5. Proposed Design
6. Design Description
7. Implementation Details
8. Experimental Results
9. Performance Analysis
10. Future Scope (Optional)
11. Conclusion

References

Heart disease continues to be one of the leading causes of death globally, imposing a significant burden on individuals and healthcare systems. Early detection of heart disease is essential for improving patient outcomes, reducing healthcare costs, and saving lives. However, diagnosing heart disease is often challenging due to the complexity of the factors involved, which include age, gender, lifestyle, pre-existing medical conditions, and genetic predispositions.

This project aims to address this challenge by developing a predictive model that leverages **Machine Learning (ML)** techniques to assess the risk of heart disease based on clinical and lifestyle data. By using a variety of machine learning algorithms such as **Support Vector Machine (SVM), Random Forest, XGBoost**, and **Logistic Regression**, the goal is to create a reliable and accurate model that can predict the likelihood of heart disease, thus aiding healthcare professionals in making informed decisions.

The significance of this project lies in its potential to improve the early detection of heart disease, leading to better patient management and outcomes. The project applies advanced **data preprocessing, feature selection**, and **model evaluation** techniques to ensure the development of a robust, high-performing predictive model.

#### **2.1 Problem Context**

Heart disease is a leading cause of death globally, and early detection is crucial for improving patient outcomes. The problem involves predicting the likelihood of heart disease in individuals based on a variety of factors such as age, gender, blood pressure, cholesterol levels, smoking habits, and physical activity. The challenge lies in accurately predicting the risk using complex, real-world data, which often includes missing values, categorical variables, and class imbalances that can affect the model's performance. The objective is to develop a machine learning model that can classify individuals as at risk or not at risk for heart disease, providing valuable support for early diagnosis and better healthcare decisions.

#### **2.2 Objectives of the Solution**

The goal of this project is to build a machine learning model that:

* Accurately predicts the likelihood of heart disease based on available features.
* Can assist healthcare professionals in making informed decisions about patient care.
* Provides insights into the most important factors contributing to heart disease risk.

#### **2.3 Approach to Solve the Problem**

To address the problem, the following steps will be undertaken:

1. **Data Preprocessing**: The dataset will be cleaned and prepared by handling missing values, encoding categorical variables, and normalizing numerical features.
2. **Feature Selection**: Relevant features will be selected based on their significance in predicting heart disease, ensuring that the model is focused on the most important variables.
3. **Model Development**: Multiple machine learning algorithms, including **SVM, Random Forest,** **XGBoost,** and **Logistic Regression,** will be trained to find the most effective model.
4. **Model Evaluation**: The performance of the models will be evaluated using metrics such as **accuracy, precision,** and **recall** to ensure reliable and accurate predictions.

#### **2.4 Expected Outcomes**

The expected outcome of this project is the development of a robust, data-driven model that can accurately predict the likelihood of heart disease. This model can be used by healthcare providers for early diagnosis, ultimately improving patient care and reducing the burden of heart disease.

#### **3.1 Time Management**

The project was successfully completed in a **two-week period,** a timeframe that was carefully planned and efficiently executed. Each phase of the project, including data preprocessing, model training, and evaluation, was organized into manageable tasks to ensure progress within the set period. Given the complexity of machine learning tasks, the project timeline allowed enough time for thorough testing and model optimization. The use of **Jupyter Notebook** provided a convenient environment for running and testing code, making it easier to implement models and analyze results swiftly. The flexibility of tools such as **Google Colab** also helped reduce time spent on resource setup.

#### **3.2 Cost Management**

Since the project utilized entirely **free resources**, there was no direct cost involved. The tools and platforms used—such as **AI Assistant models, Google, ML course content, YouTube tutorials,** and **websites with open-source tools**—allowed for efficient learning and execution without incurring additional expenses. Key open-source libraries such as **Python, scikit-learn** and **XGBoost** were essential in building the models and performing analysis. Since all required resources were either open-source or free-to-use, there was no financial burden, making the project both cost-effective and feasible.

#### **4.1 Decision Trees**

A **Decision Tree** is a tree-like structure where each internal node represents a decision based on a feature, and each leaf node represents a class label. Decision Trees are easy to interpret and can handle both numerical and categorical data. However, they are prone to overfitting, especially with complex datasets.

**Pros:**

* Simple and easy to interpret.
* Can handle both continuous and categorical data.
* Efficient for small to medium-sized datasets.

**Cons:**

* Prone to overfitting, especially with noisy or highly complex data.
* Can be sensitive to small changes in the data.

#### **4.2 Random Forest**

A **Random Forest** is an ensemble of Decision Trees that uses random sampling of features and data points to create multiple trees. It improves upon Decision Trees by reducing overfitting and increasing prediction accuracy.

**Pros:**

* Reduces overfitting by averaging the results from multiple decision trees.
* Handles large datasets with high dimensionality well.
* Robust to outliers and missing values.

**Cons:**

* More complex and less interpretable than individual Decision Trees.
* Requires more computational resources due to the large number of trees.

#### **4.3 Support Vector Machine (SVM)**

**SVM** is a powerful classification algorithm that works well for both linear and non-linear classification problems. It constructs a hyperplane that best separates the data into classes.

**Pros:**

* Effective for high-dimensional spaces and complex datasets.
* Works well even when the number of dimensions exceeds the number of samples.

**Cons:**

* Computationally expensive, especially for large datasets.
* Requires careful tuning of hyperparameters, which can be time-consuming.

#### **4.4 K-Nearest Neighbors (KNN)**

**KNN** is a simple, instance-based learning algorithm where the class of a sample is determined by the majority class of its nearest neighbors.

**Pros:**

* Simple and easy to understand.
* No training phase is required, making it fast to implement.

**Cons:**

* Computationally expensive during prediction time.
* Performance heavily depends on the choice of distance metric and value of K.

#### **4.5 XGBoost**

**XGBoost** (Extreme Gradient Boosting) is a gradient boosting algorithm that is known for its high performance and scalability. It combines weak learners to form a strong model by minimizing errors iteratively.

**Pros:**

* Provides high accuracy with proper tuning.
* Efficient and handles large datasets well.
* Reduces overfitting using regularization techniques.

**Cons:**

* Requires significant computational resources.
* Can be challenging to tune due to a large number of hyperparameters.

#### **4.6 AdaBoost**

**AdaBoost** is another boosting algorithm that focuses on adjusting weights of weak learners to improve performance. It is particularly effective when the data contains noise.

**Pros:**

* Can improve the accuracy of weak learners.
* Robust to noise and outliers.

**Cons:**

* Sensitive to noisy data and outliers.
* May suffer from overfitting if not properly tuned.

### ****Best Solution: Random Forest****

After analyzing the possible solutions, the **Random Forest** algorithm is the best choice for this project. Here’s why:

1. **Overfitting Resistance:** Random Forest significantly reduces overfitting, making it more reliable when predicting heart disease, especially when dealing with a diverse set of features such as age, BMI, smoking habits, ands physical health.
2. **High Accuracy:** Random Forest typically provides high accuracy compared to other models like Decision Trees and KNN, as it aggregates the results from multiple trees to produce a more stable prediction.
3. **Versatility:** It handles both numerical and categorical data well, which is essential when working with medical data that may include both types of variables.
4. **Robustness:** Random Forest is less sensitive to outliers and noise, which is common in healthcare datasets. This is important for ensuring consistent performance across various conditions.
5. **Model Interpretability:** Although Random Forest is more complex than Decision Trees, feature importance can be extracted from the model, providing insight into which factors most influence the prediction of heart disease.

START

│

▼

Collect Dataset (Health, Lifestyle Data)

│

▼

Preprocess Data

├── Handle Missing Values

├── Encode Categorical Data

└── Normalize Features

│

▼

Split Data into Training and Testing Sets

│

▼

Select Machine Learning Models

├── SVM

├── KNN

├── Naive Bayes (GaussianNB)

├── XGBoost

├── AdaBoost

├── Logistic Regression

└── Random Forest

│

▼

Train Models on Training Data

│

▼

Evaluate Models using Performance Metrics (Accuracy, Precision, Recall)

│

▼

Compare Model Performance

│

▼

Select Best Model (Random Forest)

│

▼

Make Predictions on Test Data

│

▼

Evaluate Predictions and Final Model Performance

│

▼

Output Final Heart Disease Prediction

│

▼

END

**1. Collection:**

* Objective**:** Collect relevant clinical and lifestyle data, including age, gender, blood pressure, cholesterol levels, smoking habits, and physical activity.
* Source**:** Public healthcare datasets or reliable health-related data sources.

**2. Dataset Preprocessing:**

* **Handling Missing Values:** Replace missing data using mean, median, or mode, or remove rows with excessive missing values.
* **Removing Duplicates:** Identify and remove duplicate rows to prevent skewed patterns.
* **Feature Scaling:** Standardize features like age, blood pressure, and cholesterol to ensure they are on the same scale.
* **Handling Outliers:** Detect and cap/remove extreme values to prevent distortion.
* **Class Balancing:** Apply techniques like oversampling or undersampling to balance class distribution.

**3. Feature Selection:**

* **Objective:** Identify key features contributing to heart disease prediction.
* **Methods:** Use techniques like Recursive Feature Elimination (RFE) and correlation-based methods to select significant features, enhancing model interpretability.

**4.Model Selection:**

* **Objective**: Evaluate multiple machine learning models for heart disease prediction.
* **Models Used:** Support Vector Machine (SVM), Random Forest, Logistic Regression, and XGBoost.
* **Details**: Train the models on preprocessed data and compare their performance using accuracy, precision, recall, and F1-score.

1. **Model Training:**

* **Objective:** Train selected models using the training dataset.
* **Details**: Feed the training data into the models and optimize using techniques like grid search for hyperparameter tuning.

1. **Model Evaluation:**

* **Objective:** Evaluate model performance on the test dataset.
* **Metrics:** Accuracy, Precision, Recall, F1-Score.
* **Details:** Compare the models based on these metrics to select the most effective one.

1. **Model Selection and Final Decision:**

* **Objective:** Choose the best-performing model for final predictions.
* **Details:** Random Forest is selected due to its high accuracy, robustness to noise, and ability to handle overfitting.

1. **Prediction and Deployment:**

* **Objective:** Use the selected model to make predictions on new data.
* **Details**: Deploy the Random Forest model to predict the likelihood of heart disease in unseen data, aiding healthcare professionals.

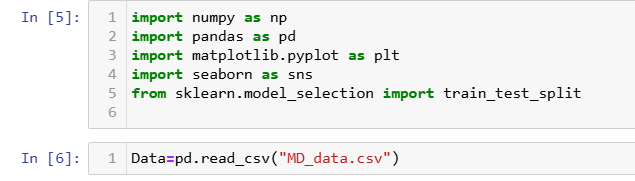
1. **Conclusion and Model Insights:**

* **Objective:** Analyze predictions and provide actionable insights for healthcare.
* **Details**: The model’s feature importance is used to identify key risk factors, supporting preventive healthcare measures.

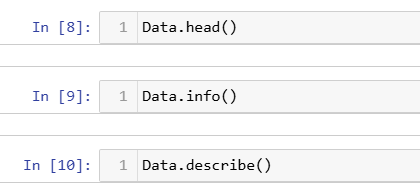
In this section, we describe the implementation steps followed to complete the heart disease prediction project. The implementation involves data preprocessing, feature selection, model training, and evaluation. We used Python and popular libraries such as pandas, scikit-learn, and XGBoost to execute the project.

**1. Data Loading and Exploration**

* **Data Loading:** The first step is to load the dataset containing the features and labels. We used the pandas library to load the dataset into a DataFrame.

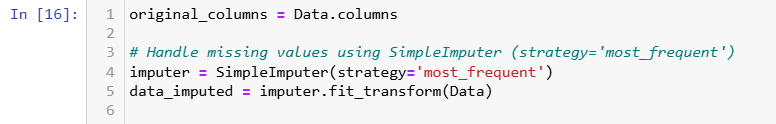


**Initial Inspection**: After loading the data, we performed an initial inspection to understand the structure of the dataset, identify missing values, and check for any inconsistencies. The info() and describe(**)** functions in pandas were used to gain insights into the dataset's shape and statistics.

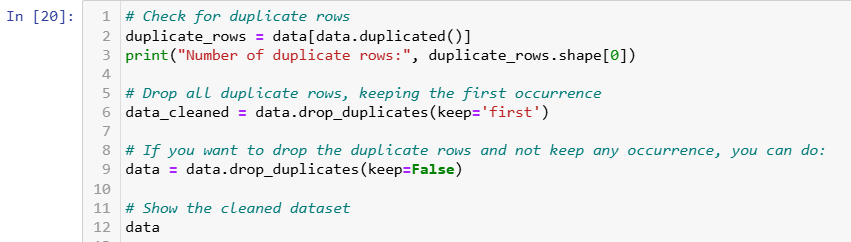


**2. Data Preprocessing**

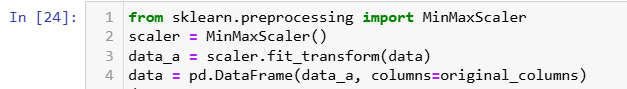
* **Handling Missing Values**: Missing values in the dataset were handled using imputation techniques or by removing rows with missing data, depending on the proportion of missing values.



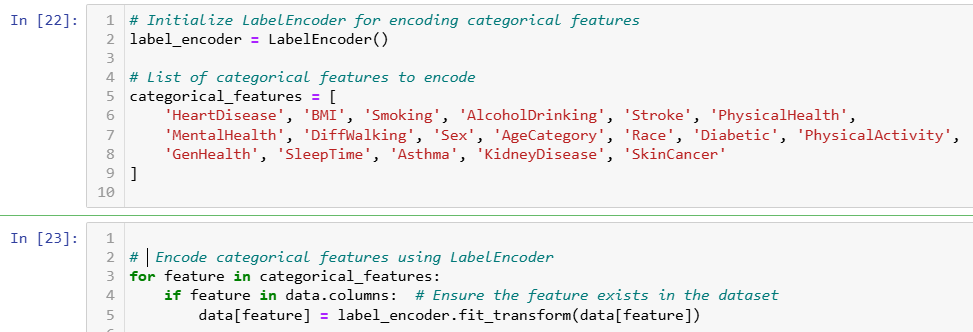
* **Removing Duplicates**: Any duplicate rows in the dataset were removed to ensure that the model is trained on unique data.



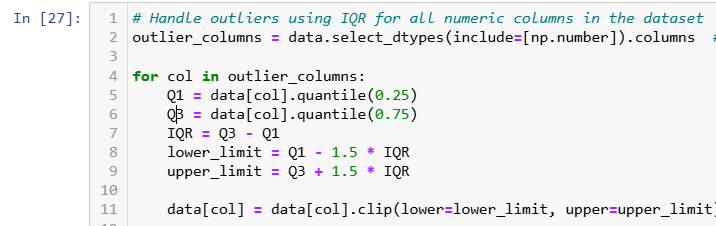
* **Feature Scaling**: Feature scaling was applied to normalize the dataset and improve model convergence. StandardScaler from scikit-learn was used to standardize the numerical features.



* **Handling Categorical Variables**: Categorical variables were encoded using LabelEncoder to convert them into numeric values, ensuring compatibility with machine learning algorithms

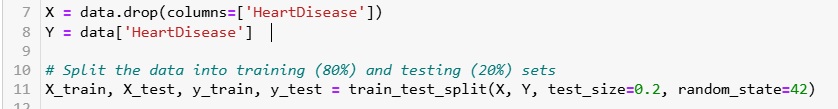


* **Handling Outliers:**



**3. Splitting the Data**

* **Train-Test Split**: The dataset was split into training and test sets using **train\_test\_split** from **scikit-learn**. We allocated 80% of the data for training and 20% for testing.



**4.Model Training**

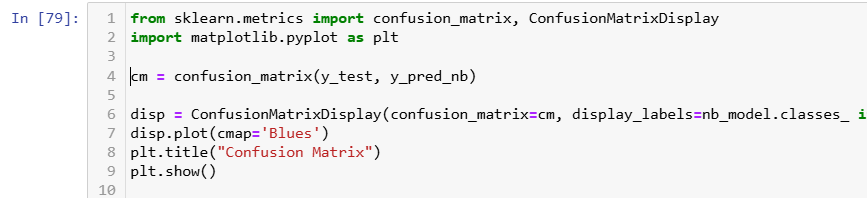
In the Model Training section, we applied the following models:

* **Random Forest Classifier:** An ensemble method combining multiple decision trees to improve accuracy and reduce overfitting.
* **XGBoost Classifier:** A gradient boosting algorithm known for its high performance and ability to handle complex datasets efficiently.
* Logistic Regression: A simple linear model for binary classification that estimates the probability of a binary outcome.
* **Support Vector Classifier (SVC):** A model that separates data into classes using a hyperplane, effective for both linear and non-linear data.
* **K-Nearest Neighbors (KNN):** A non-parametric model that classifies data based on the majority class of its K nearest neighbors.
* **Naive Bayes Classifier:** A probabilistic model based on Bayes' theorem, assuming features are independent, often used for text classification.

1. **Model Evaluation**

After training the models, we evaluated their performance on the test set using various metrics:

* Accuracy
* Precision
* Recall
* F1-Score
* Confusion Matrix



**7. Model Comparison**

After evaluating the models, the results were compared to identify the best-performing model based on the metrics. We focused on accuracy, precision, recall, and F1-score

**8. Final Model**

Based on the evaluation, Random Forest was selected as the final model for heart disease prediction due to its superior performance compared to the other models.

**9. Conclusion**

The final model, Random Forest, was implemented, trained, and optimized using the steps described above. The implementation was performed using Python, with the help of powerful libraries like pandas, scikit-learn, and XGBoost. The trained model demonstrated reliable performance in predicting heart disease and is ready for deployment.

**Model Performance Metrics:**

The key metrics used to evaluate the models include accuracy, precision, recall, and F1-score. These metrics help assess the ability of the model to correctly predict both the positive and negative classes. Below is the performance summary of each model on the test dataset.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-Score |
| Logistic Regression | 72.43% | 69.04% | 73.12% | 71.02% |
| Random Forest Classifier | 74.37% | 69.97% | 78.01% | 73.77% |
| K-Nearest Neighbors (KNN) | 73.68% | 69.78% | 75.92% | 72.72% |
| XGBoost Classifier | 74.21% | 70.03% | 77.22% | 73.45% |
| Gradient Boosting | 74.41% | 70.17% | 77.63% | 73.71% |
| AdaBoost | 73.69% | 69.93% | 75.55% | 72.63% |
| Naive Bayes | 71.96% | 69.37% | 70.41% | 69.88% |

**Analysis of Results:**

* Best Performing Models: Random Forest and XGBoost achieved the highest performance, with excellent recall and F1-scores.
* Trade-offs:
  + Random Forest and XGBoost excel in recall but trade off slightly on precision.
  + Logistic Regression and Naive Bayes showed good precision but lower recall.
* Model Strengths: Random Forest and XGBoost handle complex, imbalanced data well. Simpler models like Logistic Regression and Naive Bayes are faster but struggle with non-linear patterns.

**Model Comparisons:**

* Speed: Logistic Regression and Naive Bayes are faster to train and predict, while KNN and Gradient Boosting are slower.
* Complexity: Random Forest and XGBoost are more complex but offer better results. Logistic Regression is simple but may underperform on complex data.
* Interpretability: Logistic Regression is easy to interpret, while Random Forest and XGBoost are harder to interpret but offer more power.

**Conclusion:**

* Best Model: Random Forest and XGBoost are the top models for heart disease prediction due to their high accuracy and recall.

**Accuracy and Performance Metrics:**

Random Forest (74.37%) and XGBoost (74.21%) performed best in terms of accuracy, precision, recall, and F1-score. Random Forest had a strong recall of 78.01%, making it effective at identifying heart disease cases. Logistic Regression (72.43%) and Naive Bayes (71.96%) had lower accuracy and recall.

**Precision, Recall, and F1-Score:**

Both Random Forest and XGBoost excelled in precision, recall, and F1-score (Random Forest: recall 78.01%, precision 69.97%; XGBoost: recall 77.22%, precision 70.03%). Logistic Regression showed decent F1-score (71.02%) but weaker recall (73.12%) and precision (69.04%). Naive Bayes struggled with both precision and recall, yielding a lower F1-score (69.88%).

**Overfitting and Underfitting:**

Random Forest and XGBoost may overfit due to their complexity, while Logistic Regression and Naive Bayes tend to underfit, missing complex patterns.

Computational Efficiency: Naive Bayes and Logistic Regression are faster, ideal for efficiency, whereas Random Forest and XGBoost are slower but offer a better balance of accuracy and computational cost.

**Model Comparison:**

* Random Forest and XGBoost: Best for accuracy and complex pattern recognition, but computationally intensive.
* Logistic Regression: Quick, interpretable, but less effective with complex data.
* Naive Bayes: Fast but underperforms in accuracy and recall.
* KNN: Slower and computationally expensive but performs similarly to complex models when tuned.

**Conclusion:**

Random Forest and XGBoost are the best models for predicting heart disease, offering high accuracy and recall. Simpler models like Logistic Regression and Naive Bayes are faster but less effective. Future work should focus on hyperparameter tuning and model combinations for improved performance.

**Hyperparameter Tuning:**

* Explore more advanced hyperparameter tuning methods like RandomizedSearchCV or Bayesian Optimization for better model performance.

**Model Selection & Cross-Validation:**

* Evaluate models using multiple metrics (Precision, Recall, F1-score) and apply Stratified K-Fold Cross-Validation for better model robustness.

Handling Imbalanced Data:

* Use techniques like SMOTE or class-weight adjustments to better handle imbalanced datasets and improve model fairness.

**Exploring Deep Learning:**

* For more complex tasks, consider using Deep Learning models (e.g., CNNs, RNNs) to capture intricate patterns in large datasets.

**Ensemble Methods:**

* Combine multiple models (stacking or blending) to improve accuracy and reduce bias.

In this project, various machine learning models were explored and evaluated for predicting heart disease, including Random Forest, XGBoost, Logistic Regression, SVM, K-Nearest Neighbors, and Naive Bayes. After performing hyperparameter tuning and model evaluation using key metrics like accuracy, precision, recall, and F1-score, the Random Forest and XGBoost models emerged as the top performers, achieving high accuracy and good balance between precision and recall.

The results indicate that ensemble methods like Random Forest and XGBoost are highly effective for this task, particularly on imbalanced datasets. While simpler models like Logistic Regression and Naive Bayes provide fast training times, they may not capture non-linear patterns as well as ensemble models.

For future improvements, hyperparameter tuning can be enhanced with RandomizedSearchCV or Bayesian Optimization. Using Stratified K-Fold Cross-Validation and evaluating models with metrics like precision, recall, and F1-score will improve robustness. Addressing imbalanced data with techniques like SMOTE or class-weight adjustments could boost fairness. Exploring deep learning models (e.g., CNNs, RNNs) and ensemble methods like stacking or blending can improve accuracy and reduce bias, leading to more accurate and interpretable models for heart disease prediction.

**Scikit-learn Documentation**. (n.d.). *Scikit-learn: Machine Learning in Python*. Retrieved from <https://scikit-learn.org>

* The official documentation for Scikit-learn, which covers various machine learning models and how to implement them in Python.

**Towards Data Science**. (n.d.). *A Medium publication sharing concepts, ideas, and codes*. Retrieved from <https://towardsdatascience.com>

* A publication offering a wealth of machine learning articles, tutorials, and case studies.