Title: Heart Attack Prediction

Project Report

TEB2043

Data Science

https://github.com/xinghao2003/TEB2043-Data-Science-Group-Project

Author	Student ID
Wong Xing Hao	21000612
Cheng Pin-Jie	21000548
Loo Pei Xin	21000483
Nur Adriana Binti Muhamad Aizam	21000162

1.0 Problem and solution

Identifying High-Risk Individuals. Patients with a higher chance of a heart attack need to be accurately identified to facilitate timely intervention. Develop a predictive model using machine learning algorithms to classify individuals based on their risk of a heart attack.

Understanding Key Contributors to Heart Attack Risk. Identify the most influential factors contributing to the likelihood of a heart attack. Conduct feature importance analysis to determine the key variables affecting the prediction and provide insights into preventive measures.

2.0 Motivation and Background

Cardiovascular diseases, including heart attacks, stand as a major cause of global death and disability. According to the World Health Organization (WHO), approximately 17.9 million deaths occur annually due to cardiovascular diseases, constituting roughly 31% of global fatalities. The early identification and precise prediction of heart attack risks are pivotal for implementing timely preventive measures, personalized therapies, and optimizing healthcare resources.

The study of data science is critical in extracting useful information from big and complex datasets, allowing healthcare workers to identify patterns and risk factors related with heart attacks. Data scientists contribute to the creation of reliable risk prediction models by using advanced analytics, machine learning algorithms, and predictive modelling. These models have the potential to transform patient care by enabling proactive and personalised treatments based on an individual's risk profile.

Data science's multidisciplinary characteristics allows the combination of varied data sources such as medical records, lifestyle data, and genetic information. An in-depth review of these data sets improves our understanding of the complex interactions between multiple variables and their impact on cardiovascular health. Furthermore, the use of data-driven approaches in healthcare promotes evidence-based decision-making, optimises therapies, and improves patient outcomes.

3.0 Dataset

The dataset used for this heart attack analysis and prediction was obtained from Kaggle. There is a total of 303 records with 1 duplicate in the dataset.

Attributes in dataset:

- **age**: age of the patient (in years)
- **sex**: gender of the patient
- **cp**: categorized chest pain types
    - 0 - typical angina
    - 1 - atypical angina
    - 2 - non-anginal pain
    - 3 - asymptomatic
- **trtbps**: resting blood pressure (in mm Hg)
- **chol**: cholesterol (in mg/dl) fetched via BMI sensor.
- **fbs**: fasting blood sugar > 120 mg/dl (1 = true, 0 = false)
- **restecg**: types of resting electrocardiographic results
    - 0 - normal,
    - 1 - having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV),
    - 2 - showing probable or definite left ventricular hypertrophy by Estes' criteria
- **thalachh**: maximum heart rate achieved.
- **exng**: exercise-induced angina (1 = yes, 0 = no)
- **oldpeak**: ST depression induced by exercise relative to rest.
- **slp**: slope of the peak exercise ST segment (0-2)
- **thall**: Thalassemia rate (0-3)
- **caa**: number of major vessels (0-3)
- **output** (target): 0= less chance of heart attack, 1= more chance of heart attack

4.0 Methodology

Exploratory Data Analysis (EDA) and Preparation for Model Training

Before performing analysis, we did data cleaning on our dataset. We identify and eliminate duplicate records from the dataset. Next, we check for missing values and understand their patterns.

The dataset is now ready for exploration, we performed:

- **Univariate Analysis:**
  We generated comprehensive statistics for each variable using to understand their distributions. Then, we performed normality testing and visualize it, provided visual insights to assess if numerical attributes followed a normal distribution.

- **Bivariate/Multivariate Analysis:**
  We calculated correlations between variables and visualize it to explore relationships.

Now, we had a basic understanding of the dataset. The next step was to prepare prerequisites for model training. We generated a train and test sets in 70:30 ratio using a predefined seed for reproducible. Based on the understanding of our dataset, we have identified it as a classification problem. Multiple algorithms have been selected for classification, and under each algorithm will have their own data pre-processing method that specifically tailor for their best performance. Since we lack understanding on how each algorithm performs, hence our goal is to test on each algorithm and select the best for our dataset.

4.1 Decision Tree

The methodology employed in the modified code involves an iterative process to comprehensively evaluate the performance of a decision tree model across various random partitions of the dataset. The initial step includes loading essential libraries and the dataset, laying the foundation for subsequent analyses. A dedicated function, predict_accuracy, is then defined to systematically calculate and display the confusion matrix for model predictions, enhancing interpretability and evaluation capabilities.

1) **Preparation:**
   - Loaded essential libraries
   - Load dataset from global
   - Setting seed for reproducible
   - Determine number of iterations
2) **Define Function:**
   - Define predict_accuracy to calculate model accuracy
   - Define class_tree for decision tree model

3) **Building Model iteratively:**
   - Generate partition for train and test dataset
   - Standardize numeric features
   - Convert categorical variables to factors
   - Classification Decision Tree
4) **Display Each Model Accuracy:**
   - Iterated through the modeling process five times for robustness.
   - Calculated and displayed accuracy for each pruned Decision Tree model.
   - Revealed variations in accuracy across iterations, providing insights into model stability.

4.2 K-Nearest Neighbors (kNN)

The kNN model is configured with 5 neighbors, and a dedicated function, "train_evaluate_knn," is employed to encapsulate the training and evaluation process. The script iteratively assesses the KNN model's accuracy over multiple iterations, initially with the original features and subsequently with standardized features. This systematic comparison allows for insights into the impact of feature standardization on classification accuracy.

1) **Preparation:**
   - Load nessecary libraries
   - Set seed for reproducibility
   - Load dataset from 'dataset.RData'
2) **Data Preprocessing:**
   - Checked for and removed duplicates using unique() and examined missing data patterns with md.pattern().
   - Created train and test sets through a 70/30 split, maintaining reproducibility with the set seed.
   - Numeric features were identified for standardization.
3) **Building Model:**
   - The variable k is set to 5, representing the number of neighbors in the KNN model.
   - The code defines a function train_evaluate_knn to train and evaluate the KNN model using the specified number of neighbors.
   - The function returns the accuracy of the model on the test data.
4) **Display Each Model Accuracy Accuracy:**
   - The KNN model is trained and evaluated on standardized features for the same number of iterations.
   - The average accuracy is calculated and printed for the models with standardized features.

4.3 Logistic Regression

Logistic Regression is a widely used statistical method for binary classification. It models the probability of an event occurring as a function of predictor variables.

1. **Preparation:**
    - Import necessary R libraries: mice, **tidyverse, dplyr, dlookr**, and **caret**.
    - Load the dataset from a pre-configured global environment using the "dataset.RData" file.
2. **Data Preprocessing:**
    - Utilize the **glm** function to build a Logistic Regression model **log_model** on the training dataset **train_df**.
    - Define a function, **predict_accuracy**, to make predictions on a test dataset and calculate the model's accuracy.
    - Convert the datasets **train_df, test_df, train_df_p, test_df_p** into matrix representations **train_features_m, test_features_m, train_features_pm, test_features_pm** suitable for modeling.
3. **Building Initial Model:**
    - Train the Logistic Regression model **log_model** on the training dataset **train_df**.
    - Evaluate the accuracy of the initial model using the **predict_accuracy** function on the test dataset **test_df**

4.4 Naive Bayes

Naive Bayes is a probabilistic classifier based on Bayes' theorem with the "naive" assumption of independence between features.

1. **Preparation:**
    - Import necessary R libraries: **mice**, **tidyverse**, **dplyr**, **dlookr**, **caret**, and **e1071** for Naive Bayes.
    - Load a pre-configured global environment containing the dataset.
2. **Data Preprocessing:**
    - Standardize the numeric features by scaling.
    - Convert categorical variables into factor variables.
    - Split the pre-processed dataset into training and testing sets.
3. **Building Initial Model:**
    - Construct an initial Naive Bayes model using the **naiveBayes** function.
    - Evaluate the accuracy of the initial model.

4.5 Random Forest

Random Forest is an ensemble learning method based on decision tree classifiers that operate by constructing multiple decision trees during training. It operates by aggregating predictions from a collection of decision trees, thereby reducing overfitting and enhancing predictive accuracy.

Steps taken in modelling:

1. **Preparation**
    a) Import necessary R libraries.
    b) Initialized parallel processing.
    c) Loaded a pre-configured global environment containing the dataset.
2. **Data Preprocessing**
    a) Standardization by scaling numeric features.
    b) Converted categorical variables in factor variables.
    c) Split the pre-processed dataset into training and testing.
3. **Building Initial Model**
    a) Constructed an initial model using “randomForest” library.
    b) Evaluated the accuracy of the initial model.
4. **Hyperparameter Tuning**
    a) Using “caret” library to perform hyperparameter tuning.
    b) Use repeated cross-validation and randomized parameter search to identify optimal hyperparameters for the model.
5. **Model Retraining with Best Parameters**
    a) Extracted the best parameters from the tuned model in last step and retrained a new model using “randomForest” library.
    b) Evaluated the accuracy of the final model.
6. **Model Accuracy Comparison**
    a) Created a bar plot comparing the accuracies of the initial model, tuned model, and final model to assess performance differences.

4.6 Support Vector Machines (SVM)

Support Vector Machines (SVM) is a machine learning algorithm used for classification and regression tasks. It works by finding the optimal decision boundary, or hyperplane, that best separates different classes in the dataset.

Steps taken in modelling using SVM is like Random Forest, the difference are:

1. **Preparation and Training**
    a. Replace Random Forest method with SVM method from “kernlab”.
2. **Data Preprocessing**
    a. Other than standardization, we add transformation to our dataset, we apply one-hot encoding on all categorical variables except out target.

4.7 XGBoost

XGBoost stands for eXtreme Gradient Boosting, a powerful and efficient gradient boosting algorithm widely used in machine learning competitions and various domains. It's an ensemble learning method that builds multiple decision trees sequentially, aiming to correct errors made by the preceding trees.

Steps taken in modelling using XGBoost is also like Random Forest, the difference are:

1. **Preparation and Training**
    a. Replace Random Forest method with method from “xgboost” library.
2. **Data Preprocessing**
    a. Instead of standardization and transformation, we use the dataset by converting it into XGBoost matrix format.

4.8 Rule Induction

Rule induction is a method in machine learning that creates simple "if-then" rules to classify data. These rules are easy to understand and provide insights into how decisions are made. This approach is useful in fields like medicine or law where understanding the reasoning behind predictions is important.

The following are the steps taken to use Rule Induction:

1. **Prepare Data:**
    a. Get your dataset with features and the target you want to predict.
    b. Make sure data is clean by handling missing values and outliers.
    c. Split your data into training and testing sets.
2. **Choose Rule Induction Algorithm:**
    a. Pick an algorithm like Apriori or Ripper that fits your data and problem.
    b. Set the algorithm parameters, such as how strict the rules should be.
3. **Train the Model:**
    a. Apply the algorithm to your training data to create rules.
    b. The rules are generated based on patterns found in the training data.
4. **Evaluate Rules:**
    a. Check if the rules make sense by testing them on the training set.
    b. Adjust rules or parameters to improve performance.
5. **Test the Model:**
    a. Use the rules on your testing data to see how well they predict.
    b. Measure performance using metrics like accuracy or precision.
6. **Understand Rules:**
    a. Look at the generated rules to understand how decisions are made.
    b. Visualize rules for better understanding.
7. **Fine-Tune (if needed):**
    a. Adjust parameters or features to improve the model.

b. Fine-tune until you're satisfied with performance.

4.9 Neural Net

Neural networks are a powerful class of machine learning models inspired by the human brain's structure and function. In classification tasks, neural networks learn complex patterns and relationships within data to make predictions. They consist of interconnected layers of nodes (neurons) that process and transform input features to produce an output, making them highly effective for tasks like image recognition, natural language processing, and more.

The following are the steps taken to use Neural Net:

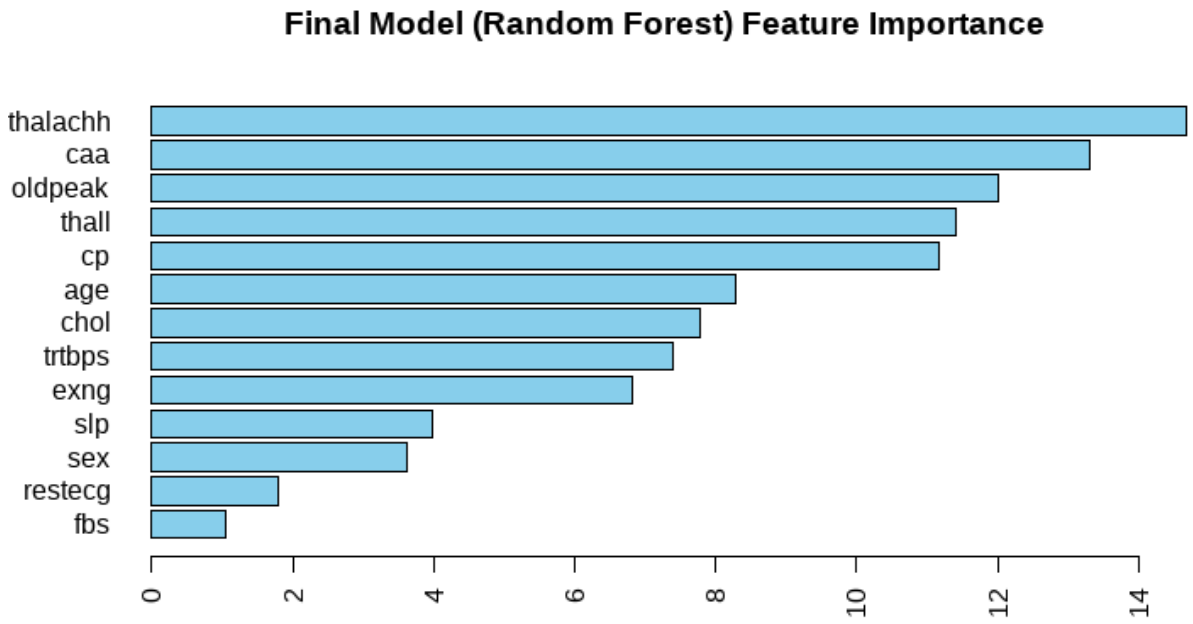
1. **Data Collection and Preprocessing:**
    a. Gather a dataset with labelled examples for training and testing.
    b. Preprocess data by handling missing values, scaling features, and encoding categorical variables.
2. **Split Data:**
    a. Divide the dataset into training and testing sets to assess the model's performance.
3. **Build the Neural Network:**
    a. Choose the architecture of the neural network, including the number of layers and neurons in each layer.
    b. Decide on the activation functions for each layer.
    c. Initialize the weights and biases of the network.
4. **Compile the Model:**
    a. Specify the loss function, optimizer, and evaluation metric for the model.
    b. Compile the neural network to prepare it for training.
5. **Train the Model:**
    a. Feed the training data into the neural network.
    b. Adjust the weights and biases during training to minimize the loss function.
    c. Monitor the training process for convergence.
6. **Evaluate Performance:**
    a. Use the trained model to make predictions on the testing set.
    b. Assess the model's performance using metrics like accuracy, precision, recall, and F1 score.
7. **Tune Hyperparameters (if needed):**
    a. Adjust hyperparameters such as learning rate, batch size, and the number of neurons to improve performance.
8. **Interpret Model Outputs:**
    a. Analyze the model's predictions and understand how it's making decisions.
    b. Visualize important features or learned representations.
9. **Fine-Tune and Iterate:**
    a. Based on the evaluation results, fine-tune the model or revisit earlier steps.
    b. Iteratively improve the model until satisfactory performance is achieved.

5.0 Results

We have decided to use Accuracy metric to evaluate the performance of each algorithm. It measures the proportion of correctly classified instances out of the total instances in a dataset, representing the model's overall correctness in its predictions.

Algorithm	Accuracy	Recall (Sensitivity)
Decision Tree	0.8000	-
K-Nearest Neighbors (kNN)	0.8333	-
Logistic Regression	0.8132	-
Naïve Bayes	0.8022	-
Random Forest	0.8667	0.9697
Support Vector Machine (SVM)	0.8667	0.9697
XGBoost	0.8667	0.9429
Rule Induction	0.8000	-
Neural Net	0.8000	-

From the table, we found that there are multiple algorithms reaching the same accuracy. So, we decided to use another metric call Recall or Sensitivity to evaluate for those algorithms. It is representing the actual positives that were correctly identified by a model. Again, there is two algorithms achieve the same recall value, we have decided to select the algorithm that is easier to model. We selected Random Forest as the final algorithm for our classification problem in heart attack prediction. As compared to SVM, it has less steps in data preparation and less parameter to tune with.



From the final model using Random Forest, we have extracted its feature importance, and plot it into a bar chart. Based on the chart, we discover that certain features significantly influence the model predictions. 'thalachh' (maximum heart rate achieved), 'oldpeak' (ST depression induced

by exercise relative to rest), 'caa' (number of major vessels), 'cp' (chest pain type), and 'thall' (thalassemia rate) appear to be the most impactful features in predicting the target. These findings suggest that factors related to heart rate, exercise-induced changes, chest pain type, and thalium stress test results hold considerable importance in predicting the target variable, potentially indicating critical markers for cardiovascular health or related conditions.

6.0 Individual Reflection

a) Wong Xing Hao
I had learnt multiple classification algorithms from this project, knowing how they works and implementing in R code, and learn about hypertuning parameters to reach a higher performance in model on the dataset using caret library. I wish I had known the algorithm's characteristic and its mechanism, it would make me much easier to start on with the modelling. Hence, I would advise to understand the algorithms before you start modelling, it would make your modelling process much faster and more efficient.

b) Loo Pei Xin
I have learnt about the most important steps in a heart attack prediction project, such as data cleaning, exploratory data analysis, and the application of several algorithms. I wish that I have an earlier familiarity with the characteristics of the dataset. With that, I will have a clearer understanding on the datasets and algorithms. To future students, I recommend focusing on an in-depth understanding of the dataset, exploring different algorithms, and iteratively refining model choices for optimal outcomes.

c) Cheng Pin-Jie
In the context of training model to predict heart attack, I immersed myself in the intricacies of Decision Trees and K-Nearest Neighbors (kNN) has been enlightening me. The iterative Decision Tree methodology, especially with functions like predict_accuracy, not only sharpened my ability to interpret model predictions but also provided crucial insights into enhancing the model's adaptability, a vital aspect in foreseeing heart attaack issues. The emphasis on reproducibility and consistency aligns with the critical nature of maintaining stable predictions in a health attack context. Exploring K-Nearest Neighbors further emphasized the importance of nuanced understanding, especially regarding feature standardization, in optimizing accuracy for heart attack prediction models. These insights not only enhance my technical skills but also reinforce the significance of a thorough understanding of algorithms in making precise and informed predictions.

d) Nur Adriana
As I was doing Rule Induction on the heart dataset, I noticed how important what value the support and the confidence is as I initialized it. The values depend on the relationship between the varaibles, a higher value is recommended but too high woud lead to inaccurate readings between the data, and too low would show no relationship between the crucial variables. This becomes apparent as well for the Neural Network method, as the architecture of the network can influence the result of the reading.