**School of Computing**

**6006CEM Machine Learning and Related Applications**

**Assignment Brief August 2023**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Module Title  Machine Learning and Related Applications | Individual | | Cohort - AUG | Module Code  **6006CEM** |
| Coursework Title (e.g. CWK1) Report  Portfolio | | | | Hand out date:  21/08/2023 |
| Lecturers  Vasuky Mohanan | | | | Due date:  **24/11/2023** |
|  | | Coursework type  Practical work | | % of Module Mark  100% |

Module Learning Outcomes Assessed:

1. Apply the knowledge behind the principles, techniques and applications of machine learning
2. Critically evaluate existing machine learning methods and select the most appropriate ones for a certain task
3. Analyse information, compare different machine learning techniques and produce an academic written report as a result
4. Conceptualise the role of modern machine learning approaches and their impact on society

Table of Contents

[Table of Figures 3](#_Toc151676779)

[1.0 Introduction 4](#_Toc151676780)

[2.0 Problem Statement 4](#_Toc151676781)

[3.0 Existing Approaches or Methods and Their Results 4](#_Toc151676782)

[3.1 Logistic Regression 4](#_Toc151676783)

[3.2 Similarities and Differences Between Our Work and Existing Approaches 5](#_Toc151676784)

[4.0 Implementation 6](#_Toc151676785)

[4.1 Data Analysis 6](#_Toc151676786)

[4.2 Data Preprocessing 7](#_Toc151676787)

[4.3 Applied Machine Learning 10](#_Toc151676788)

[4.4 Model Tuning 12](#_Toc151676789)

[5.0 Result 12](#_Toc151676790)

[5.1 Analysis and Evaluation 12](#_Toc151676791)

[5.2 Conclusion 16](#_Toc151676792)

[6.0 Appendix 17](#_Toc151676793)

# **Table of Figures**

[**Figure 1**: Existing Logistic Regression 4](#_Toc151676696)

[**Figure 2:** Dataset Attributes 6](#_Toc151676697)

[**Figure 3:** Data Pre-processing (Duplicates) 7](#_Toc151676698)

[**Figure 4:** Data Pre-processing (Outliers) 8](#_Toc151676699)

[**Figure 5**: BoxPlot for Outliers 8](#_Toc151676700)

[**Figure 6**: Data Pre-processing (StandardScaler) 9](#_Toc151676701)

[**Figure 7**: Logistic Regression Code Snippet 10](#_Toc151676702)

[**Figure 8**: Decision Tree Code Snippet 12](#_Toc151676703)

[**Figure 9**: Logistic Regression Result 13](#_Toc151676704)

[**Figure 10**: Confusion Matrix (Logistic Regression) 14](#_Toc151676705)

[**Figure 11**: Decision Tree Result 15](#_Toc151676706)

[**Figure 12**: Confusion Matrix (Decision Tree) 15](#_Toc151676707)

# **1.0 Introduction**

Heart disease and other cardiovascular illnesses continue to be major global health concerns. Predicting heart disease accurately and promptly is essential for better patient outcomes and early intervention. This report explores current strategies for predicting heart disease and introduces a novel approach, emphasizing the advantages and disadvantages of our work relative to the existing approaches.

# **2.0 Problem Statement**

Globally, heart disease is one of the main causes of morbidity and death. It is critical to promptly identify those who are at risk to put preventive measures in place. Even though they work well, current prediction models might not be robust enough for accurate risk assessment. Our work aims to improve heart disease prediction models' accuracy and investigates new areas for improvement.

# **3.0 Existing Approaches or Methods and Their Results**

## **3.1 Logistic Regression**

A screenshot of a computer program

Description automatically generated

**Figure 1**: Existing Logistic Regression

The provided code aims to predict the presence or absence of heart disease based on 13 attributes by training and evaluating a logistic regression model using the 'heart.csv' dataset. Out of the dataset, the target variable ‘target’ is extracted as ‘y’ and the features are represented by ‘x’ with the target variable excluded. The ‘train\_test\_split’ function is then used to divide the data into training and testing sets with a split ratio of 70–30. ‘LogisticRegression()’ from scikit-learn is used to instantiate the logistic regression model, which is then fitted to the training data ‘x\_train and y\_train’ and tested on the testing set. Calculated and kept in the 'accuracies' dictionary is the model's accuracy on the test set. The model's performance in accurately classifying cases of heart disease is indicated by the printed final accuracy. This code offers a simple method for evaluating the model's accuracy on hypothetical data, serving as a foundational example of training, and testing a logistic regression model for heart disease prediction. The accuracy for this existing logistic regression model is 87.01%.

## **3.2 Similarities and Differences Between Our Work and Existing Approaches**

**Similarities**

There are significant structural and overall goal similarities between the modified and current logistic regression models. The models utilize identical 13 attributes to forecast the probability of heart disease, indicating a common goal in the analysis of cardiovascular health. To emphasize the standard method of preparing data for machine learning tasks, the first step entails splitting the dataset into features (X) and the target variable (y). Furthermore, both implementations use the scikit-learn LogisticRegression() class, highlighting the fact that logistic regression is the algorithm of choice for heart disease prediction in both cases. Moreover, a critical similarity is that train-test splitting is used to assess the model's performance on unknown data, which is consistent with machine learning best practices.

**Differences**

On the other hand, differences between the two implementations' specific details become apparent. Notably, the modified model modifies the train-test split ratio to 80-20 from the existing model's 70-30. The split ratio variance may have an effect on the model's capacity to generalize to new data. The random state parameter, which affects the results' reproducibility, represents yet another notable distinction. In addition, the updated model adds a StandardScaler() standardization In conclusion, although having the same goal and algorithm, the two models differ in subtle ways that may have an impact on the results of the models. These differences include split ratio variations, random states, and the inclusion of feature standardization step to guarantee consistent feature scales—an important logistic regression consideration. The current model lacks this standardization process, which could have an impact on the convergence and general performance of the model.

# **4.0 Implementation**

## **4.1 Data Analysis**

A white text with black text

Description automatically generated

**Figure 2:** Dataset Attributes

13 health-related attributes make up the dataset, which attempts to predict whether a person will have heart disease. Age, sex, clinical indicators, maximum heart rate reached, electrocardiogram results, fasting blood sugar, maximum heart rate, exercise-induced angina, exercise-induced ST depression, slope of peak exercise ST segment, number of major vessels colored by fluoroscopy, and a categorical variable indicating a fixed or reversible defect in the thalassemia region are just a few examples of the attributes that fall under this category. The binary result, denoting the existence (1) or absence (0) of heart disease, is represented by the 'target' variable. The abundance of these characteristics offers a thorough foundation for teaching a machine learning model to identify trends and connections, which helps create a predictive model for heart disease that works well.

## **4.2 Data Preprocessing**

A computer code with text

Description automatically generated with medium confidence

**Figure 3:** Data Pre-processing (Duplicates)

This little piece of code looks for duplicate rows in the DataFrame ‘df’ and eliminates them. The ‘duplicated()’ function is first used to find duplicate rows in the dataset. The ‘duplicates’ DataFrame contains the duplicates that are produced. The conditional statement that follows determines whether ‘duplicates’, which indicates the existence of duplicate rows, is empty. If duplicates are discovered, they are eliminated using the ‘drop\_duplicates()’ function, and the updated DataFrame is then assigned to the variable ‘df’. Essentially, by removing any unnecessary rows from the dataset, this code protects its integrity and improves the precision and dependability of later analyses and the training of machine learning models using the data it provides.

A screen shot of a computer code

Description automatically generated

**Figure 4:** Data Pre-processing (Outliers)

A screenshot of a graph

Description automatically generated

**Figure 5**: BoxPlot for Outliers

This snippet of code uses box plots and further data manipulation to find and fix outliers in the dataset. To visualize the distribution and identify possible outliers in the numerical columns of the DataFrame ‘df’, a box plot is first made using Seaborn. Visibility is improved by the large figure sizes (15, 10). Next, the code looks at the box plot and finds outliers in the following columns: ‘age’, ‘trestbps’, ‘chol’, ‘thalach’, ‘oldpeak’. The interquartile range (IQR) is determined for every column after the first and third quartiles (Q1 and Q3) have been computed. Outliers are identified and deleted from the DataFrame when their values fall outside of the 1.5 times IQR range from the quartiles. Based on the insights gleaned from the box plot, this data cleansing step ensures a more representative and robust dataset for further analyses and model training, thereby reducing the possible influence of extreme values on statistical measures and machine learning algorithms.

A screen shot of a computer code

Description automatically generated

**Figure 6**: Data Pre-processing (StandardScaler)

This code snippet standardizes the features of the dataset using the ‘StandardScaler’ from scikit-learn. Standardization is a crucial preprocessing step in machine learning, ensuring that numerical features are on a consistent scale. The ‘StandardScaler’ transforms the data by centering it around the mean and scaling it based on the standard deviation. Specifically, for each feature, the mean is subtracted, and the result is divided by the standard deviation. This process is essential when working with algorithms, such as logistic regression, that are sensitive to the scale of input features. By standardizing the features, the model becomes less influenced by the varying magnitudes of the original data, enhancing its stability, convergence, and overall predictive performance. In this context, the standardized features ‘X\_train’ and ‘X\_test’ are ready for input into the logistic regression model, ensuring a more robust and reliable training process.

## **4.3 Applied Machine Learning**

**Logistic Regression**

A computer code with text

Description automatically generated



**Figure 7**: Logistic Regression Code Snippet

Logistic regression section is to create and assess a heart disease prediction model using the characteristics found in the 'heart.csv' dataset. First, preprocessing steps are performed on the data, such as utilizing box plots to identify possible outliers and verifying if there are duplicates. Next, the dataset is divided into features (X) and the target variable (y), which denotes the presence or absence of heart disease in an individual. Using the `train\_test\_split’ function, the data is further split into training and testing sets in accordance with standard machine learning procedure. The data is standardized using the ‘StandardScaler’ in order to guarantee standardized features and lessen the impact of different scales. Next, ‘LogisticRegression()’ from scikit-learn is used to instantiate a logistic regression model. Using the `fit` method, the model is trained on the standardized training data ‘X\_train’, ‘y\_train’. The target variable on the standardized test set ‘X\_test’ is then predicted by the trained model, producing ‘logreg\_predictions’. Metrics like accuracy, confusion matrix, and classification report are used to assess the performance of the logistic regression model and provide information about how well the model can classify cases of heart disease. A heatmap is used to visually represent the confusion matrix, which makes it easier to interpret predictions that are false positive, false negative, true positive, and true negative. All things considered; this logistic regression analysis is an essential first step in evaluating how well the model predicts heart disease using the given dataset.

**Decision Trees**

A computer code with text

Description automatically generated



A black background with red text

Description automatically generated

**Figure 8**: Decision Tree Code Snippet

The goal of the provided code's decision tree section is to build and evaluate a heart disease prediction model using a decision tree classifier. The code starts with preprocessing the data, looking for duplicates, and using a box plot visualization to find possible outliers. The dataset is then divided into features (X) and the target variable (y), which represents the presence or absence of heart disease, after these checks. Using the ‘train\_test\_split’ function, the dataset is further split into training and testing sets to aid in model evaluation. Using the ‘StandardScaler’, features are standardized to make sure that variables are on a similar scale, which is an important step for decision tree models. Next, using ‘DecisionTreeClassifier()’ from scikit-learn, the Decision Tree classifier is instantiated, and the model is trained using the `fit` method on the standardized training data ‘X\_train’, ‘y\_train’. The target variable on the standardized test set ‘X\_test’ is then predicted by the trained Decision Tree model, yielding ‘dt\_predictions’. To evaluate how well the model performs in correctly classifying cases of heart disease, metrics for model evaluation are computed, such as accuracy, confusion matrix, and classification report. To help with the interpretation of true positive, true negative, false positive, and false negative predictions, the confusion matrix is shown graphically as a heatmap. In conclusion, using the dataset that was supplied, this Decision Tree analysis is essential for determining how well the model predicts heart disease.

## **4.4 Model Tuning**

# **5.0 Result**

## **5.1 Analysis and Evaluation**

A screenshot of a computer

Description automatically generated

**Figure 9**: Logistic Regression Result

A screenshot of a graph

Description automatically generated

**Figure 10**: Confusion Matrix (Logistic Regression)

The performance of the logistic regression model is comprehensively assessed by the evaluation metrics. The percentage of correctly predicted cases is 85.96%, which indicates a relatively high overall accuracy in categorizing people as having or not having heart disease. The model's ability to discriminate between the two classes is illustrated by the confusion matrix, which shows 21 true negatives, 28 true positives, 5 false positives, and 3 false negatives. While recall (0.90) highlights the model's ability to capture all real positive instances, precision (0.85) shows how accurate positive predictions are. By balancing recall and precision, the F1-score of 0.88 offers a reliable indicator of the model's overall efficiency. Lastly, the support values provide a more thorough understanding of the dataset's distribution by indicating the quantity of instances in each class. Overall, the balanced precision, recall, and F1-score metrics point to a robust logistic regression model that does a good job of correctly identifying people who have heart disease.

**Decision Tree**

A screenshot of a computer

Description automatically generated

**Figure 11**: Decision Tree Result

A screenshot of a graph

Description automatically generated

**Figure 12**: Confusion Matrix (Decision Tree)

Decision Tree model's performance is given by the evaluation metrics. According to the reported accuracy of 73.68%, which represents the percentage of correctly predicted instances, there is a moderate degree of overall accuracy in the classification of individuals as having or not having heart disease. The model's ability to discriminate between the two classes is demonstrated by the confusion matrix, which shows 18 true negatives, 24 true positives, 8 false positives, and 7 false negatives. Recall (0.77) highlights the ability of the model to capture all real positive instances, whereas precision (0.75) indicates the accuracy of positive predictions. A reliable indicator of the model's overall efficacy, the F1-score (0.76) finds a balance between recall and precision. The support values provide information about the distribution of the dataset by showing the number of instances in each class. In comparison to the logistic regression model, the decision tree model performs reasonably, but its metrics point to a slightly lower accuracy and a more complex trade-off between precision and recall.

## **5.2 Conclusion**

# **6.0 Appendix**

**GitHub URL**: <https://github.com/DamienTan01/6006CEM-Machine-Learning-DamienTanLekKhee>

**Turnitin**  
A screenshot of a cellphone

Description automatically generated