Machine Learning Model for Predicting Heart Disease

Ozair Khan, Zaina Basit Alavi

04/06/2023

Abstract

Heart failure is a severe and life-threatening illness that is responsible for the majority of deaths worldwide. According to WHO, approximately 17.9 million people die from heart disease each year, with patients often unaware of their health status, making them vulnerable to various cardiovascular diseases. Heart disease patients have a high mortality rate, with 55% of them dying within the first three years of diagnosis. Treatment is also quite expensive, accounting for about 4% of the annual healthcare system cost. A machine learning algorithm would help address these challenges by automating the diagnosis process resulting in a reduction in costs, more accurate diagnoses and an improvement in treatment outcomes. The following project will handle this problem in the form of supervised binary classification and will include the classes yes and no (in regards to the diagnosis of heart disease). Machine learning models including Logistic Regression and K-Nearest-Neighbors will be utilized for the purpose of finding the best possible classifier for heart disease.

Introduction

The human heart is arguably one of the most important organs and responsible for controlling the flow of blood throughout the human body. Any issues with the heart or cardiovascular system may result in sudden death by heart failure. Multiple factors such as smoking, an unhealthy lifestyle and alcohol can play a role in the onset of heart diseases or heart failure. The objective of this project is to use two machine learning models, namely Logistic Regression and K-Nearest-Neighbors to train a supervised classification model that is able to answer whether a person has heart disease or not. Both models will be compared in terms of accuracy, with the better model being selected as the classifier for heart disease.

Project Description

Data Collection

The data set dates from 1988 and was retrieved from four databases: Cleveland, Hungary, Switzerland, and Long Beach V. The dataset has a total of 76 attributes but referring to all published experiments suggests using a subset of 14 features. The target field indicates whether the patient has cardiac disease. It has a value of 0 (no presence) to 1 (present). Since it is a public dataset the source used is https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset. The following table indicates the attributes (age,sex..etc) and the description and range to detect heart disease:

Attributes	**Description**
age	Age in years
sex	[1=male;0=female]
chest_pain_type	Chest pain type(Value 1: typical angina, Value 2: atypical angina, Value 3: non-anginal pain, Value 4: asymptomatic)
blood_pressure_resting	resting blood pressure (mm Hg on admission to the hospital)
cholesterol	cholesterol measurement in mg/dl
blood_sugar_fasting	(Fasting blood sugar > 120 mg/dl)(1=true;0=false)
electrocardiogram	Resting electrocardiographic measurement (0 = normal, 1 = having ST-T wave abnormality, 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria)
max_heart_rate	Maximum heart rate achieved
angina	Exercise induced angina(1=yes;0=no)

s_trajectory	ST depression induced by exercise relative to rest
slope	slope of the peak exercise ST segment
blood_vessels	Number of major vessels (0-3) by fluoroscopy
thalassemia	0 = normal; 1 = fixed defect; 2 = reversible defect
target	1[exists] or 0[does not]

Exploratory Data Analysis

Exploratory Data Analysis refers to the crucial process of doing preliminary investigations on data to uncover patterns, detect anomalies, test hypotheses, and validate assumptions using summary statistics and graphical representations. It is best practice to first analyze the data and then strive to glean as many insights as possible from it.

We intended to investigate the distribution tally of heart detection, as shown in figure 1. In the dataset, the detected heart disease exceeded undetected heart disease with 0 indicating no heart disease and 1 indicating the presence of heart disease. Additionally, univariate outlier is a data point that consists of an extreme value on one variable. The graphs in figure 1 comparing heart diseases versus non heart diseases represent that univariate outlier is a case with an extreme value that exceeds the projected population values for a single variable and so differs from the majority of cases seen in the normal distribution of that variable. Figure 2 aims to comprehend the identification of heart disease dependent on gender. It was fascinating to note that the female count had more detected cases than the male count. The male graph predicts that the males suffer less from heart diseases while females suffer more. Furthermore, a multivariate outlier is a combination of unusual scores on at least two variables. Both types of outliers female vs male

heart diseases count can influence the outcome of statistical analyses. Outliers exist in multiple dimensions. Incorrect data entry can cause data to contain extreme cases.Figure 3 depicts a correlation matrix between all the dataset attributes. Correlation is a measure of the relationship between two variables. The correlation matrix has values ranging from -1 to +1. Those near to -1 are considered negative correlations, whereas values close to +1 are considered positive correlations. Based on the correlation plot the strongest correlations exist between attributes including target, chest_pain_type, max_heart_rate and angina, indicating these variables are highly correlated with heart disease.

The IQR (Interquartile Range) is the difference between a distribution's third and first quartiles (or the 75th percentile minus the 25th percentile). Because this range comprises half of the points in the dataset, it represents a measure of how broad our distribution is. It's quite helpful to get a sense of the distribution's form. It's the width of the boxes in a box plot. Hence in Figure 4 , using the IQR function found the lower and upper bound values including the found value representing the outliers as well.

We can use IQR to find outliers once we've calculated it. If a point meets one of the following criteria, it is considered an outlier. It exceeds the 75th percentile plus 1.5 IQR. It falls below the 25th percentile (-1.5 IQR). Simply discover the outliers in our distribution using this simple algorithm. Outliers are plotted as points outside the whiskers in Boxplot using the same manner. The concept is that a point that is too distant from the 75th percentile (or the 25th percentile) is an "abnormal" point that might be characterized as an outlier. The IQR is the order of magnitude of such a distance.

Figure 1

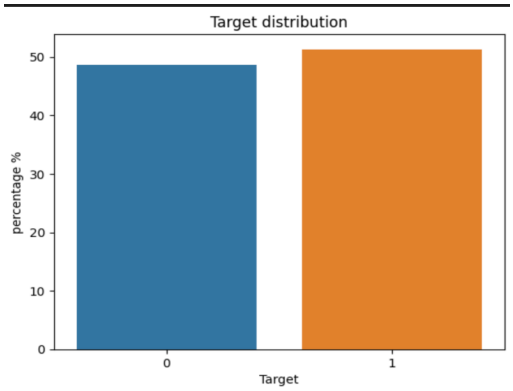


Figure 2

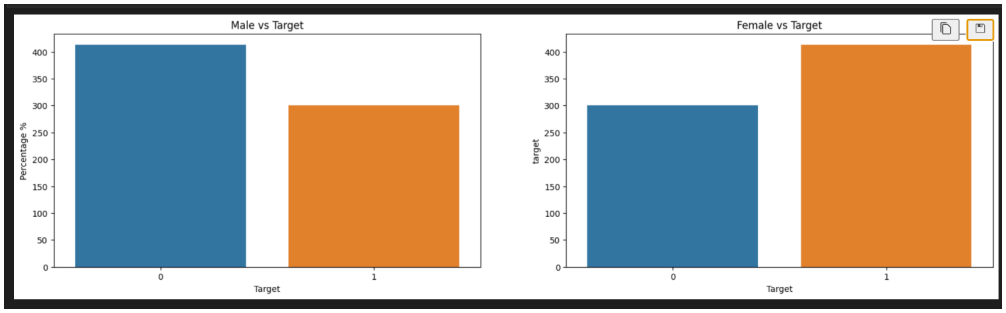


Figure 3

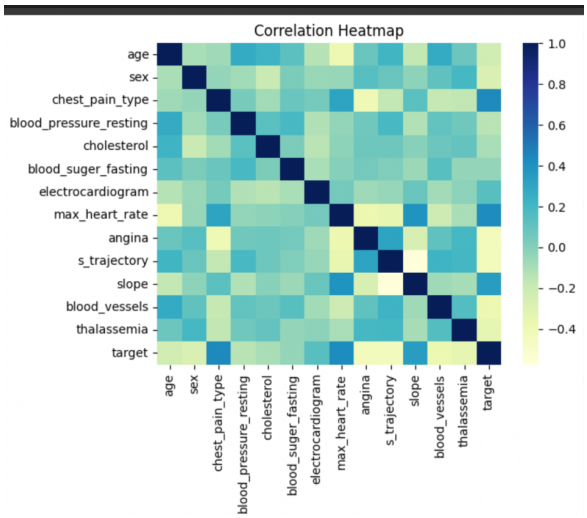


Figure 4:

```
import numpy as np

def iqrOutliers(df, column):
  global lower, upper
  q25, q75 = np.quantile(df[column],0.25), np.quantile(df[column],0.75)
  iqr = q75 - q25
  cut_off = iqr * 1.5
  lower, upper = q25 - cut_off , q75 + cut_off
  print('the found value of iqr is',iqr)
  print('the lower bound value is', lower)
  print('the upper bound value is', upper)
  df1 = df[df[column] > upper]
  df2 = df[df[column] < lower]
  return print('total number of outliers are', df1.shape[0]+df2.shape[0])

iqrOutliers(df,'cholesterol')
```

```
the found value of iqr is 64.0
the lower bound value is 115.0
the upper bound value is 371.0
total number of outliers are 16
```

Feature Selection and Extraction

This project utilizes feature extraction for simplifying the dataset into a new set of features that capture the most important information. This is done with the use of principal component analysis which reduces the dimensions of the dataset into new features known as principal components which are linear combinations of the original features. By reducing the dimensions and subsequently the noise and redundancy of the dataset, the model is able to prevent overfitting. This is demonstrated in figure 5, where the dataset features are transformed by PCA() and then used by train_test_split() to create the training and testing sets.

Figure 5

```
def split_datset_LR(self, t_size: float, r_state: int):
    x = self.dataset.drop('target', axis=1)
    y = self.dataset['target']

    scaler = StandardScaler()
    x = scaler.fit_transform(x)
    pca = PCA(n_components=5)
    pca_features = pca.fit_transform(x)
    print(f'PCA Features: \n {pca_features}')

    x_train, x_test, y_train, y_test = train_test_split(pca_features, y, test_size=t_size, random_state=r_state)

    return x_train, x_test, y_train, y_test
```

Methodology

The implementation of both the logistic regression and k-nearest-neighbor models went as follows. The dataset class was used to initialize a dataset object with methods for preprocessing and splitting the dataset (based on model type). A machine learning model could then be initialized as an object which accepts a dataset object as an initialization parameter. The model classes KNN and LR include methods for fitting, predicting and returning scores for the associated model.

The KNN (K-Nearest Neighbors) model class was implemented and trained using the "KNeighborsClassifier" class from the "sklearn.neighbors" module. The model's training and testing sets are created using the dataset's splitting method which includes parameters for split ratio (80/20) and the number of random states being 42. The fit method was implemented with an option to tune hyperparameters using GridSearchCV as displayed in figure 6. If hyperparameter tuning is selected, a parameter grid is defined with options for "n_neighbors", "weights", and "algorithm". The model is then trained on all combinations of hyperparameters in the grid, and the best performing model is chosen and set as the model for the class instance. If hyperparameter tuning is not selected, the model is fit on the training data using the default hyperparameters. The predict method is used to generate predictions for the testing data, and the score method calculates and returns the accuracy, classification report, and confusion matrix for the predictions.

Figure 6

```python
# Normal Fitting and Hyper Parameter fitting
def fit(self, tune_fit: str):
    # Tuning with Hyper Parameters
    if tune_fit == "yes":
        param_grid = {
            'n_neighbors': [3, 5, 10, 15, 20],
            'weights': ['uniform', 'distance'],
            'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']
        }
        grid_search = GridSearchCV(self.model, param_grid, cv = 5) # 5 folds
        grid_search.fit(self.x_train, self.y_train) # training the datasets on all the different hyper params
        self.best_params = grid_search.best_params_
        self.model = grid_search.best_estimator_ # setting model to best predictor
        self.best_score = grid_search.best_score_

        print(f"Best Param: {grid_search.best_params_}, Best Model: {grid_search.best_estimator_ }\n")

    else:
        self.model.fit(self.x_train, self.y_train)

    return
```

The LR (Logistic Regression) model was implemented and trained using the "LogisticRegression" class from the "sklearn.linear_model" module. The training and testing sets are created with the same parameters as KNN, with the addition of PCA for feature extraction. The fit method was implemented to fit the model on the training data using the default hyperparameters. The predict method is used to generate predictions for the testing data, and the score method calculates and returns the accuracy, classification report, and confusion matrix for the predictions.

Results/Experimental Data Analysis:

A classification report is used to assess the accuracy of a classification algorithm's predictions. On a per-class level, the report displays the primary classification metrics accuracy, recall, and f1-score. Two model algorithms were used, K-Nearest Neighbour(KNN) and Logistic Regression(LR) to test the heart disease dataset. KNN had a higher accuracy of about 97% hence proving it to be a better model than Logistic Regression. Logistic regression accuracy was about 80.48%. The KNN model achieved a precision of 0.95 for the positive class and 0.99 for the

negative class. The recall for the positive class was 0.99, while for the negative class, it was 0.95. The LR model, on the other hand, had a precision of 0.77 for the positive class and 0.85 for the negative class. The recall for the positive class was 0.87, while for the negative class, it was 0.74.

Furthermore, the confusion matrix is a table that is often used to describe the performance of a classification model. It shows the number of correct and incorrect predictions made by the classification model compared to the actual outcomes in the dataset.

In the results, the confusion matrix shows the following:

[[97 5]

[1 102]]

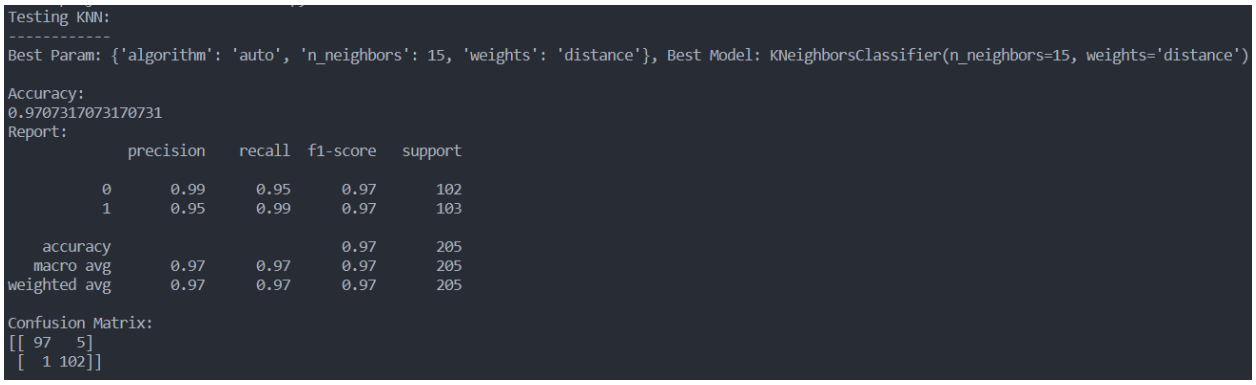
This means that there were 97 true negatives (TN), 5 false positives (FP), 1 false negative (FN), and 102 true positives (TP) in the test set for the KNN model.

Similarly, for the logistic regression (LR) model:

[[75 27]

[13 90]]

This means that there were 75 true negatives (TN), 27 false positives (FP), 13 false negatives (FN), and 90 true positives (TP) in the test set for the LR model. Confusion matrix was used in order to test both models since it is a powerful tool for assessing classification models



```
Testing LR:
-----------
PCA Features:
 [[-0.52255555 -1.11280319  0.95681551 -1.14919793 -0.55925188]
 [ 2.59038087 -0.53316168  1.46731492  1.53661364  1.34533474]
 [ 3.04235195 -1.32752064 -0.42476481  1.56720359  0.28381352]
 ...
 [ 1.24507316 -1.45735643 -0.4738734  -0.64524022 -0.27119664]
 [-1.62005298  0.12444348 -1.32795605 -1.19680377 -0.22491296]
 [ 0.93416859 -1.77854872 -0.00588162  0.35337228 -0.7433807 ]]
Accuracy:
0.8048780487804879
Report:
              precision    recall  f1-score   support

           0       0.85      0.74      0.79       102
           1       0.77      0.87      0.82       103

    accuracy                           0.80       205
   macro avg       0.81      0.80      0.80       205
weighted avg       0.81      0.80      0.80       205

Confusion Matrix:
[[75 27]
 [13 90]]
```

Conclusion

The above analysis shows that K-Nearest-Neighbor is the better classifier for diagnosing heart disease. It is important to note that KNN should be used with hyper parameter tuning to achieve the best possible accuracy. In the future this project could be expanded to a web application for predicting heart disease on live patient data.

References

American Heart Association. (n.d.). Understand Your Risks to Prevent a Heart Attack. Retrieved March 29, 2023, from https://www.heart.org/en/health-topics/heart-attack/understand-your-risks-to-prevent-a-heart-attack

Mayo Clinic. (2022). Heart disease diagnosis. https://www.mayoclinic.org/diseases-conditions/heart-disease/diagnosis-treatment/drc-20353124

National Heart, Lung, and Blood Institute. (2021). How is heart disease diagnosed? https://www.nhlbi.nih.gov/health-topics/heart-disease-diagnosis

John Smith. (2021). Heart Disease Dataset. Kaggle. https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset