COMPARATIVE STUDY ON MACHINE LEARNING ALGORITHMS FOR HEART DISEASE PREDICTION

Sanskar Aggarwal
E-Mail Id: sanskaraggarwal2001@gmail.com
J.C. Bose University of Science & Technology, YMCA, Faridabad, Haryana, India

Abstract- Heart disease is one of the most critical human diseases in the world and affects human life to a very large extent. An accurate and timely diagnosis of heart disease is important to treat and prevent a heart failure. Using machine learning techniques and the data procured by the health care industry, a disease can be detected, predicted and even cured. In this paper, the Naive Bayes, Linear Classifier, K-nearest Neighbour and Random Forest machine learning algorithms have been applied. The results of these four algorithms were compared on the basis of accuracy, specificity and sensitivity for prediction of disease.

Keywords: Machine Learning, Heart Disease, K-nearest neighbour, Naïve Bayes, Random Forest

1. INTRODUCTION

The dataset is available at https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data. The data has been obtained from the Cleveland region and is segregated into 14 columns. After loading the data, we observe that there are 303 rows and 14 columns. The description of each column is given below:

- age : age in years
- sex : sex (1 = male; 0 = female)
- cp : chest pain type
  - Value 1: typical angina
  - Value 2: atypical angina
  - Value 3: non-anginal pain
- trestbps : resting blood pressure (in mm Hg on admission to the hospital)
- chol : serum cholestoral in mg/dl
- fbs : (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
- restecg : resting electrocardiographic results
- thalach : maximum heart rate achieved
- exang : exercise induced angina (1 = yes; 0 = no)
- oldpeak : ST depression induced by exercise relative to rest
- slope : the slope of the peak exercise ST segment
- thal : 3 = normal; 6 = fixed defect; 7 = reversable defect
- num(the predicted attribute) : diagnosis of heart disease (angiographic disease status)
  - Value 0: < 50% diameter narrowing
  - Value 1: > 50% diameter narrowing

2. DATA EXPLORATION

The "num" variable predicts whether a person has a heart disease or not. We can plot the graph based on two factors i.e., disease and healthy using the following code:

```
heart$num<-ifelse(heart$num > 0,"Disease","Healthy") table(heart$num)
ggplot(heart,aes(x = num)) + geom_bar(fill="blue")
```

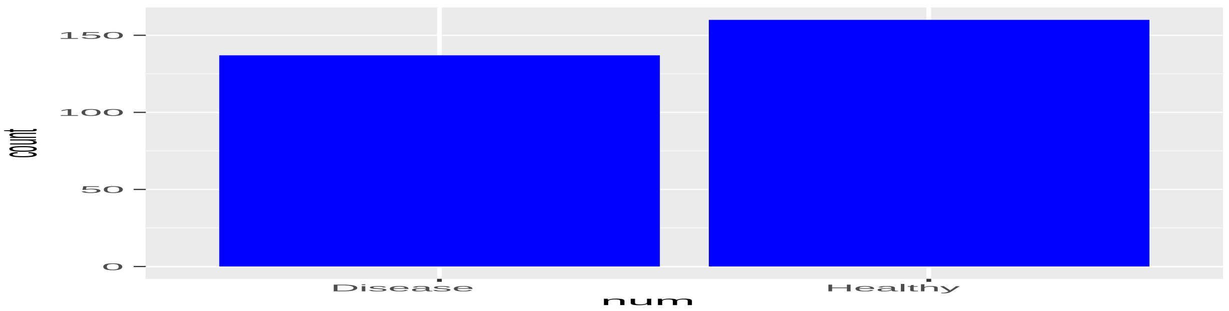

Fig. 2.1 Disease and Healthy Count

Out of the 297 observations, 137 people have been diagnosed with the heart disease. The sex of a person is represented as 1 for Male and 0 for Female. We will now replace that by strings “Male” and “Female” for better analysis.

```
heart$sex<-ifelse(heart$sex > 0,"Male","Female")
table(heart$num, heart$sex)
ggplot(heart,aes(x=sex)) + geom_bar(fill="blue") + facet_wrap(~num)
```

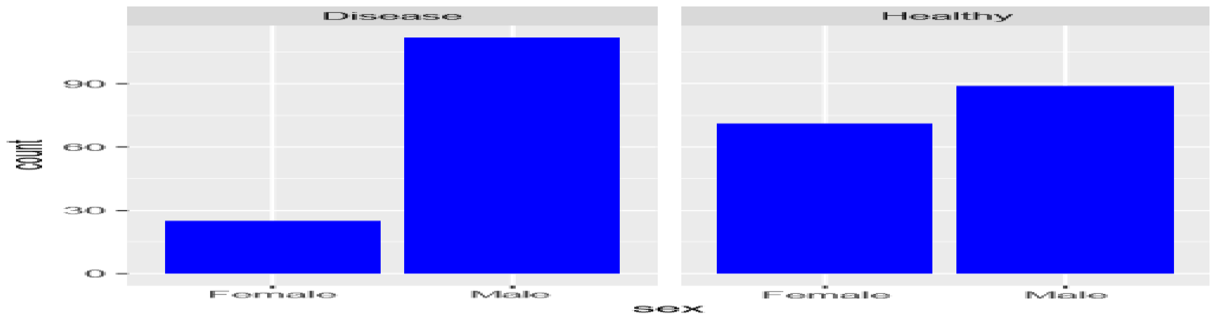

Fig. 2.2 Diagnosed Disease and Healthy Male-Female Count

It is evident from the above graph that males have a higher probability of suffering from a heart disease as compared to females.

3. ALGORITHMS

The algorithms are compared on three parameters:

- Accuracy: It is defined as the number of correct predictions from all the predictions made.
- Sensitivity: It is defined as the ability of an algorithm to predict a positive outcome when the actual outcome is positive. In our case, it defines the proportion of people who were correctly identified as “Healthy” when they were actually not diseased.
- Specificity: It is defined as the ability of an algorithm to not predict a positive when the actual outcome is not positive. In our model, specificity defines the proportion of people who were identified as “Diseased” when they were actually diseased.

```
set.seed(1, sample.kind="Rounding")
heart$num<-as.factor(heart$num)
levels(heart$num) <-c("Healthy","Disease")
```

We are dividing the dataset into training and test sets. The complete dataset has been divided into a ratio of 70:30 for training and test set respectively.

```
partition <- createDataPartition(heart$num,p=0.7,list=FALSE)
train<- heart[partition,]
test<-heart[-partition,]
```

The results are stored in a data frame which is used for comparision among all models.

```
results <- data.frame(Model = character(),
Accuracy = double(),
Sensitivity = double(),
Specificity = double(), stringsAsFactors = FALSE)
```

4. MODELING FOR PREDICTION

4.1 Naive Bayes

It is one of the most basic models used for prediction.

```
nb_model = train(num ~ ., data = train, method = "nb") predictions
= predict(nb_model, newdata = test) confusionMatrix <-
confusionMatrix(predictions, test$num)
results[nrow(results) + 1, ] <- c(as.character('Naive Bayes (nb)'),

                                          confusionMatrix$overall['Accuracy'],
                                          confusionMatrix$byClass['Sensitivity'],
                                          confusionMatrix$byClass['Specificity'])
rm(nb_model, predictions)
confusionMatrix
## Confusion Matrix and Statistics
##
##          Reference
## Prediction Healthy Disease
##   Healthy     34      7
##   Disease      7      41
##
##             Accuracy : 0.8427
##               95% CI : (0.7502, 0.9112)
##      No Information Rate : 0.5393
##     P-Value [Acc > NIR] : 1.452e-09
##
##                Kappa : 0.6834
##
##   Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8293
##             Specificity : 0.8542
##          Pos Pred Value : 0.8293
##          Neg Pred Value : 0.8542
##             Prevalence : 0.4607
##           Detection Rate : 0.3820
##      Detection Prevalence : 0.4607
##       Balanced Accuracy : 0.8417
##
##          'Positive' Class : Healthy
##
```

4.2 Linear Classifier

It makes a classification decision based on the value of a linear combination of the characteristics. A (generalized) linear model is fit using a boosting algorithm based on component-wise univariate linear model to make the predictions.

```
lr_model = train(num ~ ., data = train, method = "glmboost")
predictions = predict(lr_model, newdata = test) confusionMatrix <-
confusionMatrix(predictions, test$num)
results[nrow(results) + 1, ] <- c(as.character('Linear Classifier (glmboost)'),

                                   confusionMatrix$overall['Accuracy'],
                                   confusionMatrix$byClass['Sensitivity'],
                                   confusionMatrix$byClass['Specificity'])
rm(lr_model, predictions)
confusionMatrix
## Confusion Matrix and Statistics
##
##          Reference
## Prediction Healthy Disease
##   Healthy      33      8
##   Disease       8     40
##
##             Accuracy : 0.8202
##               95% CI : (0.7245, 0.8936)
##     No Information Rate : 0.5393
##     P-Value [Acc > NIR] : 2.567e-08
##
##                Kappa : 0.6382
##
##  Mcnemar's Test P-Value : 1
##
##            Sensitivity : 0.8049
##            Specificity : 0.8333
##         Pos Pred Value : 0.8049
##         Neg Pred Value : 0.8333
##            Prevalence : 0.4607
##          Detection Rate : 0.3708
##     Detection Prevalence : 0.4607
##      Balanced Accuracy : 0.8191
##
##        'Positive' Class : Healthy
##
```

4.3 K-nearest Neighbour

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

```
## k-Nearest Neighbors
knn_model = train(num ~ ., data = train, method = "knn", preProcess=c('knnImpute')) knn_model
##
## 208 samples
##         13 predictor
##          2 classes: 'Healthy', 'Disease'
##
## Pre-processing: nearest neighbor imputation (13), centered (13), scaled (13)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 208, 208, 208, 208, 208, 208, ...
## Resampling results across tuning parameters:
##
##         k          Accuracy            Kappa
##         5          0.7819603           0.5596682
##         7          0.7900055           0.5753638
##         9          0.7990690           0.5939902
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
predictions = predict(knn_model, newdata = test) confusionMatrix <- confusionMatrix(predictions, test$num)
results[nrow(results) + 1, ] <- c(as.character('K-nearest neighbours (knn)'),
confusionMatrix$overall['Accuracy'], confusionMatrix$byClass['Sensitivity'],
confusionMatrix$byClass['Specificity'])
rm(knn_model, predictions) confusionMatrix
## Confusion Matrix and Statistics
##
##         Reference
## Prediction Healthy Disease
## Healthy       33      4
## Disease        8     44
##
##             Accuracy : 0.8652
##               95% CI : (0.7763, 0.9283)
##     No Information Rate : 0.5393
##     P-Value [Acc > NIR] : 5.93e-11
##
##                Kappa : 0.7267
##
##  Mcnemar's Test P-Value : 0.3865
##
##            Sensitivity : 0.8049
##            Specificity : 0.9167
##         Pos Pred Value : 0.8919
##         Neg Pred Value : 0.8462
##             Prevalence : 0.4607
##          Detection Rate : 0.3708
##     Detection Prevalence : 0.4157
##       Balanced Accuracy : 0.8608
```

```
##
##          'Positive' Class : Healthy
##
```

4.4 Random Forest

Random Forest algorithm will create a “forest” of randomly chosen decision trees, which result in a classification. The results of these trees will then be compared and the best performing tree is selected.

```
rf_model = train(num ~ ., data = train, method = "rf") predictions =
predict(rf_model, newdata = test) confusionMatrix <-
confusionMatrix(predictions, test$num)
results[nrow(results) + 1, ] <- c(as.character('Random Forest (rf)'),
                                   confusionMatrix$overall['Accuracy'],
                                   confusionMatrix$byClass['Sensitivity'],
                                   confusionMatrix$byClass['Specificity'])
 rm(rf_model, predictions)
 confusionMatrix
## Confusion Matrix and Statistics
##
##          Reference
## Prediction Healthy Disease
##   Healthy      31      6
##   Disease      10     42
##
##             Accuracy : 0.8202
##               95% CI : (0.7245, 0.8936)
##     No Information Rate : 0.5393
##     P-Value [Acc > NIR] : 2.567e-08
##
##               Kappa : 0.6356
##
##  Mcnemar's Test P-Value : 0.4533
##
##           Sensitivity : 0.7561
##           Specificity : 0.8750
##         Pos Pred Value : 0.8378
##         Neg Pred Value : 0.8077
##           Prevalence : 0.4607
##         Detection Rate : 0.3483
##    Detection Prevalence : 0.4157
##      Balanced Accuracy : 0.8155
##
##         'Positive' Class : Healthy
##
```

5. RESULTS AND DISCUSSION

The results of all the above models are:

Model	Accuracy	Sensitivity	Specificity
Linear Classifier	0.820224719101124	0.804878048780488	0.833333333333333
Random Forest	0.820224719101124	0.75609756097561	0.875
Naive Bayes	0.842696629213483	0.829268292682927	0.854166666666667
K-nearest neighbours	0.865168539325843	0.804878048780488	0.916666666666667

We can see that good accuracy is given by K-nearest Neighbours algorithm. Random Forest and Linear Classifier gave the exact same accuracy but different specificity and sensitivity. A good combination of specificity and sensitivity can be seen in Naive Bayes algorithm.

Various algorithms have been described. Modeling for prediction has been carried out using various algorithms. The accuracy of various algorithms is different and it is highest for K-nearest neighbours model. The sensitivity is highest for Naïve Bayes model and specificity is highest for K-nearest neighbours. It is established here that K-nearest neighbours is probably best choice model among all.

CONCLUSION

Looking at all the metrics from all the models, K-nearest Neighbour seems to be the best choice among the four to predict whether a person is diseased or not. It has highest specificity and accuracy among all the four. This means that we have a better chance of predicting a heart disease patient correctly. This comparative study helps us to understand how a comparision among different machine learning algorithms based on several performance metrices is necessary to build a good prediction system.

REFERENCES

[1] Divyansh Khanna, Rohan Sahu, Veeky Baths, and Bharat Deshpande, Comparative Study of Classification Techniques (SVM, Logistic Regression and Neural Networks) to Predict the Prevalence of Heart Disease, International Journal of Machine Learning and Computing, Vol. 5, No. 5, October 2015.

[2] Poornima Singh, Sanjay Singh, Gayatri S Pandi-Jain, Effective heart disease prediction system using data mining techniques, International Journal of Nanomedicine, Volume 13, T-NANO 2014.

[3] Sonam Nikhar, A.M. Karandikar “Prediction of Heart Disease Using Machine Learning Algorithms” in International Journal of Advanced Engineering, Management and Science (IJAEMS) June-2016 vol-2.

[4] Ponrathi Athilingam, Bradlee Jenkins, Marcia Johansson, Miguel Labrador "A Mobile Health Intervention to Improve Self-Care in Patients With Heart Failure: Pilot Randomized Control Trial" in JMIR Cardio 2017, vol. 1, issue 2, pg no:1.

[5] Mai Shouman, Tim Turner, Rob Stocker “Applying k-Nearest Neighbour in Diagnosing Heart Disease Patients” in International Journal of Information and Education Technology, Vol. 2, No. 3, June 2012.

[6] G. Parthiban, S. K. Srivatsa "Applying machine learning methods in diagnosing heart disease for diabetic patients" in International Journal of Applied Information Systems, Vol.3, No.7, pp.2249-0868, 2012.

[7] Nagaraj M Lutimath,Chethan C,Basavaraj S Pol. “Prediction Of Heart Disease using Machine Learning” in International journal Of Recent Technology and Engineering,8,(2S10), pp 474-477, 2019.