Heart Disease Data Set

Shashi Roshan (NetID : sroshan2)

Abstract

Statistical learning models were applied to heart disease data in order to predict presence of possible heart disease on attributes of the various health related tests. A variety of learning techniques were explored and validated, and ultimately the predictive power of the models appears to be good.

Introduction

Heart disease is a common problem which is prevelant in the society. There are various factors which can cause heart disease, including lifestyle habits, smoking, etc. Data can be used to extract information based on which predictions about presence or absence of heart disease can be made. This can be helpful is reinforcing the diagnostics of a doctor's decision about the presence of heart disease. In certain scenarios, the predictions made from the data might be helpful to avoid the use of complicated costlier tests which are used to detect the presence of heart diseases. We use data to predict if a person might be having a heart disease or not - a binary outcome.

Methods

Data

We took the data from UCI Heart Disease Data set repository. The original data had 76 columns, and a lot more rows, but we took a sample of it, which had 12 attributes. It includes demographic information like age, sex and location, as well as the results of few medical tests, which can help diagnose the presence or absence of heart diseases. Based on the data we received, we created an additional attribute (num_bin), which indicates if some form of heart disease was detected or not. The data set is quite balanced - the proportion of cases with presence and absence of heart diseases is almost same. Exploratory Data Analysis was performed on the data set.

```
# load packages
library("tidyverse")
library("caret")
library('kableExtra')
library('randomForest')

# set seed
set.seed(5079)

# read in data
heart = read_csv('heart-disease.csv')

# coerce character columns to factor
heart = heart %>%
```

```
  mutate_if(is.character, as.factor)

# create binary response variable
heart$num_bin = factor(case_when(heart$num == "v0" ~ "none",
                                 TRUE ~ "some"))

# randomly subset data for training
heart_trn = heart %>% sample_frac(0.80)
heart_tst = setdiff(heart, heart_trn)
```

Modelling

In order to predict the presence or absence of heart disease, four modeling techniques were considered: logistic regression model, k-nearest neighbors model, random forest model and decision tree model. - Logistic regression model was trained using all available predictor variables. - k-nearest neighbors model was trained using all available predictor variables. The choice of k was chosen using cross validation. - Decision tree model was trained using all available predictors. The choice of the complexity parameter was chosen using cross validation. - Random Forest model was trained using all available predictors. The choice of mtry was chosen using out-of-bag metric.

Evaluation

To evaluate the performance of different models, we split the data into train and test set. On the training set, we did cross validation technique to find the optimal set of parameters. Accuracy metric was used to select the best model while performing hyperparameter tuning. Once all the model's performance was recorded using cross-validation results, we picked the best model. Then, the best model was tested on the test dataset.

```
set.seed(42)
knn_model = train(
  num_bin ~ . - num,
  data = heart_trn,
  method = 'knn',
  trControl = trainControl(method = 'cv')
)
```

```
tree_model = train(
  num_bin ~ . - num,
  data = heart_trn,
  method = 'rpart',
  trControl = trainControl(method = 'cv', number = 5)
)
```

```
rf_model = train(
  num_bin ~ . - num,
  data = heart_trn,
  method = 'rf',
  trControl = trainControl(method = 'oob', number = 5)
)
```

```
glm_model = train(
  num_bin ~ . - num,
  data = heart_trn,
  method = 'glm',
  trControl = trainControl(method = 'cv', number = 5),
  family = 'binomial'
)
```

Results

```
model_names = c('Generalized Linear Model',
                'KNN',
                'Decision Tree',
                'Random Forest')

model_results = c(
  max(glm_model$results$Accuracy),
  max(knn_model$results$Accuracy),
  max(tree_model$results$Accuracy),
  max(rf_model$results$Accuracy)
)

model_description = c('', 'K = 5', 'cp = 0.051', 'mtry = 2')

df = data.frame(model_names, model_results, model_description)
df$model_results = round(df$model_results, 2)

tibble(
  'Model' = df$model_names,
  'Accuracy' = round(df$model_results, 2),
  'Model Description' = df$model_description
) %>%
  kable(digits = 2) %>%
  kable_styling("striped", full_width = FALSE)
```

Model	Accuracy	Model Description
Generalized Linear Model	0.81	
KNN	0.67	K = 5
Decision Tree	0.76	cp = 0.051
Random Forest	0.79	mtry = 2

```
rf = randomForest(num_bin ~ . - num,
  data = heart_trn,
  mtry=2)
rf
```

```
##
## Call:
##  randomForest(formula = num_bin ~ . - num, data = heart_trn, mtry = 2)
##                Type of random forest: classification
```

```
##                       Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 20.95%
## Confusion matrix:
##      none some class.error
## none  235   59   0.2006803
## some   65  233   0.2181208
```

```
importance(rf)
```

```
##          MeanDecreaseGini
## age             27.071384
## sex             12.886693
## cp              42.926767
## trestbps        22.964888
## chol            28.664902
## fbs              4.997740
## restecg          8.307199
## thalach         37.896765
## exang           30.225671
## oldpeak         34.846344
## location        19.595646
```

```
rf2 = randomForest(num_bin ~ cp + thalach + oldpeak + exang,
  data = heart_trn,
  mtry=2)
rf2
```

```
##
## Call:
##  randomForest(formula = num_bin ~ cp + thalach + oldpeak + exang,      data = heart_trn, mtry = 2)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 25.17%
## Confusion matrix:
##      none some class.error
## none  217   77   0.2619048
## some   72  226   0.2416107
```

```
rf_model2 = train(
  num_bin ~ cp + thalach  + oldpeak + exang,
  data = heart_trn,
  method = 'rf',
  trControl = trainControl(method = 'oob', number = 5)
)
max(rf_model2$results$Accuracy)
```

```
## [1] 0.7652027
```

```
glm_model2 = train(
  num_bin ~ cp + thalach + oldpeak + exang,
  data = heart_trn,
  method = 'glm',
  trControl = trainControl(method = 'cv', number = 5),
  family = 'binomial'
)
max(glm_model2$results$Accuracy)
```

```
## [1] 0.7770546
```

```
p1 = predict(glm_model, heart_tst, type = 'prob')[, 'some']
p1 = factor(ifelse(p1 >= 0.4, 'some', 'none'), levels = c('none', 'some'))
cf = confusionMatrix(data = p1,
                     reference = heart_tst$num_bin,
                     positive = 'some')
```

Discussion

We used the UCI Heart disease data set to predict if a person has heart disease or not. Logistic regression model seems to be giving the best performance.

Accuracy : 81.63 %
Specificity : 77.78 %
Sensitivity : 84.52 %

We found that the predictive performance of the best model is quite satisfactory, and it predicts positive cases of heart diseases with very good accuracy. The No Information rate for this model is 50%. We tweaked the logistic model's threshold from 0.5 (default) to 0.4, in order to better classify the positive cases of heart diseases.

We should not that the data which we have used is a subset from the larger population, and future work can include creation of statistical models on the original data set. Future work can also include using this data (with data from additional sources) to predict the severity of the heart disease. Multi-class classification is a difficult problem, as compared to binary classification problem.