Heart Attack Prediction

Sercan Öncü

5/28/23

Problem, Features and Target

Analyze and predict the occurrence of heart attacks based on the given dataset. The dataset includes various features that provide information about individuals. These features include age, sex, cp, chol etc. The target variable in this dataset is the presence or absence of a heart attack. It is represented by the "output" column, where 1 indicates the presence of a heart attack and 0 indicates its absence.

Terms of the dimension, variable type

The dataset includes 303 observations and 14 double variables in total.

```
glimpse(heart)
```

```
Rows: 303
Columns: 14
$ age      <dbl> 63, 37, 41, 56, 57, 57, 56, 44, 52, 57, 54, 48, 49, 64, 58, 5~
$ sex      <dbl> 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1~
$ cp       <dbl> 3, 2, 1, 1, 0, 0, 1, 1, 2, 2, 0, 2, 1, 3, 3, 2, 2, 3, 0, 3, 0~
$ trtbps   <dbl> 145, 130, 130, 120, 120, 140, 140, 120, 172, 150, 140, 130, 1~
$ chol     <dbl> 233, 250, 204, 236, 354, 192, 294, 263, 199, 168, 239, 275, 2~
$ fbs      <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0~
$ restecg  <dbl> 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1~
$ thalachh <dbl> 150, 187, 172, 178, 163, 148, 153, 173, 162, 174, 160, 139, 1~
$ exng     <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0~
$ oldpeak  <dbl> 2.3, 3.5, 1.4, 0.8, 0.6, 0.4, 1.3, 0.0, 0.5, 1.6, 1.2, 0.2, 0~
$ slp      <dbl> 0, 0, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 0, 2, 2, 1~
$ caa      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0~
```

```
$ thall    <dbl> 1, 2, 2, 2, 2, 1, 2, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3~
$ output   <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
```

Splitting The Dataset

This code sets a random seed, then samples 80% of the data to be used for training by selecting a random set of row indices. The remaining 20% of the data is assigned to the test dataset.

```
set.seed(123)
heart_split <- initial_split(data = heart,
                             prop = 0.80)
train <- heart_split |> training()
test  <- heart_split |> testing()
```

Training Logistic Regression Model

```
lr_model <- glm(output ~ ., data = train, family = "binomial")
summary(lr_model)
```

```

Call:
glm(formula = output ~ ., family = "binomial", data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6127  -0.4311   0.1795   0.5835   2.4205

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.555347   2.794683   1.630  0.10310
age         -0.013283   0.025289  -0.525  0.59941
sex         -1.540661   0.509861  -3.022  0.00251 **
cp           0.769378   0.195309   3.939 8.17e-05 ***
trtbps      -0.017916   0.011003  -1.628  0.10348
chol        -0.003182   0.004177  -0.762  0.44617
fbs          0.132748   0.566248   0.234  0.81465
restecg      0.658516   0.389009   1.693  0.09049 .
thalachh     0.019761   0.011439   1.727  0.08408 .
```

```
exng        -1.077694   0.448609  -2.402  0.01629 *
oldpeak     -0.678520   0.240961  -2.816  0.00486 **
slp          0.263729   0.411859   0.640  0.52195
caa         -0.646435   0.201346  -3.211  0.00132 **
thall       -1.015510   0.330194  -3.075  0.00210 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 333.48  on 241  degrees of freedom
Residual deviance: 173.07  on 228  degrees of freedom
AIC: 201.07

Number of Fisher Scoring iterations: 6
```

The model's fit was evaluated using the null deviance (333.5 with 241 degrees of freedom) and the residual deviance (173.07 with 228 degrees of freedom). A lower residual deviance indicates a better fit. The AIC value (201.07) can be used to compare the quality of different models, with lower AIC values indicating better models.

Model Performance

```r
predicted_probs <- predict(lr_model, test[,-14], type = "response")
```

```r
predicted_classes <- ifelse(predicted_probs > 0.5, 1, 0)
```

```r
TP <- sum(predicted_classes[which(test$output == "1")] == 1)
FP <- sum(predicted_classes[which(test$output == "1")] == 0)
TN <- sum(predicted_classes[which(test$output == "0")] == 0)
FN <- sum(predicted_classes[which(test$output == "0")] == 1)
recall      <- TP / (TP + FN)
specificity <- TN / (TN + FP)
precision   <- TP / (TP + FP)
accuracy    <- (TN + TP) / (TP + FP + TN + FN)
recall
```

```
[1] 0.7948718
```

```
specificity
```

```
[1] 0.9090909
```

```
precision
```

```
[1] 0.9393939
```

```
accuracy
```

```
[1] 0.8360656
```

The accuracy is 0.8360, which shows that 83.60% of all cases are correctly classified by the model.

```
table(train$output) / dim(train)[1]
```

```

        0         1 
0.4545455 0.5454545 
```

The class distribution in the training dataset is balanced, with approximately 54.54% of examples belonging to the negative class and 45.45% belonging to the positive class.

```
confusionMatrix(table(ifelse(test$output == "1", "1", "0"),
                      predicted_classes),
                positive = "1")
```

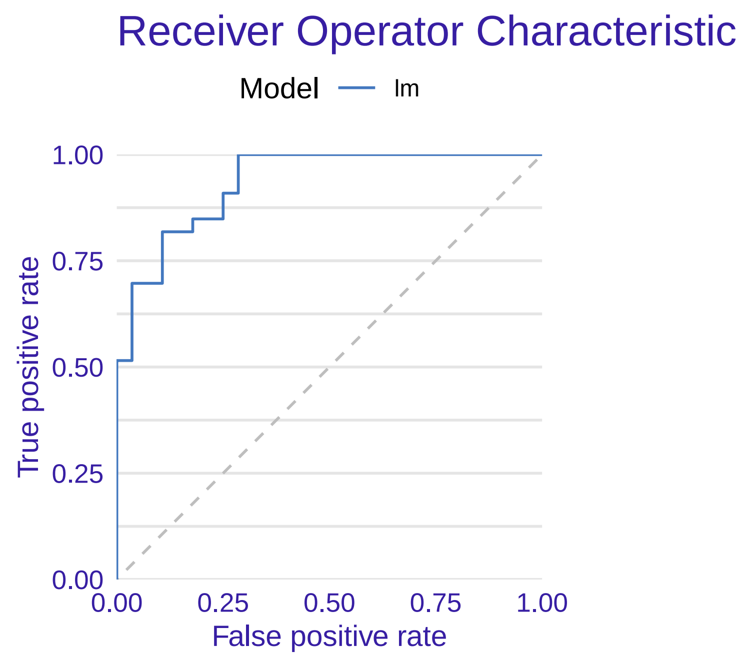
```
Confusion Matrix and Statistics

   predicted_classes
     0  1
  0 20  8
  1  2 31
                                          
               Accuracy : 0.8361          
                 95% CI : (0.7191, 0.9185)
```

```
   No Information Rate : 0.6393
   P-Value [Acc > NIR] : 0.000614

                 Kappa : 0.6645

Mcnemar's Test P-Value : 0.113846

           Sensitivity : 0.7949
           Specificity : 0.9091
        Pos Pred Value : 0.9394
        Neg Pred Value : 0.7143
            Prevalence : 0.6393
        Detection Rate : 0.5082
  Detection Prevalence : 0.5410
     Balanced Accuracy : 0.8520

      'Positive' Class : 1
```

Model evaluation metrics for the test dataset are as follows:Accuracy = 0.8361 (proportion of correct predictions), Kappa = 0.6645 (an index measuring classification accuracy ranging between 0 and 1, with higher values indicating better performance), Balanced Accuracy = 0.8520 (average of sensitivity and specificity). Overall, the model's performance on the test dataset appears to be good, with high accuracy, specificity, and negative predictive value.

```
explain_lr <- explain(model   = lr_model,
                      data    = test[, -14],
                      y       = test$output == "1",

                      type    = "classification",
                      verbose = FALSE)
```

```
performance_lr <- model_performance(explain_lr)
plot(performance_lr, geom = "roc")
```




```
performance_lr
```

```
Measures for:  classification
recall      : 0.9393939
precision   : 0.7948718
f1          : 0.8611111
accuracy    : 0.8360656
auc         : 0.9339827

Residuals:
         0%         10%         20%         30%         40%         50% 
-0.92500426 -0.66440247 -0.23803316 -0.05420990 -0.00710291  0.01057154 
        60%         70%         80%         90%        100% 
 0.02908870  0.05916045  0.19665257  0.30022165  0.59491469 
```

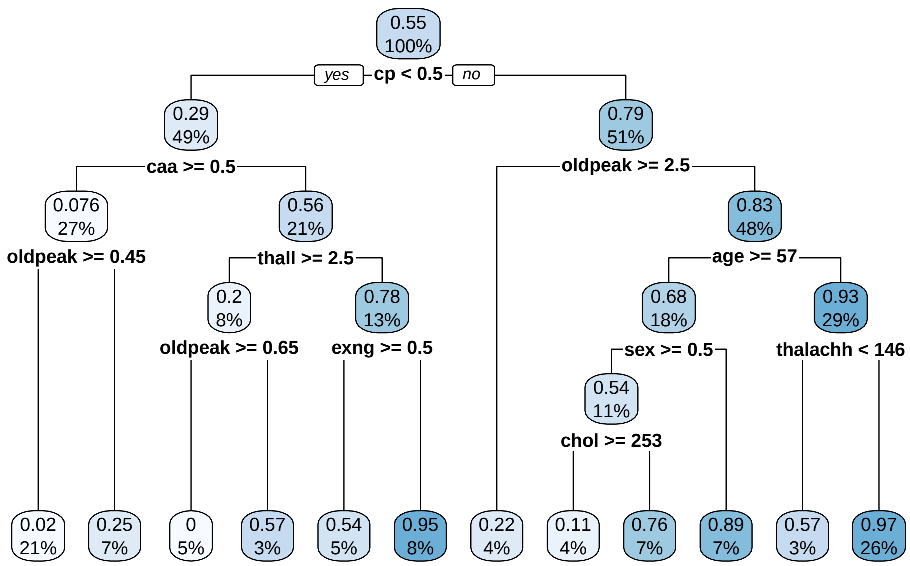
F1 Score:0.8611 - The harmonic mean of precision and recall, used to measure the model's balanced performance. AUC (Area Under Curve):0.9340 - The area under the Receiver Operating Characteristic (ROC) curve, used to measure the model's classification performance (values closer to 1 indicate better performance).

Training Desicion Tree

```r
dt_model <- decision_tree() |>
  set_engine("rpart") |>
  set_mode("regression")
```

```r
dt_heart <- dt_model |>
  fit(output ~., data = train)
```

```r
rpart.plot(dt_heart$fit)
```




```r
heart_predictions <- dt_heart |>
  predict(new_data = test)
```

```r
heart_results <- tibble(predicted = heart_predictions$.pred,
                        actual    = test$output)
```

```r
heart_results |> rmse(truth = actual, estimate = predicted)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard       0.361
```

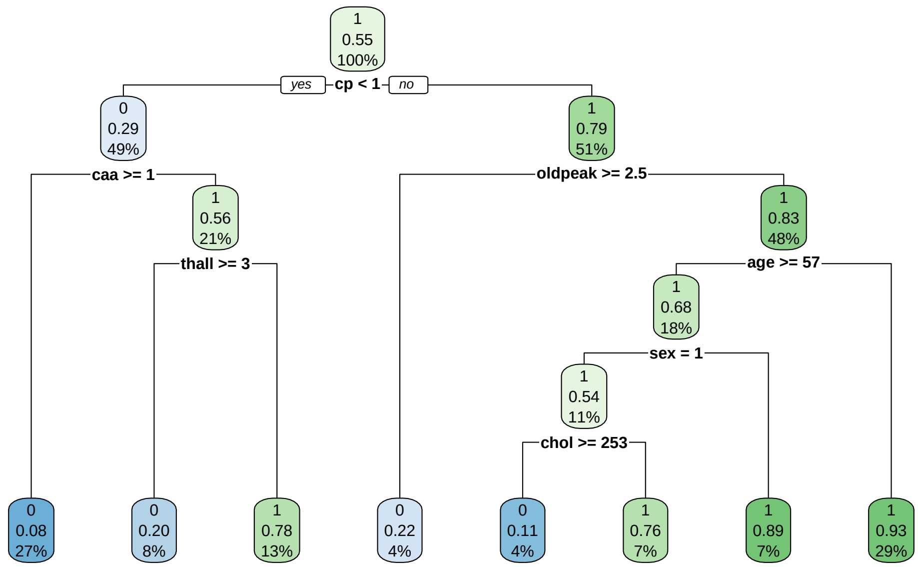
```r
heart_last_fit <- dt_model |>
  last_fit(output ~., split = heart_split)

heart_last_fit |> collect_metrics()
```

```
# A tibble: 2 x 4
  .metric .estimator .estimate .config
  <chr>   <chr>          <dbl> <chr>
1 rmse    standard       0.361 Preprocessor1_Model1
2 rsq     standard       0.500 Preprocessor1_Model1
```

```r
heart_last_fit |> collect_predictions()
```

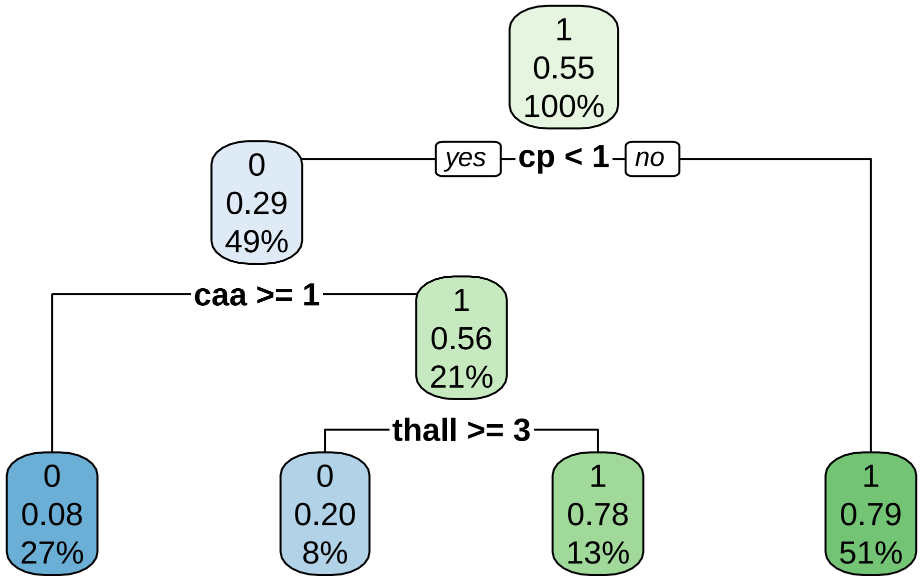
```
# A tibble: 61 x 5
   id               .pred  .row output .config
   <chr>            <dbl> <int>  <dbl> <chr>
 1 train/test split 0.222     2      1 Preprocessor1_Model1
 2 train/test split 0.969     3      1 Preprocessor1_Model1
 3 train/test split 0.571    12      1 Preprocessor1_Model1
 4 train/test split 0.889    15      1 Preprocessor1_Model1
 5 train/test split 0.947    19      1 Preprocessor1_Model1
 6 train/test split 0.571    28      1 Preprocessor1_Model1
 7 train/test split 0.969    38      1 Preprocessor1_Model1
 8 train/test split 0.947    44      1 Preprocessor1_Model1
 9 train/test split 0.969    47      1 Preprocessor1_Model1
10 train/test split 0.571    49      1 Preprocessor1_Model1
# i 51 more rows
```

```r
heart_dt <- rpart(output ~ .,
                   data = train,
                   method = "class")
rpart.plot(heart_dt)
```



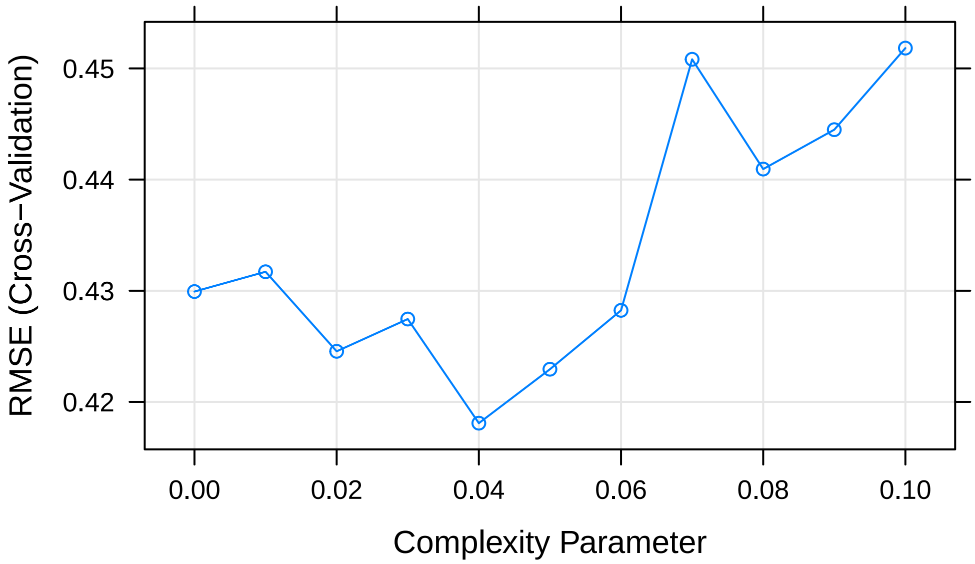
Training decision tree by tuning `cp` and `maxdepth`

```
less_dt1 <- rpart(output ~ .,
                  data = train,
                  method = "class",
                  maxdepth = 3,
                  cp = 0.046)
rpart.plot(less_dt1)
```

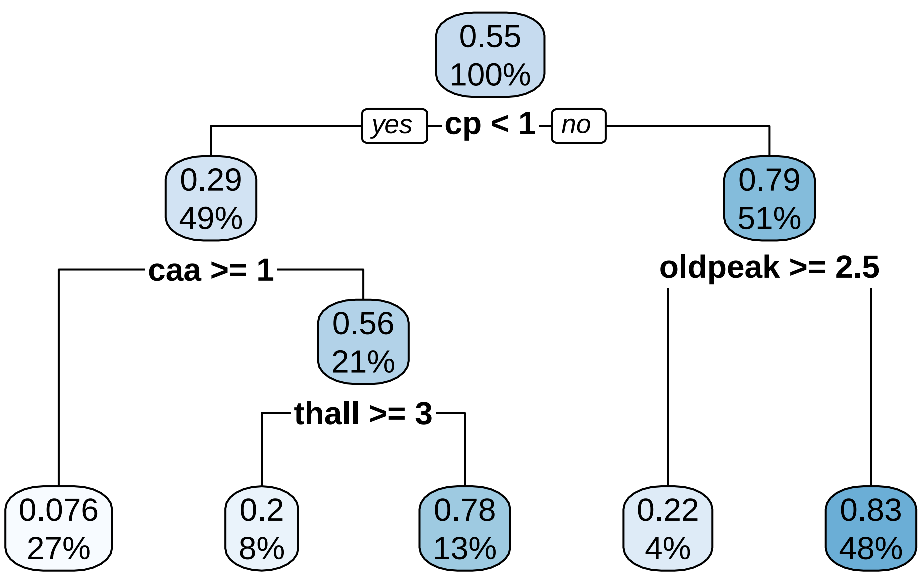


Grid search in caret

```
fit_control <- trainControl(method = "cv", number = 10)
dt_model2 <- train(output ~ .,
                   data = train,
                   method = "rpart",
                   trControl = fit_control,
                   tuneGrid = expand.grid(cp = seq(0, 0.1, 0.01)))
plot(dt_model2)
```




```
rpart.plot(dt_model2$finalModel)
```



Model Performance

```r
factor_test <- as.factor(test$output)
heart_preds <- predict(heart_dt, test, type = "class")
confusionMatrix(heart_preds,
                factor_test,positive = "1")
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 20  2
         1  8 31

               Accuracy : 0.8361
                 95% CI : (0.7191, 0.9185)
    No Information Rate : 0.541
    P-Value [Acc > NIR] : 1.184e-06

                  Kappa : 0.6645

 Mcnemar's Test P-Value : 0.1138

            Sensitivity : 0.9394
            Specificity : 0.7143
         Pos Pred Value : 0.7949
         Neg Pred Value : 0.9091
             Prevalence : 0.5410
         Detection Rate : 0.5082
   Detection Prevalence : 0.6393
      Balanced Accuracy : 0.8268

       'Positive' Class : 1
```

Model evaluation metrics for the test dataset are as follows:Acccuracy = 0.8361 (proportion of correct predictions), Kappa = 0.6645, Balanced Accuracy = 0.8268

Training Random Forests

```
set.seed(123)
trained_rf <- ranger(output ~ .,data = train)
```

Model Performance

```
preds_rf <- predict(trained_rf, test)
confusionMatrix(as.factor(ifelse(preds_rf$predictions > 0.5, "1", "0")),
                as.factor(test$output),positive = "1")
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 19  1
         1  9 32

               Accuracy : 0.8361
                 95% CI : (0.7191, 0.9185)
    No Information Rate : 0.541
    P-Value [Acc > NIR] : 1.184e-06

                  Kappa : 0.6626

 Mcnemar's Test P-Value : 0.02686

            Sensitivity : 0.9697
            Specificity : 0.6786
         Pos Pred Value : 0.7805
         Neg Pred Value : 0.9500
             Prevalence : 0.5410
         Detection Rate : 0.5246
   Detection Prevalence : 0.6721
      Balanced Accuracy : 0.8241

       'Positive' Class : 1
```

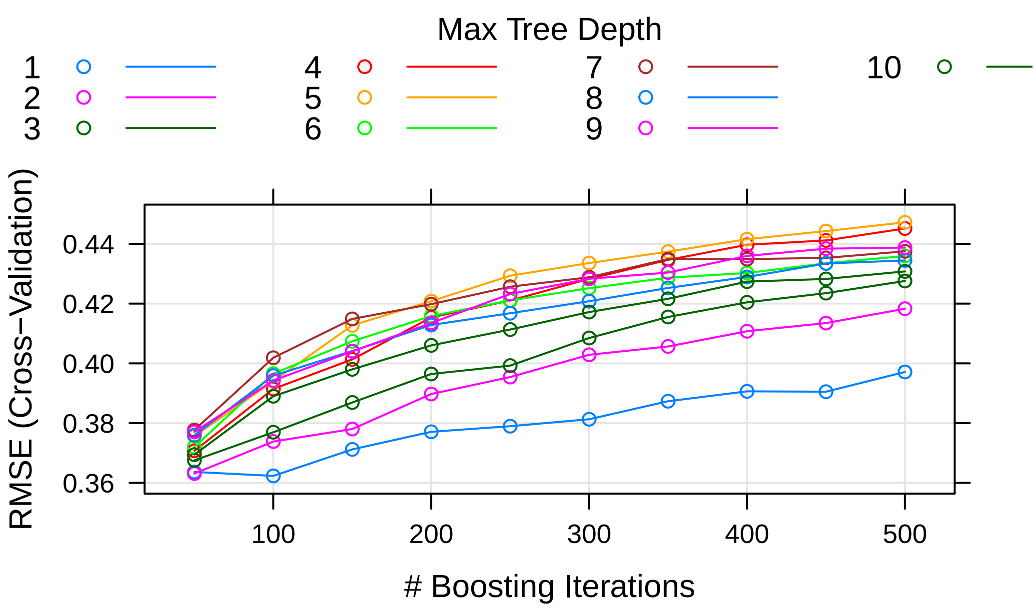
Model evaluation metrics for the test dataset are as follows: Accuracy = 0.8361 (proportion of correct predictions), Kappa = 0.6626, Balanced Accuracy = 0.8241

Training a GBM model

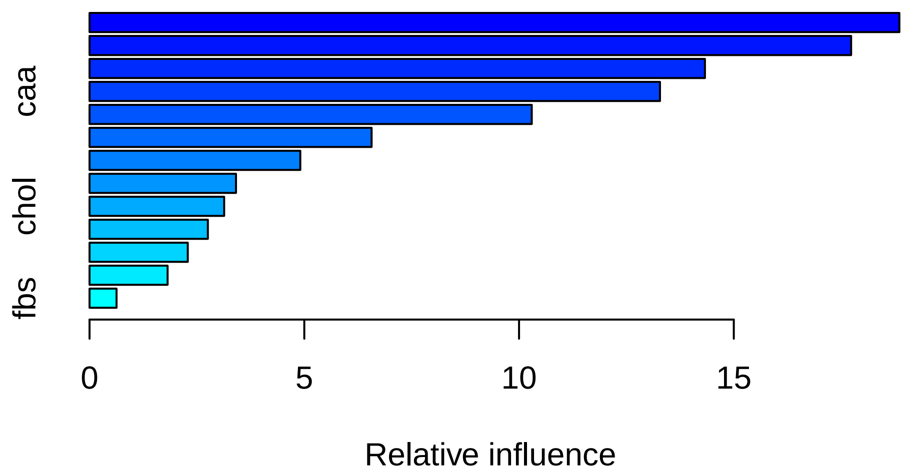
```
set.seed(123)
gbm_fit = train(output ~ .,
                data = train,
                set.seed(123),
                method = "gbm",
                trControl = trainControl(method = "cv", number = 5),
                verbose = FALSE,
                tuneLength = 10)
gbm_fit$bestTune
```

```
  n.trees interaction.depth shrinkage n.minobsinnode
2     100                 1       0.1             10
```

```
plot(gbm_fit)
```



```r
set.seed(123)
summary(gbm_fit)
```




```
              var    rel.inf
cp             cp 18.8552799
oldpeak   oldpeak 17.7316454
thall       thall 14.3283317
caa           caa 13.2813260
thalachh thalachh 10.2964616
trtbps     trtbps  6.5671168
exng         exng  4.9067461
age           age  3.4097477
chol         chol  3.1344733
sex           sex  2.7550810
restecg   restecg  2.2872473
slp           slp  1.8173657
fbs           fbs  0.6291775
```

When looking at the RMSE graph, it can be observed that increasing the sample size leads to an increase in error and results in over-sampling. On the other hand, increasing the depth also leads to an increase in the error rate. In the other graph, it is evident that the most influential variable in predicting the target variable is ‘cp’.

Model Performance

```r
gbm_preds <- predict(gbm_fit, test)
factor_gbm <- as.factor(ifelse(gbm_preds > 0.5, "1", "0"))
confusionMatrix(factor_gbm,
                factor_test,
                positive = "1")
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 19  1
         1  9 32

               Accuracy : 0.8361
                 95% CI : (0.7191, 0.9185)
    No Information Rate : 0.541
    P-Value [Acc > NIR] : 1.184e-06

                  Kappa : 0.6626

 Mcnemar's Test P-Value : 0.02686

            Sensitivity : 0.9697
            Specificity : 0.6786
         Pos Pred Value : 0.7805
         Neg Pred Value : 0.9500
             Prevalence : 0.5410
         Detection Rate : 0.5246
   Detection Prevalence : 0.6721
      Balanced Accuracy : 0.8241

       'Positive' Class : 1
```

Model evaluation metrics for the test dataset are as follows: Accuracy = 0.8525 (proportion of correct predictions), Kappa = 0.6972, Balanced Accuracy = 0.8420

When compared to other models, this model has achieved the highest accuracy, making it the most successful among them.