Data Science Final Project

MODEL 1A: BAGGING

Kjay O. Coca

2022-12-14

LOAD PACKAGES

```r
# Helper packages
library(dplyr)       # for data wrangling
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)     # for awesome plotting
library(doParallel)  # for parallel backend to foreach
```

```
## Loading required package: foreach

## Loading required package: iterators

## Loading required package: parallel
```

```r
library(foreach)     # for parallel processing with for loops
library(rsample)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --

## v tibble  3.1.8     v purrr   0.3.5
## v tidyr   1.2.1     v stringr 1.4.1
## v readr   2.1.3     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x purrr::accumulate() masks foreach::accumulate()
## x dplyr::filter()     masks stats::filter()
## x dplyr::lag()        masks stats::lag()
## x purrr::when()       masks foreach::when()
```

```
library(readr)

# Modeling packages
library(caret)       # for general model fitting
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(rpart)       # for fitting decision trees
library(ipred)       # for fitting bagged decision trees
library(ROCR)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

IMPORTING THE DATA

```
set.seed(123)
radiomics_data <- read_csv("D:/1 MASTERS/STAT225/FINAL PROJECT/STAT 325 _FINAL PROJECT_/Normalize Radio
```

```
## Rows: 197 Columns: 431
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (1): Institution
## dbl (430): Failure.binary, Failure, Entropy_cooc.W.ADC, GLNU_align.H.PET, Mi...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

SPLITTING FOR TRAINING AND TESTING

```
for_splitted  <-  sample(1:nrow(radiomics_data), round(nrow(radiomics_data) * 0.8))
radiomicsdata_train <- radiomics_data[for_splitted,]
radiomicsdata_test  <- radiomics_data[-for_splitted,]
```

In this case, I set 80 percent for training data and 20 percent for testing data. There are 39 observation for testing and 158 observation for training and both have 413 variables.

Bagging Model 1

```r
set.seed(123)
bagging_model1 <- bagging(
  formula = Failure.binary ~ .,
  data = radiomicsdata_train,
  nbagg = 200,
  coob = TRUE,
  control = rpart.control(minsplit = 2, cp = 0)
)

bagging_model1
```

```
##
## Bagging regression trees with 200 bootstrap replications
##
## Call: bagging.data.frame(formula = Failure.binary ~ ., data = radiomicsdata_train,
##     nbagg = 200, coob = TRUE, control = rpart.control(minsplit = 2,
##         cp = 0))
##
## Out-of-bag estimate of root mean squared error:  0.2945
```

In this model, `bagging()` is being used. This model create 200 bootstrap replication. It can be adjusted by changing `nbagg` value. The out-of-bag estimate of root mean squared error is 0.2945.

Bagging Model 2

```r
set.seed(123)
bagging_model2 <- train(
  Failure.binary ~ .,
  data = radiomicsdata_train,
  method = "treebag",
  trControl = trainControl(method = "cv", number = 10),
  nbagg = 200,
  control = rpart.control(minsplit = 2, cp = 0)
)
bagging_model2
```

```
## Bagged CART
##
## 158 samples
## 430 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 142, 142, 142, 142, 142, 143, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.2875722  0.6208728  0.1688396
```

In this model, the data was trained using caret and used the `train()` function. This model use 10 cross fold validation. And found out that the RMSE value is 0.2875. Using RMSE, the best model will have lowest RMSE. Thus, Bagging Model 2 is better than Bagging Model 1 since model 2 RMSE is 0.2875 which is less than 0.2945 in model 1.

This part will create a parallel socket cluster using 8.

```
cl <- makeCluster(8)
```

Register the parallel backend.

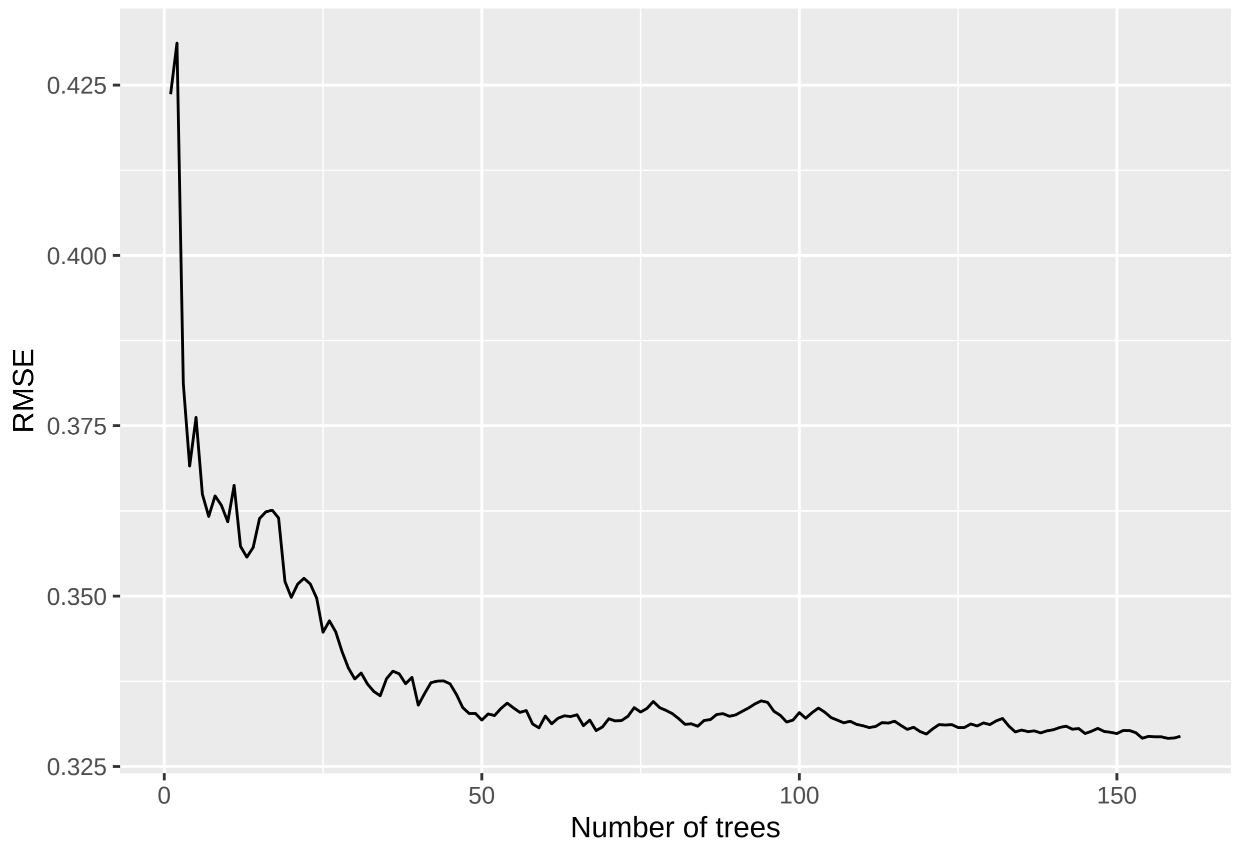
```
registerDoParallel(cl)
```

Chunk below fit trees in parallel and compute predictions on the test set. Bootstrap copy of training data. And show the 10 observation with 8 variables of prediction.

```
predictions <- foreach(
  icount(160),
  .packages = "rpart",
  .combine = cbind
) %dopar% {
  index <- sample(nrow(radiomicsdata_train), replace = TRUE)
  trainDF_boot <- radiomicsdata_train[index, ]
  bagged_tree <- rpart(
    Failure.binary ~ .,
    control = rpart.control(minsplit = 2, cp = 0),
    data = trainDF_boot
  )

  predict(bagged_tree, newdata = radiomicsdata_test)
}

predictions[1:5, 1:7]
```

```
##   result.1 result.2 result.3 result.4 result.5 result.6 result.7
## 1        0        0        0        0        0        1        0
## 2        1        0        1        1        0        1        1
## 3        1        1        1        1        1        1        1
## 4        1        0        1        1        1        1        1
## 5        0        0        0        0        0        0        0
```

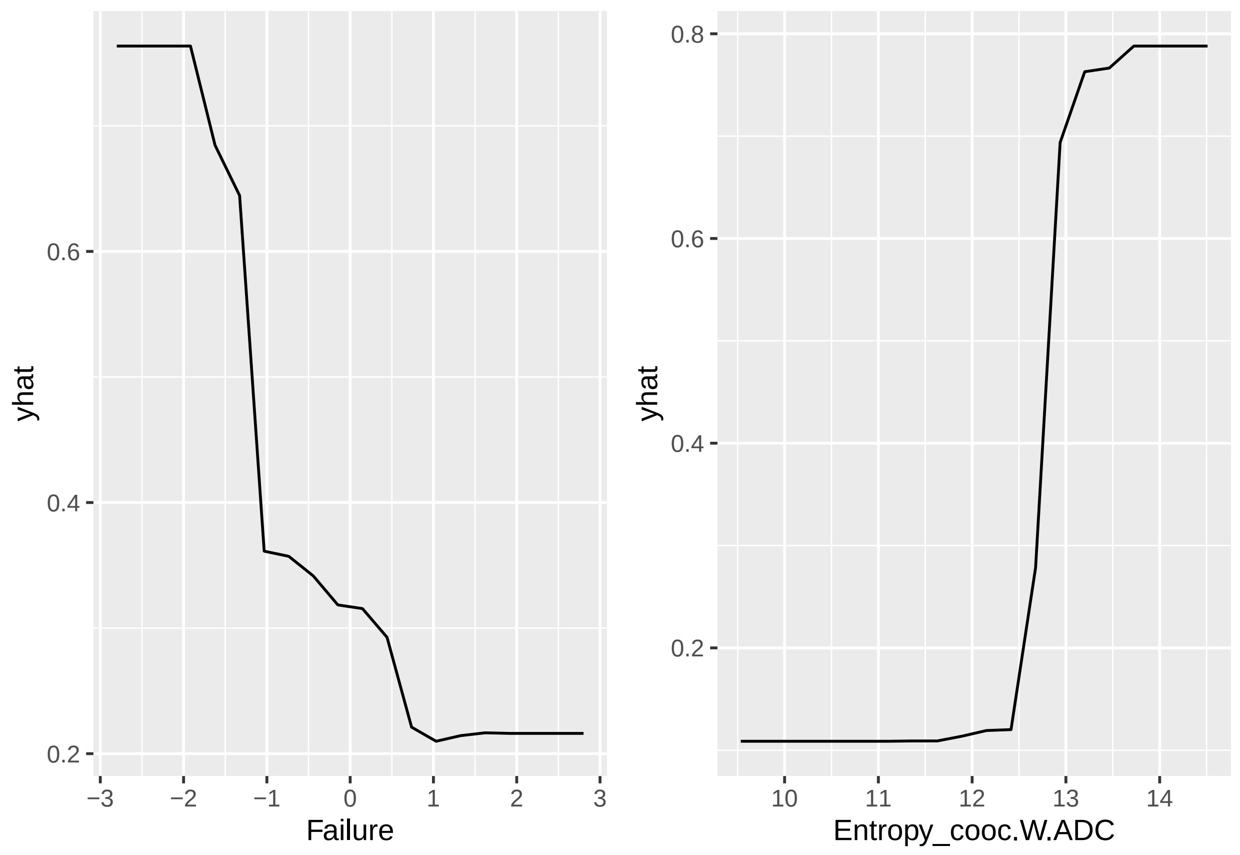
```
predictions %>%
  as.data.frame() %>%
  mutate(
    observation = 1:n(),
    actual = radiomicsdata_test$Failure.binary) %>%
  tidyr::gather(tree,
              predicted,
              -c(observation,
                 actual)) %>%
  group_by(observation) %>%
  mutate(tree = stringr::str_extract(tree, '\\d+') %>% as.numeric()) %>%
  ungroup() %>%
  arrange(observation, tree) %>%
  group_by(observation) %>%
  mutate(avg_prediction = cummean(predicted)) %>%
  group_by(tree) %>%
  summarize(RMSE = RMSE(avg_prediction, actual)) %>%
  ggplot(aes(tree, RMSE)) +
  geom_line() +
  xlab('Number of trees')
```



The above graph shows that the error is stabilizing at 50 to 75 number of trees which means there is no gain for additional trees in making the model.

Construct partial dependence plots

```
plot_1 <- pdp::partial(
  bagging_model2,
  pred.var = names(radiomics_data)[3],
  grid.resolution = 20
) %>%
  autoplot()

plot_2 <- pdp::partial(
  bagging_model2,
  pred.var = names(radiomics_data)[4],
  grid.resolution = 20
) %>%
  autoplot()

gridExtra::grid.arrange(plot_1, plot_2, nrow = 1)
```



The dependence between the Failure.binary response/target variable and these top 2 feature variables, the Failure and Entropy-cooc.W.ADCof interest has shown in the partial dependence plot shows. As shown in PDP above, lower Entropy.cooc.W.ADC implies that Failure.binary is more likely to be 0, and the lower the failure the more likely the Final.binary to be 1.

```
radiomicsdata_train$Failure.binary=as.factor(radiomicsdata_train$Failure.binary)
bagging_model2 <- train(
  Failure.binary ~ .,
  data = radiomicsdata_train,
  method = "treebag",
  trControl = trainControl(method = "cv", number = 10),
  nbagg = 100,
  control = rpart.control(minsplit = 2, cp = 0)
)
# Shutdown parallel cluster
stopCluster(cl)
```

Compute predicted probabilities on training data

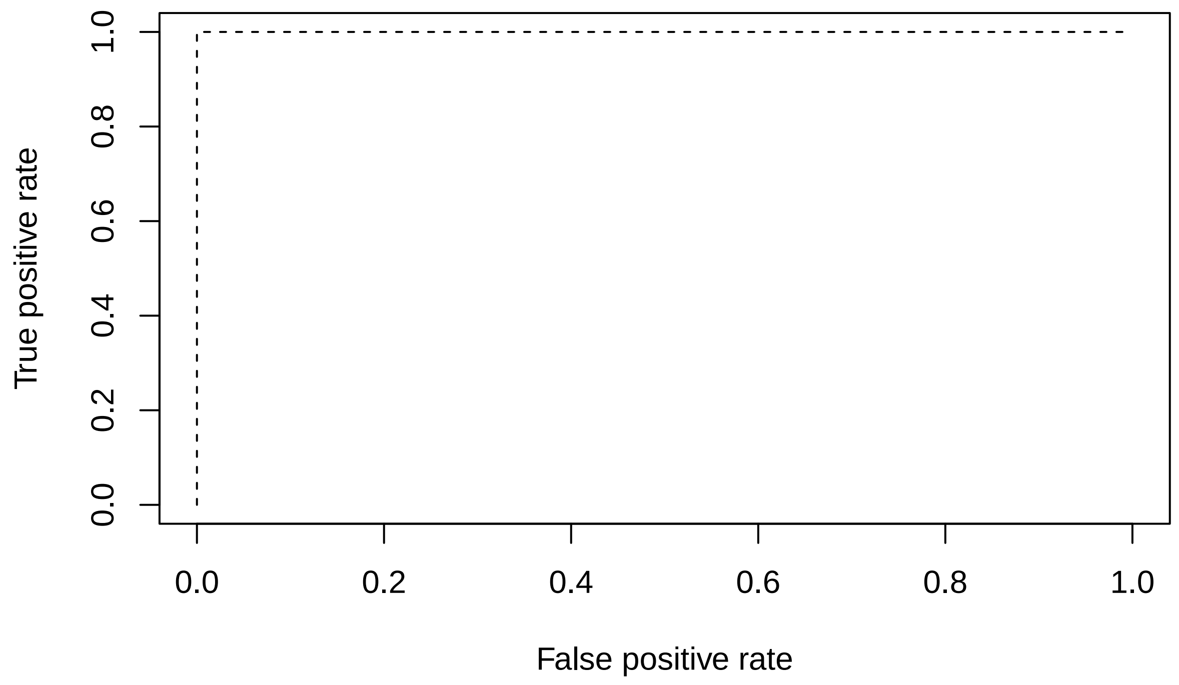
```
pred_prob1 <- predict(bagging_model2, radiomicsdata_train, type = "prob")[,2]
```

Compute AUC metrics for cv_model1,2 and 3

```
perf1 <- prediction(pred_prob1,radiomicsdata_train$Failure.binary) %>%
  performance(measure = "tpr", x.measure = "fpr")
```

Plot ROC curves for cv_model1,2 and 3

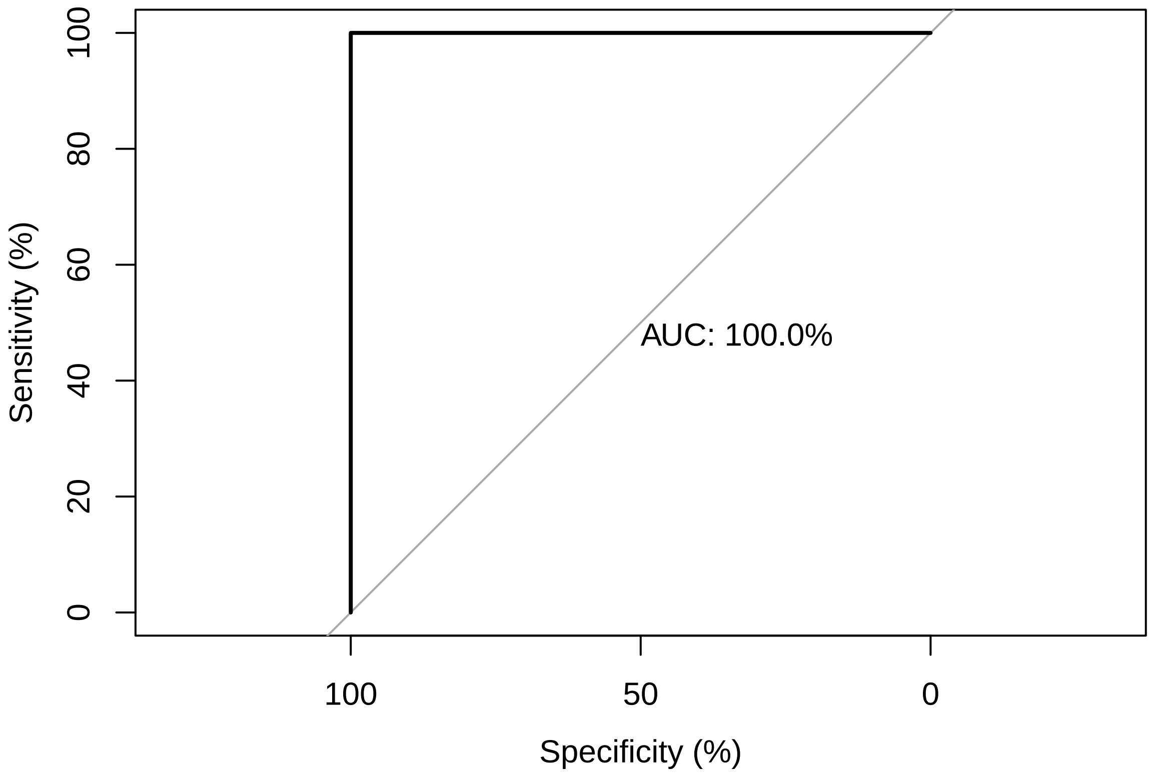
```
plot(perf1, col = "black", lty = 2)
```



ROC plot for training data

```
roc( radiomicsdata_train$Failure.binary ~ pred_prob1, plot=TRUE, legacy.axes=FALSE,
    percent=TRUE, col="black", lwd=2, print.auc=TRUE)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```



```
##
## Call:
## roc.formula(formula = radiomicsdata_train$Failure.binary ~ pred_prob1,     plot = TRUE, legacy.axes =
##
## Data: pred_prob1 in 106 controls (radiomicsdata_train$Failure.binary 0) < 52 cases (radiomicsdata_tra
## Area under the curve: 100%
```

Performance of the model on the training set and found out that the model performed well on our training set which the accuracy is 100 percent. Thus, perfectly classifying the Failure.binary 0 and 1.

Compute predicted probabilities on training data

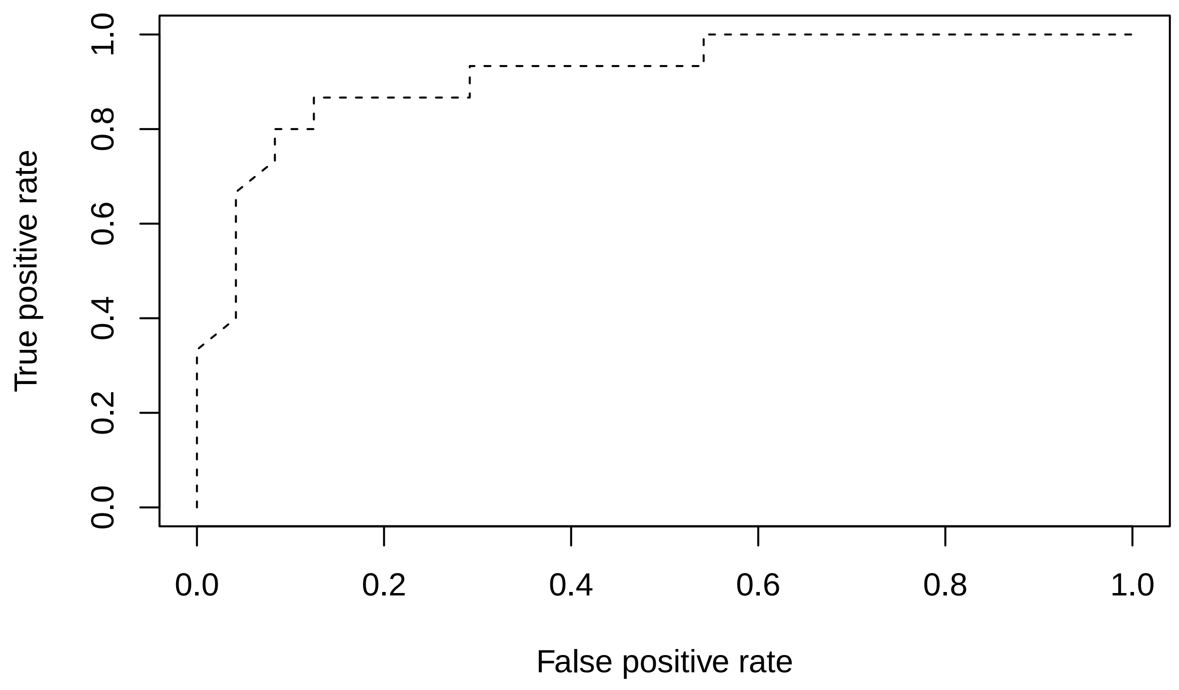
```r
pred_prob2 <- predict(bagging_model2, radiomicsdata_test, type = "prob")[,2]
```

Compute AUC metrics for cv_model1,2 and 3

```r
perf2 <- prediction(pred_prob2,radiomicsdata_test$Failure.binary) %>%
  performance(measure = "tpr", x.measure = "fpr")
```

Plot ROC curves for cv_model1,2 and 3

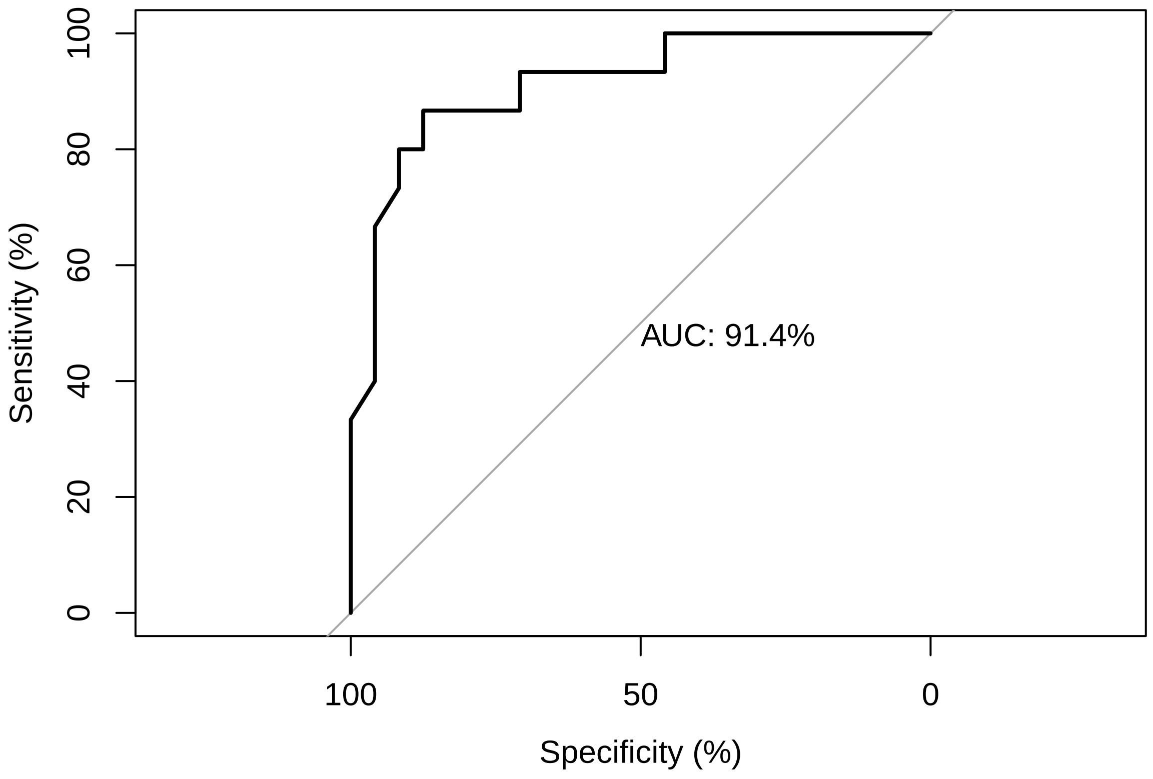
```r
plot(perf2, col = "black", lty = 2)
```



ROC plot for training data

```r
roc( radiomicsdata_test$Failure.binary ~ pred_prob2, plot=TRUE, legacy.axes=FALSE,
    percent=TRUE, col="black", lwd=2, print.auc=TRUE)
```

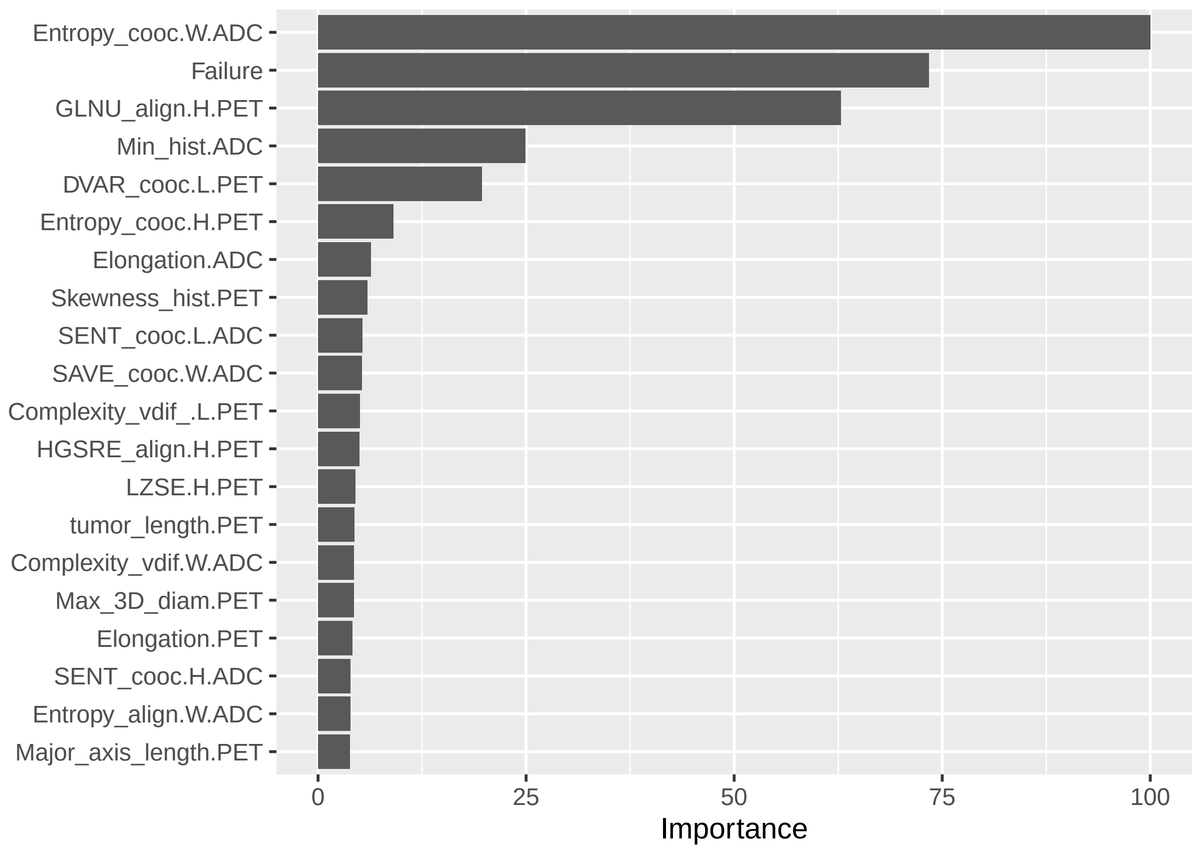
```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```



```
##
## Call:
## roc.formula(formula = radiomicsdata_test$Failure.binary ~ pred_prob2,     plot = TRUE, legacy.axes =
##
## Data: pred_prob2 in 24 controls (radiomicsdata_test$Failure.binary 0) < 15 cases (radiomicsdata_test$
## Area under the curve: 91.39%
```

Performance of the model on the training set and found out that the model performed well on our training set where the accuracy is 91.4 percent.

```r
vip::vip(bagging_model2, num_features = 20)
```



Entropy_cooc.W.ADC, Failure and GLNU_align.H.PET are the top important variables that helps the model to make an accurate model. Also, the Entropy_cooc.W.ADC, Failure and GLNU_align.H.PET are the variables that are very important for the model. The Entropy_cooc.W.ADC has 100 importance level.