Model 1 - Bagging

Freddy A. Julkanain II

Importnt packages are loaded in.

```r
# Helper packages
pacman::p_load(
  tidyverse,#wrangling
  doParallel,#parallelization
  foreach,#parallelization
  rsample,#sampling test and training data
  caret,#modeling package
  rpart,#helper package
  ipred,#modeling package helper
  ROCR,#model perfomrance
  pROC#model performance
)
```

Setting seed for reproducibility and then loading and splitting data to test and training set. The target binary variable is also converted to a factor.

```r
# make bootstrapping reproducible
set.seed(123)
df = read.csv("CleanedDF.csv")
df$Failure.binary=df$Failure.binary%>%as.factor()
indexing  =  sample(1:nrow(df), round(nrow(df) * 0.8))
trnDF = df[indexing,]
tstDF  = df[-indexing,]
```

Training a basic bagging model. (nbagg set low due to computing power limits)

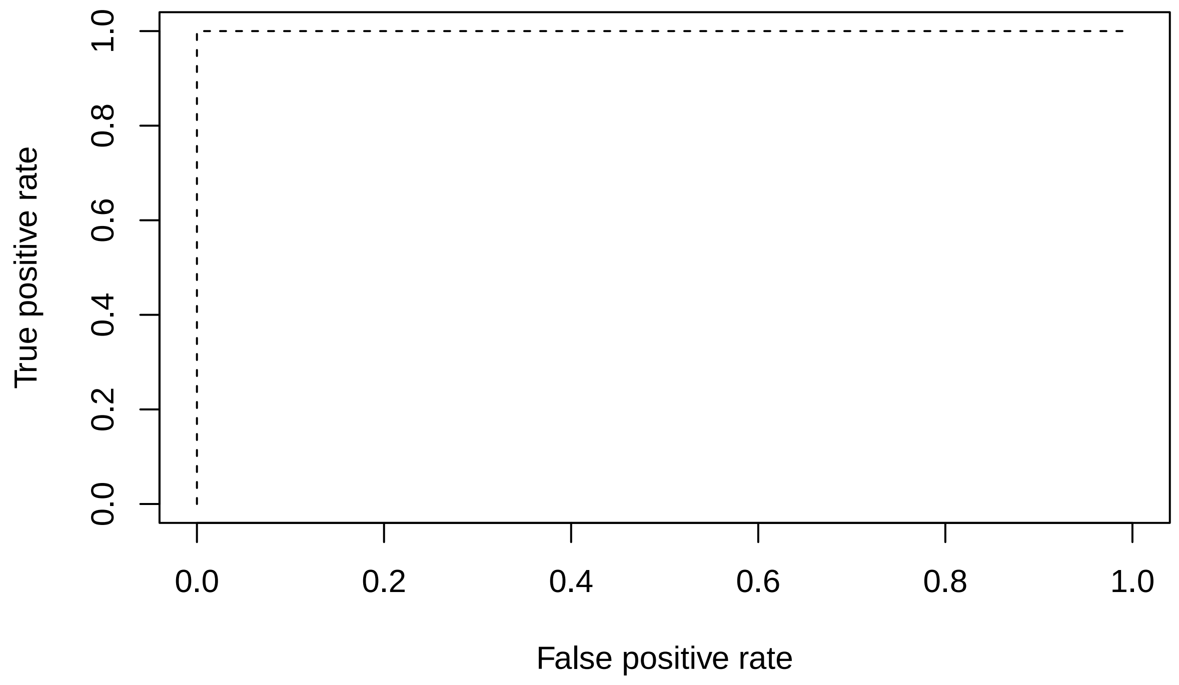
```r
#train using caret
model_bagging = train(
  Failure.binary ~ .,
  data = trnDF,
  method = "treebag",
  trControl = trainControl(method = "cv", number = 10),
  nbagg = 100,
  control = rpart.control(minsplit = 2, cp = 0)
)
model_bagging
```

```
## Bagged CART
##
## 158 samples
## 431 predictors
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 142, 143, 143, 143, 142, 142, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8978186  0.7612896
```

We see that the TP rate for prediction using the training set is at 100% which is most likely caused by the effective normalization.

```
# Compute predicted probabilities on training data
trainPred=predict(model_bagging, trnDF, type = "prob")[,2]

trainPred%>%
 prediction(trnDF$Failure.binary) %>%
  performance(measure = "tpr", x.measure = "fpr")%>%
plot( col = "black", lty = 2)
```

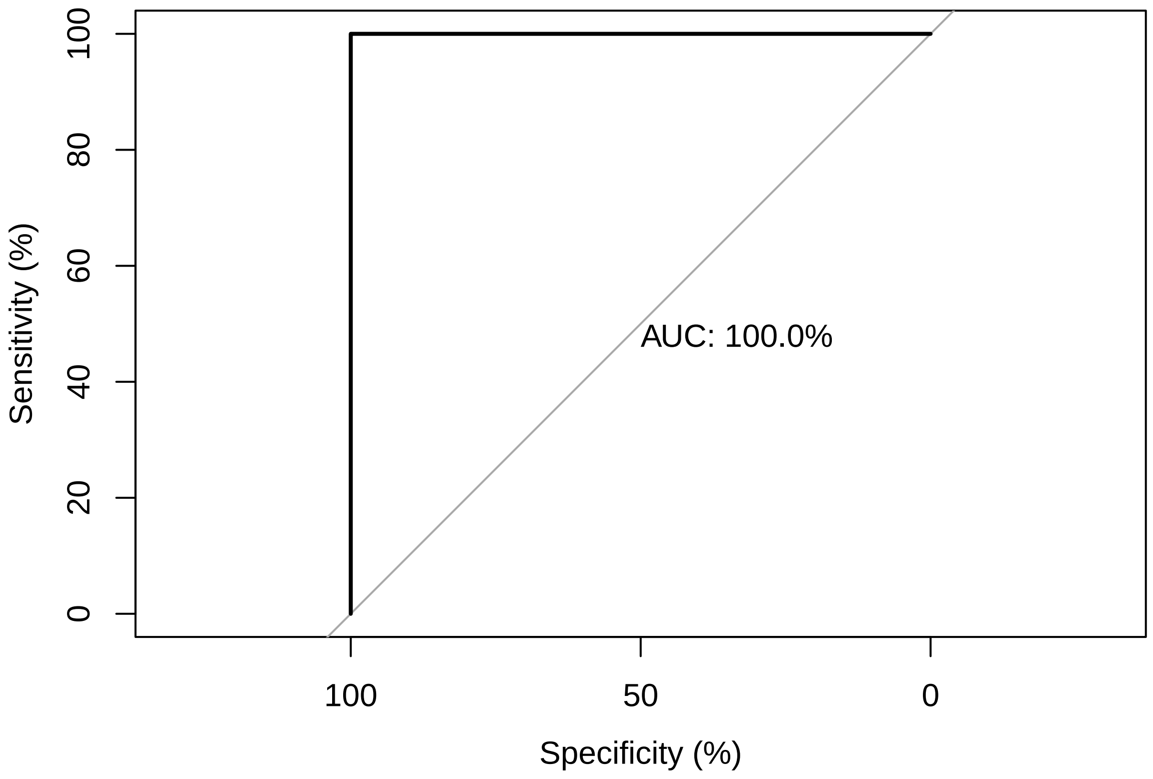


ROC and pROC packages proveds specificity, sensitivity, and an AUC value better than the previous plot

```
# ROC plot for training data
roc( trnDF$Failure.binary ~ trainPred, plot=TRUE, legacy.axes=FALSE,
    percent=TRUE, col="black", lwd=2, print.auc=TRUE)
```

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

```
## Setting direction: controls < cases
```

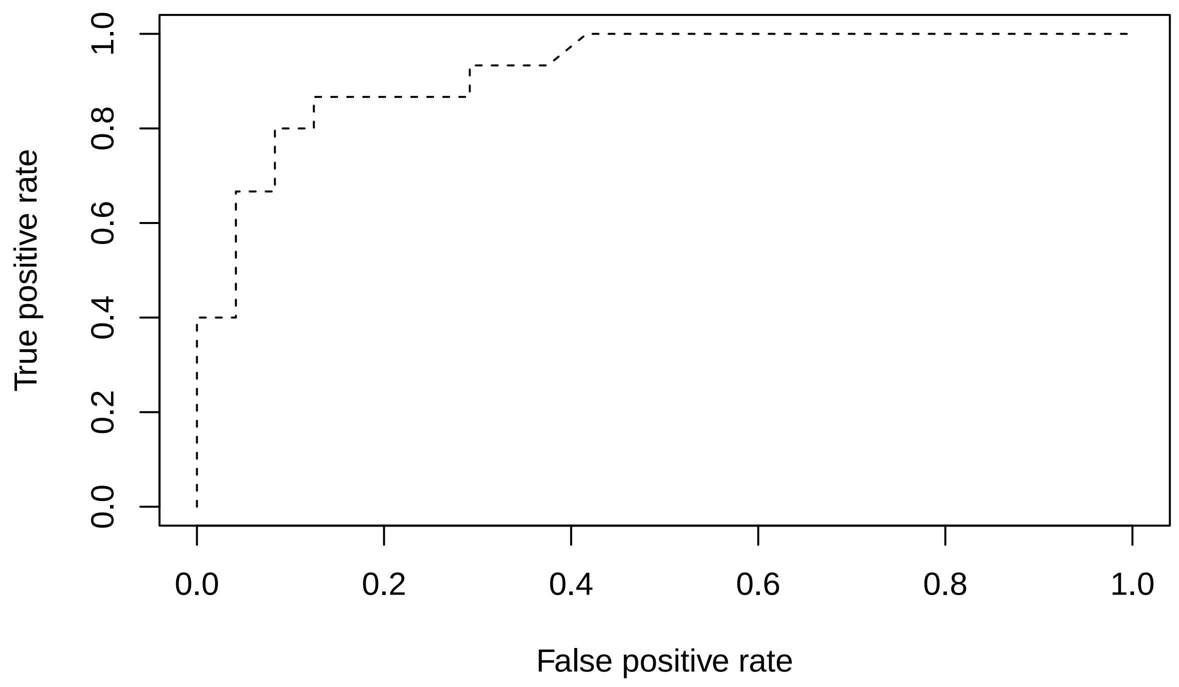


```
##
## Call:
## roc.formula(formula = trnDF$Failure.binary ~ trainPred, plot = TRUE,     legacy.axes = FALSE, percen
##
## Data: trainPred in 106 controls (trnDF$Failure.binary 0) < 52 cases (trnDF$Failure.binary 1).
## Area under the curve: 100%
```

Now testing prediction performance on testing set.

```
# Compute predicted probabilities on training data
testPred = predict(model_bagging, tstDF, type = "prob")[,2]

# Compute AUC metrics for cv_model1,2 and 3
testPred%>%
prediction(tstDF$Failure.binary) %>%
  performance(measure = "tpr", x.measure = "fpr")%>%
plot( col = "black", lty = 2)
```

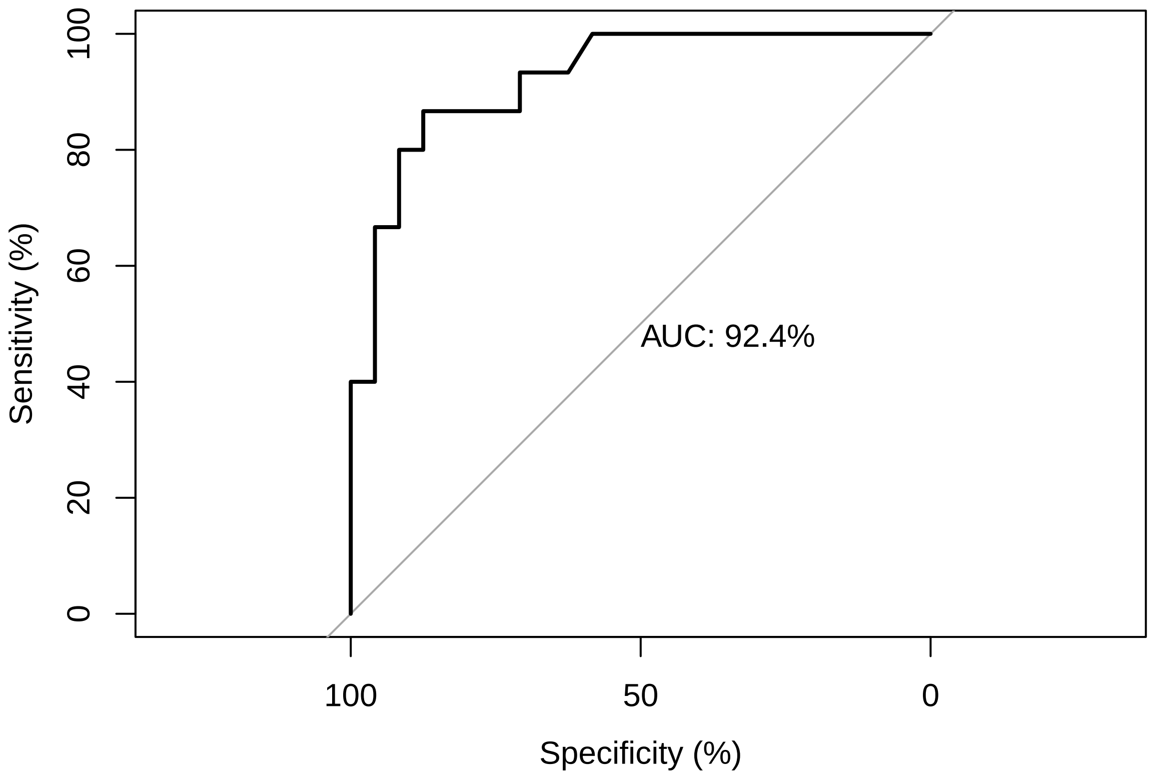


We see below that the model performed well also on the test set which would suggest that we did not overfit as seen in the training set performance.

```
# ROC plot for training data
roc( tstDF$Failure.binary ~ testPred, plot=TRUE, legacy.axes=FALSE,
    percent=TRUE, col="black", lwd=2, print.auc=TRUE)
```

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

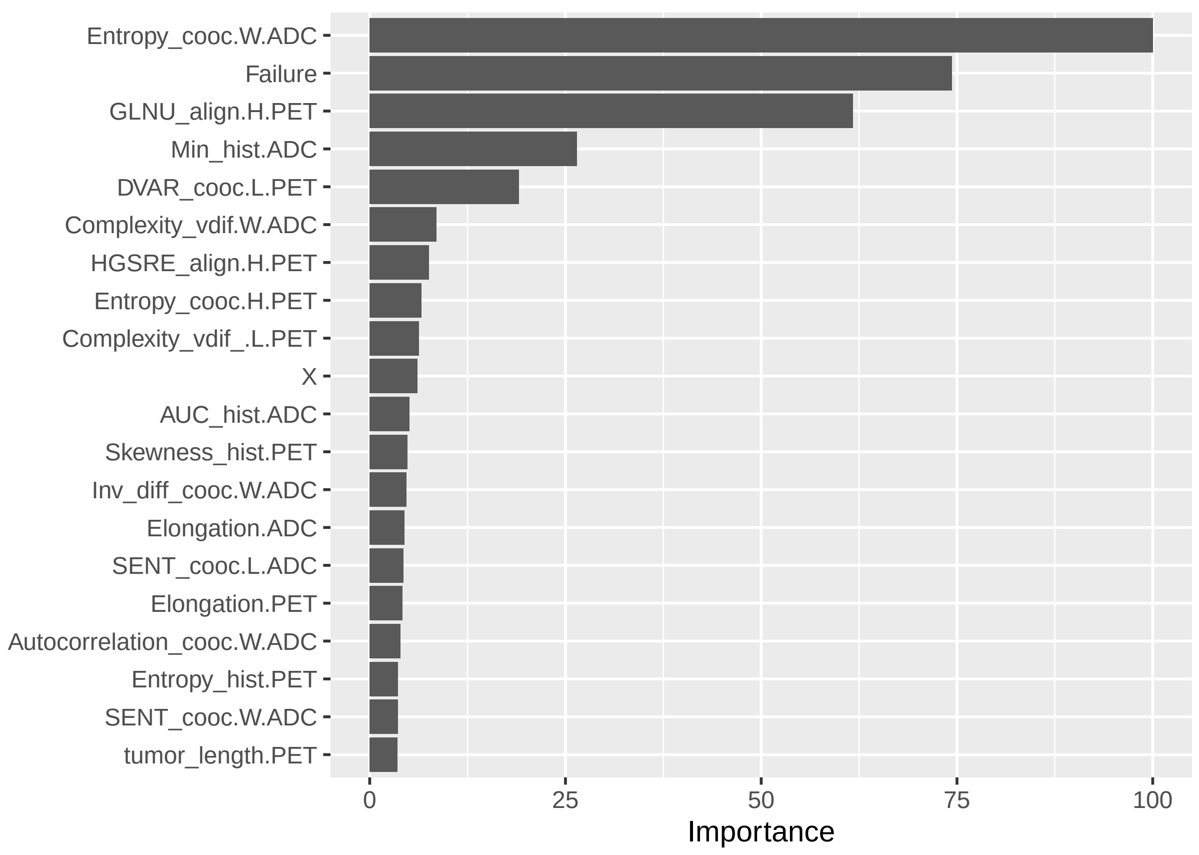
```
## Setting direction: controls < cases
```



```
##
## Call:
## roc.formula(formula = tstDF$Failure.binary ~ testPred, plot = TRUE,     legacy.axes = FALSE, percent
##
## Data: testPred in 24 controls (tstDF$Failure.binary 0) < 15 cases (tstDF$Failure.binary 1).
## Area under the curve: 92.36%
```

Calling vip function from vip package to see Variable Importance for each feature variables and we see that Entropy_cooc.W.ADC has a very high Importance level.

```
vip::vip(model_bagging, num_features = 20)
```


Entropy_cooc.W.ADC
Failure
GLNU_align.H.PET
Min_hist.ADC
DVAR_cooc.L.PET
Complexity_vdif.W.ADC
HGSRE_align.H.PET
Entropy_cooc.H.PET
Complexity_vdif_.L.PET
X
AUC_hist.ADC
Skewness_hist.PET
Inv_diff_cooc.W.ADC
Elongation.ADC
SENT_cooc.L.ADC
Elongation.PET
Autocorrelation_cooc.W.ADC
Entropy_hist.PET
SENT_cooc.W.ADC
tumor_length.PET
0
25
50
75
100
Importance