Model 1 by Christine Pallon

2022-12-14

Creating an ensemble classification model

Loading and pre-processing the data

For our first model, we will build an **ensemble classification model** using three base models with the **radiomics** dataset. First, we load and prepare the dataset.

```
radiomics <- read.csv("radiomics_completedata.csv")


is.null(radiomics)
```

```
## [1] FALSE
```

```
radiomics <- radiomics %>% select(-Failure)

radiomics$Failure.binary <- as.factor(radiomics$Failure.binary)
radiomics$Institution <- as.factor(radiomics$Institution)
radiomics$Failure.binary
```

```
##   [1] 0 1 0 1 0 1 0 0 1 1 1 1 0 1 1 0 1 1 1 1 1 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0 1
##  [38] 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0
##  [75] 0 0 0 0 1 1 1 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0
## [112] 0 0 0 0 0 1 1 0 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 1 0 1 0 0 1 0 0
## [149] 0 1 0 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 1 1 0 0 0
## [186] 0 1 0 0 0 0 0 0 0 0 0 0
## Levels: 0 1
```

```
radiomics$Institution
```

```
##   [1] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [38] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [75] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
## [112] A A A A A A A A B B B B B B B B B B B B B B B B B B B B B B B B B B B B D
## [149] C D C D D C C D C C D D C D D C C D C C D D D C D C D D C D D C C D D C D
## [186] D C C D C C D D D C C D
## Levels: A B C D
```

```
radiomics[,3:430]<-scale(radiomics[,3:430])
```

Now that we've loaded the data, checked for nulls, made sure Failure.binary and Institution are factors, and scaled the dataset, we'll store our features as an object so they're easily accessible when building our models.

```
response <- "Failure.binary"
features <- setdiff(colnames(radiomics), response)
```

Correlation

Now, we get the correlation of the whole dataset except the response and categorical variable. The number of features makes this dataset quite large so we won't call radiomics_correlation, but the correlation for all the features are contained in that object as a dataframe that we can investigate if we want to get a better understanding of our data.

```
radiomics_non_categorical <- radiomics %>% select(-Failure.binary, -Institution)
radiomics_correlation <- as.data.frame(cor(radiomics_non_categorical))
```

Data splitting

Next, we split the data into an **80/20 training/testing split**.

```
set.seed(123)
radiomics_split <- initial_split(radiomics, prop = 0.8, strata = "Failure.binary")
radiomics_train <- training(radiomics_split)
radiomics_test  <- testing(radiomics_split)
```

Since we're working with h2o models, we need to also convert our testing and training sets to h2o objects, as seen in the below code chunk.

```
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         5 hours 24 minutes
##     H2O cluster timezone:       America/Toronto
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.38.0.1
##     H2O cluster version age:    2 months and 25 days
##     H2O cluster name:           H2O_started_from_R_christinep_bkm587
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   6.74 GB
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  8
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     R Version:                  R version 4.2.1 (2022-06-23)
```

```
train.h2o <- as.h2o(radiomics_train)
```

```
##   |                                                                      |
```

```
test.h2o <- as.h2o(radiomics_test)
```

```
##   |                                                                      |
```

Building base learners

Our ensemble model will be made up of three base learners - one GBM, one RF, and one XGBoost.

When buildng these models, it is critical to verify that we've set them up in a way that will allow us to stack them in our ensemble later.

We first need to make sure we use the same training dataset for each model (train.h2o). Then, we need to make sure every model is trained with the name number of CV folds and use the same fold assignment. Finally, we need to make sure keep_cross_validation_predictions is set to TRUE with every model we build.

With that clarified, let's get to building our base learners.

Below, we've made our GBM, RF, and XGBoost models:

```
gbm <- h2o.gbm(
  x = features, y = "Failure.binary", training_frame = train.h2o, ntrees = 5000, learn_rate = 0.01,
  max_depth = 7, min_rows = 5, sample_rate = 0.8, nfolds = 10,
  fold_assignment = "Modulo", keep_cross_validation_predictions = TRUE,
  seed = 123, stopping_rounds = 50, stopping_tolerance = 0
)
```

```
##   |                                                                      |
```

```
rf <- h2o.randomForest(
  x = features, y = "Failure.binary", training_frame = train.h2o, ntrees = 5000, mtries = 20,
  max_depth = 30, min_rows = 1, sample_rate = 0.8, nfolds = 10,
  fold_assignment = "Modulo", keep_cross_validation_predictions = TRUE,
  seed = 123, stopping_rounds = 50,
  stopping_tolerance = 0
)
```

```
##   |                                                                      |
```

```
xgb <- h2o.xgboost(
  x = features, y = "Failure.binary", training_frame = train.h2o, ntrees = 5000, learn_rate = 0.05,
  max_depth = 3, min_rows = 3, sample_rate = 0.8,
  nfolds = 10, fold_assignment = "Modulo",
  keep_cross_validation_predictions = TRUE, seed = 123, stopping_rounds = 50, stopping_tolerance = 0
)
```

```
##   |                                                                      |
```

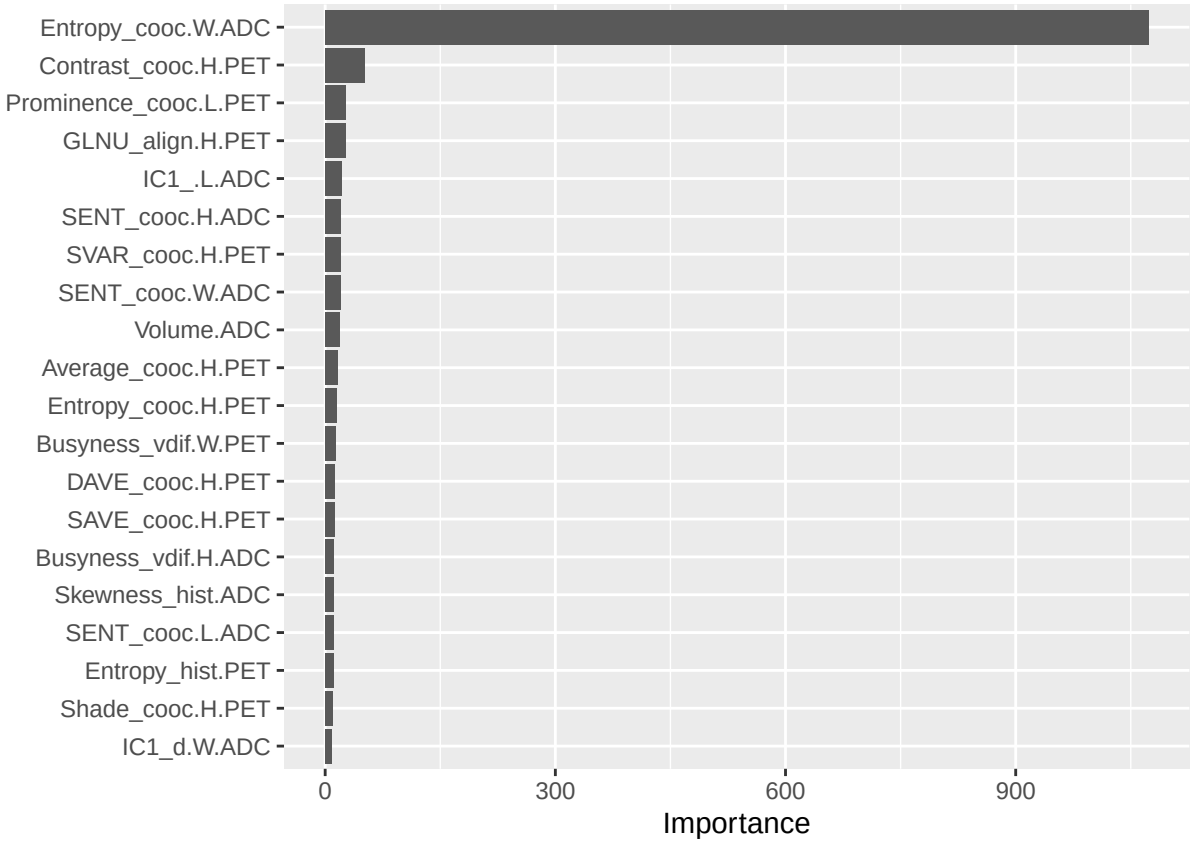
Getting our top 20 important features during training

Next, we'll use vip() to get the top 20 important features during the training phase from each of our base learners.

GBM vip()

The top 20 important features for GBM:

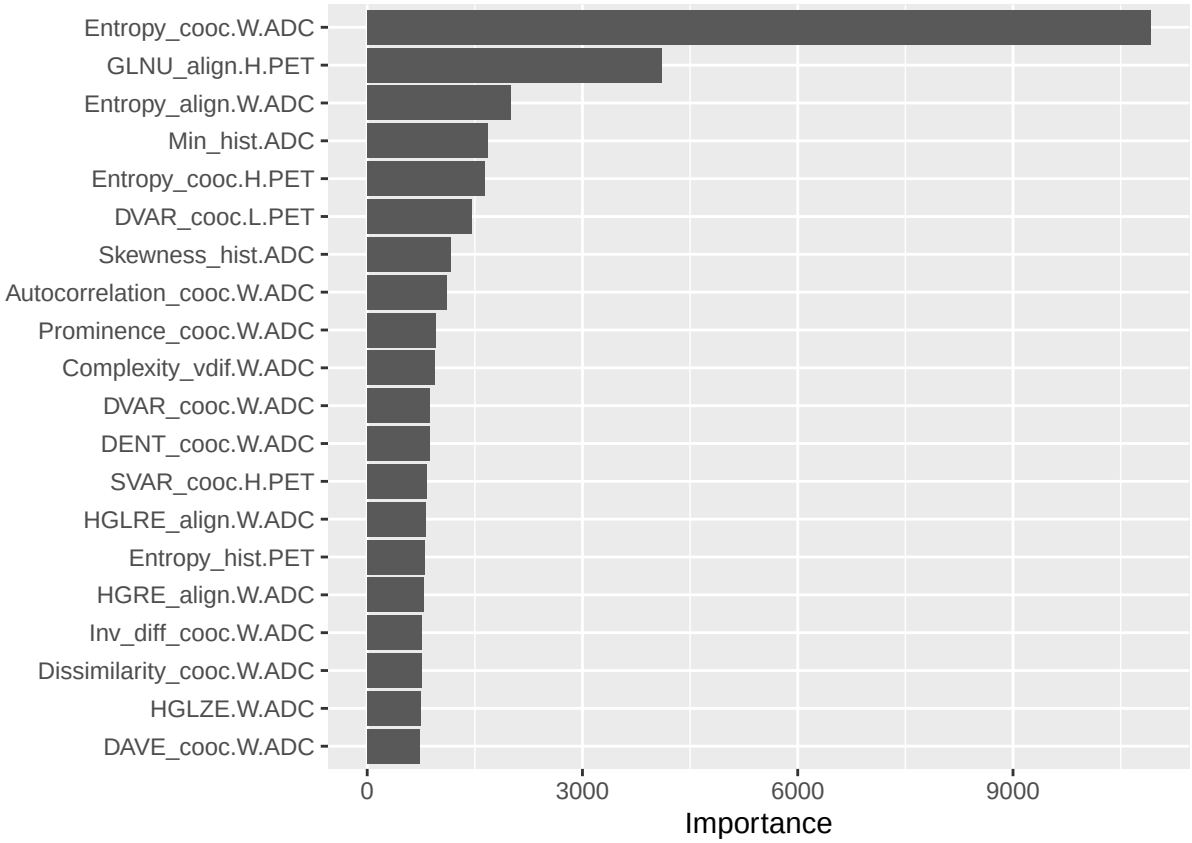
```
vip(gbm, num_features=20)
```



RF vip()

The top 20 important features for RF:

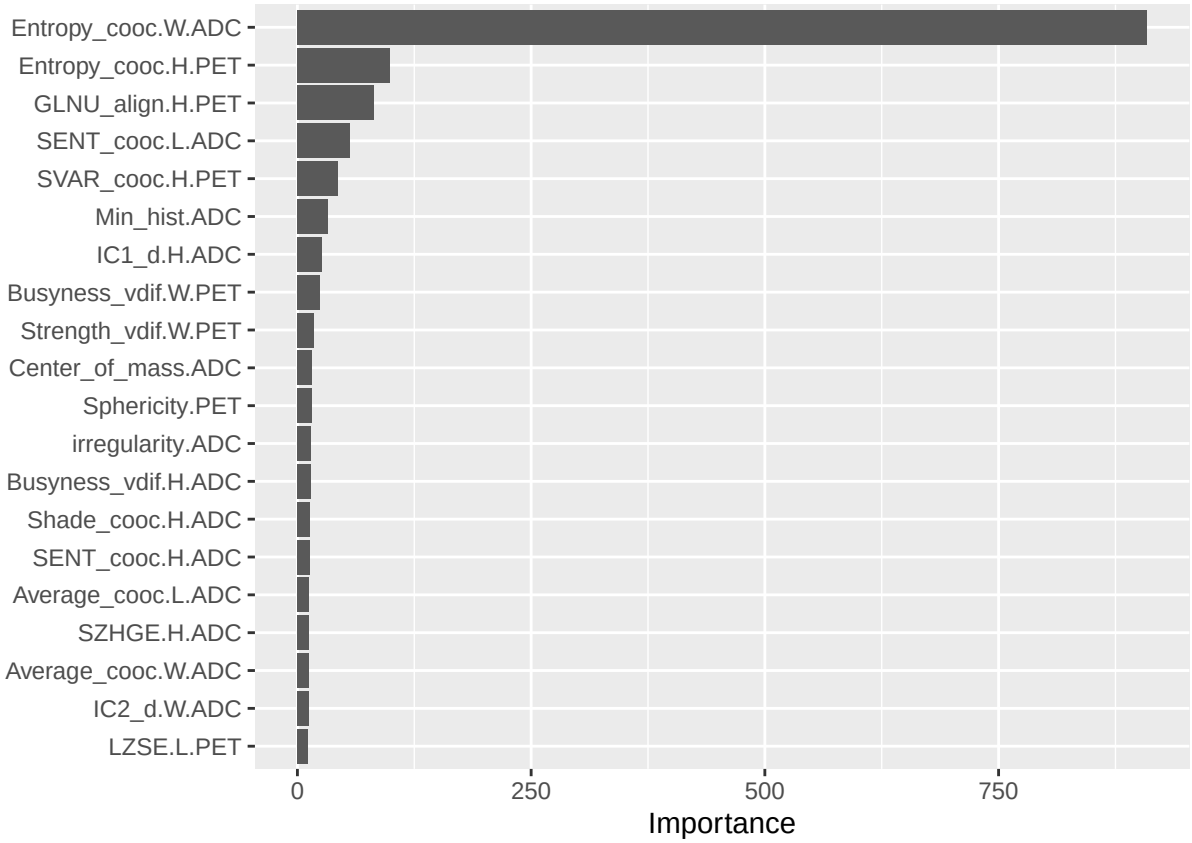
```
vip(rf, num_features=20)
```



XGBoost vip()

The top 20 important features for XGBoost:

```
vip(xgb, num_features=20)
```



Building our ensemble model

Now that we have our base learners, let's stack them into an ensemble using h2o.stackedEnsemble(). This model takes our three base learners and uses them to, hopefully, make the most optimal predictions.

```
ensemble <- h2o.stackedEnsemble(
  x = features, y = "Failure.binary", training_frame = train.h2o, model_id = "ensemble",
  base_models = list(gbm, rf, xgb), metalearner_algorithm = "gbm")
```

```
##   |                                                                              |
```

Getting the AUC values during the training phase

Now that we have all our models, let's compare their AUC values during the training phase.

Getting AUC values of base learners and ensemble model during training phase

```
ensemble_perf_train <- h2o.performance(ensemble, train.h2o)
gbm_perf_train <- h2o.performance(gbm, train.h2o)
rf_perf_train <- h2o.performance(rf, train.h2o)
xgb_perf_train <- h2o.performance(xgb, train.h2o)
```

```
gbm_auc_train <- h2o.auc(gbm_perf_train)
rf_auc_train <- h2o.auc(rf_perf_train)
xgb_auc_train <- h2o.auc(xgb_perf_train)
ensemble_auc_train <- h2o.auc(ensemble_perf_train)

print(paste0("The AUC value for the GBM model is ", gbm_auc_train))
```

```
## [1] "The AUC value for the GBM model is 1"
```

```
print(paste0("The AUC value for the RF model is ", rf_auc_train))
```

```
## [1] "The AUC value for the RF model is 1"
```

```
print(paste0("The AUC value for the XGBoost model is ", xgb_auc_train))
```

```
## [1] "The AUC value for the XGBoost model is 1"
```

```
print(paste0("The AUC value for the ensemble model is ", ensemble_auc_train))
```

```
## [1] "The AUC value for the ensemble model is 1"
```

Looks like our models are doing a great job of classifying our data in the training phase, which isn't suprising. But the real test is how the models do with classifying the testing datat.

Getting AUC values using the testing data

Now let's see how our individual base learners perform on the testing data, and how their performances compare to the ensemble model's performance.

```
ensemble_perf_test <- h2o.performance(ensemble, test.h2o)
gbm_perf_test <- h2o.performance(gbm, test.h2o)
rf_perf_test <- h2o.performance(rf, test.h2o)
xgb_perf_test <- h2o.performance(xgb, test.h2o)


gbm_auc_test <- h2o.auc(gbm_perf_test)
rf_auc_test <- h2o.auc(rf_perf_test)
xgb_auc_test <- h2o.auc(xgb_perf_test)
ensemble_auc_test <- h2o.auc(ensemble_perf_test)

print(paste0("The AUC value for the GBM model is ", round(gbm_auc_test, digits = 3)))
```

```
## [1] "The AUC value for the GBM model is 0.918"
```

```
print(paste0("The AUC value for the RF model is ", round(rf_auc_test, digits = 3)))
```

```
## [1] "The AUC value for the RF model is 0.843"
```

```
print(paste0("The AUC value for the XGBoost model is ", round(xgb_auc_test, digits = 3)))
```

```
## [1] "The AUC value for the XGBoost model is 0.942"
```

```
print(paste0("The AUC value for the ensemble model is ", round(ensemble_auc_test, digits = 3)))
```

```
## [1] "The AUC value for the ensemble model is 0.931"
```

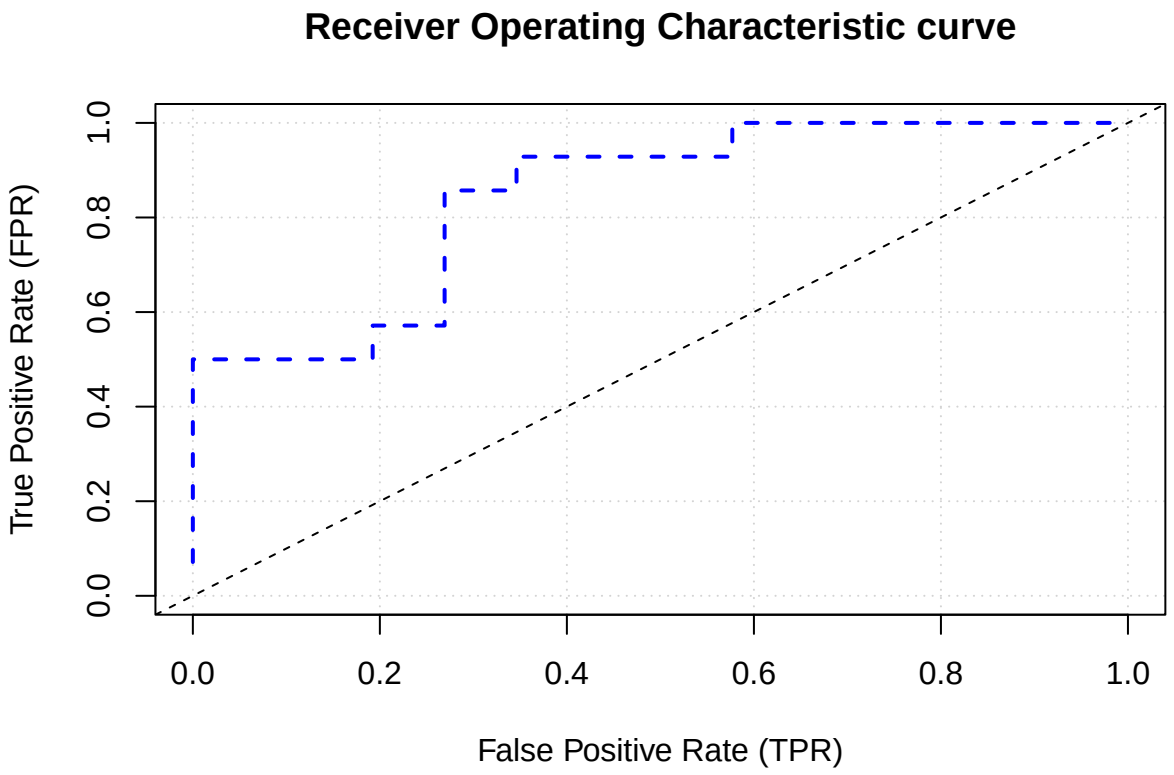
All of our models are performing fairly well on the testing data, including our ensemble. However, among our base learners, our RF model is performing noticeably worse on the testing data compared to GBM and XGBoost.

ROC plots

While we're at it, before we finish, let's take a look at the ROC graphs for each base learner and our ensemble. These graphs allow us to see a visualization of the model's True Positive Rate vs. its False Positive Rate.

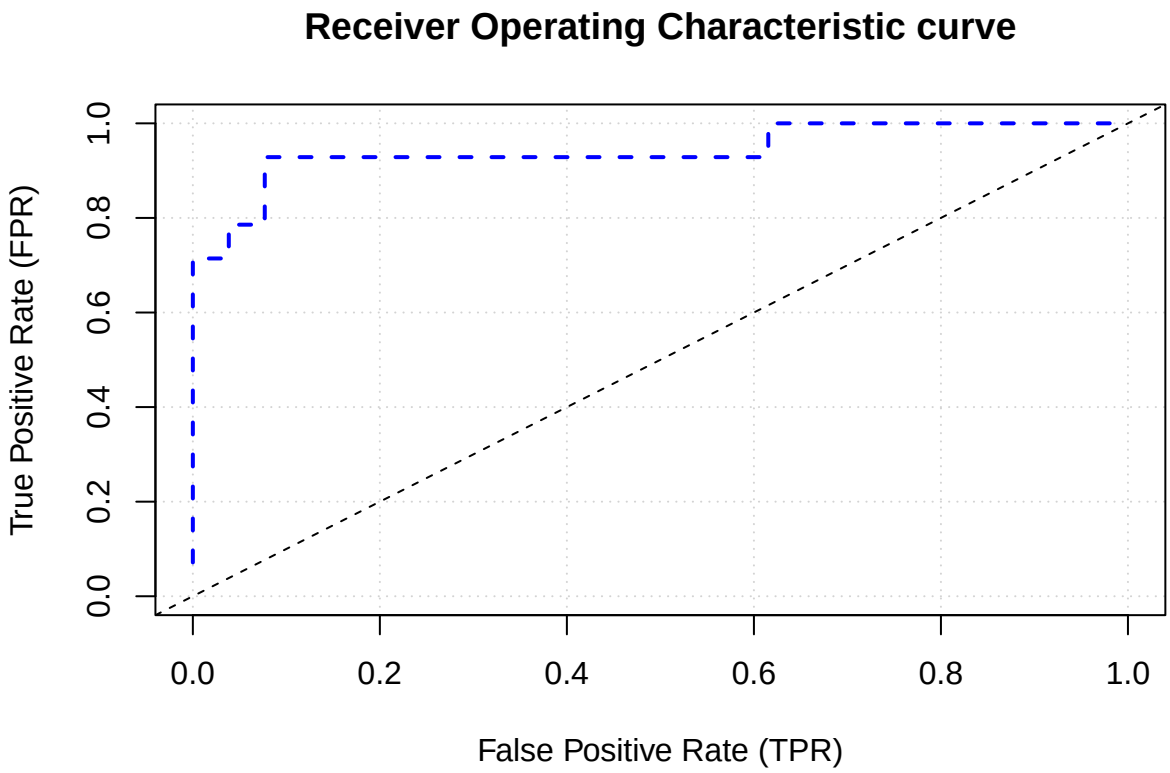
RF ROC plot

```
plot(rf_perf_test, type = "roc")
```



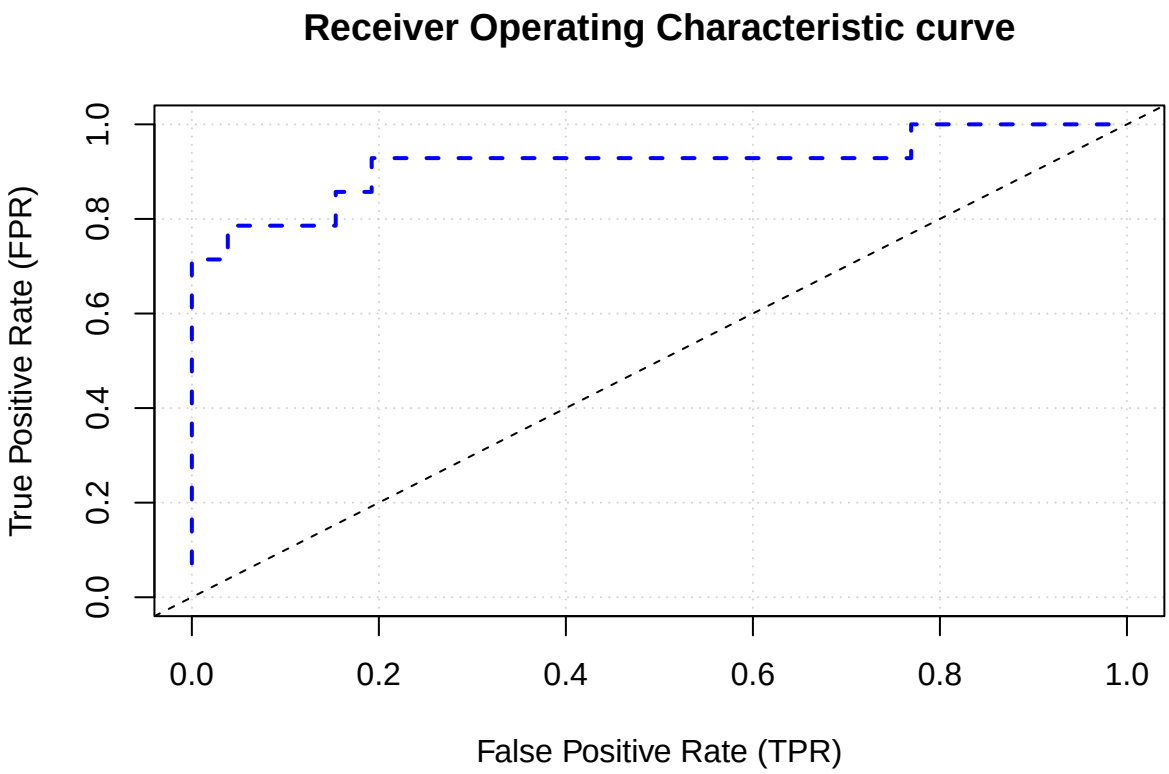
XGBoost ROC plot

```
plot(xgb_perf_test, type = "roc")
```



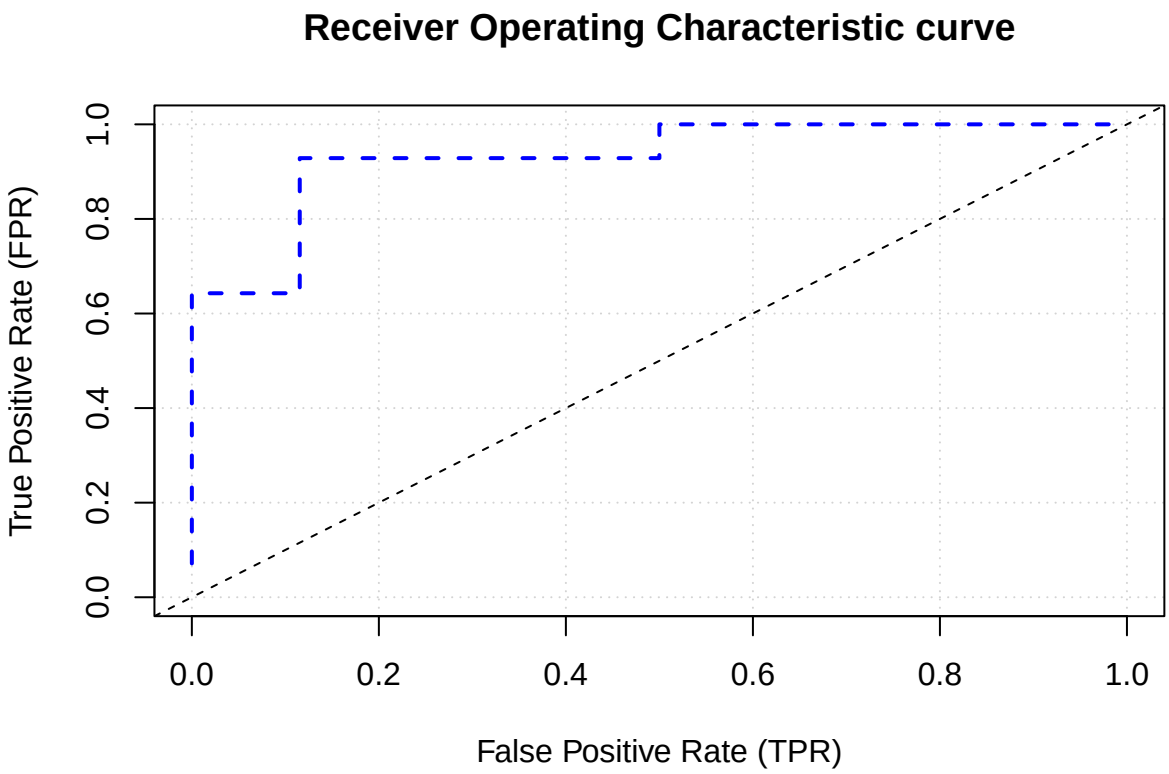
GBM ROC plot

```
plot(gbm_perf_test, type = "roc")
```



Ensemble ROC plot

```
plot(ensemble_perf_test, type = "roc")
```


Receiver Operating Characteristic curve
True Positive Rate (FPR)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (TPR)