FastCoV19: Machine Learning solution for COVID 19 from Blood Test

Huy Huynh, Nhi Le, Andy Tran, Brian Le

10/17/2020

```r
library(png)
library(grid)
img <- readPNG("Project_Title.png")
 grid.raster(img)
```

```r
#Install Packages
#install.packages("ggthemes")
#install.packages("viridis")
#install.packages("e1071")
#install.packages("png")
#install.packages("grid")
```

FastCoV19: Machine Learning solution for COVID 19 from Blood Test"

Objectives

This project is aim to predict confirmed COVID-19 casess based on laboratory result from anual physical exams.

We consider whether it be possible to predict the test result for COVID 19 (either positive or negative) based on the result of blood test?

Data Set

We retrieve the dataset from patients seen at the Hospital Israelita Albert Einstein in São Paulo, Brazil,who perform the COVID 19 test and additional laboratory tests.

```
# load aux functions
source("00-Funcs.R")

#load input dataset
library("readxl");
data <- as.data.frame(read_excel("dataset.xlsx"), stringAsFactors=F)
```

The dataset contains 109 variables (predictors), a Patient ID and one target outcome variable, which indicates whether the patient tested positive/negative for SARS-Cov-2.

This dataset has 109 variables, a Patient ID and one target outcome variable showing the result (positive/negative) of COVID 19.

Data Cleaning

Data cleaning procedures consist of: Making variable names syntactically valid, Replacing column values that should be empty for NA, Convert string categorical values to factors, Convert the variable **Urine...pH** to integer

```
# make variable names syntactically valid
names(data) <- make.names(names(data))
data$Patient.ID <- NULL

# Replace column values that should be empty for NA
data[data=='Não Realizado'] <- NA
data[data=='not_done'] <- NA
data[data=='<1000'] <- 500

data$Urine...Leukocytes <- as.integer(data$Urine...Leukocytes)
data$Urine...pH <- as.integer(data$Urine...pH)

# convert string values to factors
ind <- sapply(data, is.character)
data[ind] <- lapply(data[ind], factor)

data$Lipase.dosage <- as.factor(data$Lipase.dosage)
```

The outcome variable **SARS.Cov.2.exam.result** is a binary by which we convert such that $SARS.Cov.2.exam.result = 1$ for being positive and $SARS.Cov.2.exam.result = 0$ for being negative with COVID 19.

```
outcome.var<-"SARS.Cov.2.exam.result"
data[, outcome.var] <- as.integer(data[, outcome.var]) - 1
```

However, there are too many missing data points (>= 95%); we decided to remove them along with poor sample. But we choose to keep negative samples that have at least 10 variables with data points available.

```
data.deleted<-data
data.size<-nrow(data)
not.na.pct <- 0.05
data <- delete.na(data, n = data.size * not.na.pct, is.row = FALSE)


data.pos <- data[data$SARS.Cov.2.exam.result==1,]
data.neg <-  data[data$SARS.Cov.2.exam.result==0,]

### delete poor samples
min.non.na.vars <- 10
data.neg <- delete.na(data.neg, n = min.non.na.vars)

data <- rbind(data.pos, data.neg)
```

Hence, We have removed a lot of variables showing as below:

```
print(setdiff(names(data.deleted), names(data)))
```

```
##  [1] "Serum.Glucose"
##  [2] "Mycoplasma.pneumoniae"
##  [3] "Alanine.transaminase"
##  [4] "Aspartate.transaminase"
##  [5] "Gamma.glutamyltransferase."
##  [6] "Total.Bilirubin"
##  [7] "Direct.Bilirubin"
##  [8] "Indirect.Bilirubin"
##  [9] "Alkaline.phosphatase"
## [10] "Ionized.calcium."
## [11] "Magnesium"
## [12] "pCO2..venous.blood.gas.analysis."
## [13] "Hb.saturation..venous.blood.gas.analysis."
## [14] "Base.excess..venous.blood.gas.analysis."
## [15] "pO2..venous.blood.gas.analysis."
## [16] "Fio2..venous.blood.gas.analysis."
## [17] "Total.CO2..venous.blood.gas.analysis."
## [18] "pH..venous.blood.gas.analysis."
## [19] "HCO3..venous.blood.gas.analysis."
## [20] "Rods.."
## [21] "Segmented"
## [22] "Promyelocytes"
## [23] "Metamyelocytes"
## [24] "Myelocytes"
## [25] "Myeloblasts"
## [26] "Urine...Esterase"
## [27] "Urine...Aspect"
## [28] "Urine...pH"
## [29] "Urine...Hemoglobin"
## [30] "Urine...Bile.pigments"
## [31] "Urine...Ketone.Bodies"
## [32] "Urine...Nitrite"
## [33] "Urine...Density"
## [34] "Urine...Urobilinogen"
## [35] "Urine...Protein"
## [36] "Urine...Sugar"
## [37] "Urine...Leukocytes"
## [38] "Urine...Crystals"
## [39] "Urine...Red.blood.cells"
## [40] "Urine...Hyaline.cylinders"
## [41] "Urine...Granular.cylinders"
## [42] "Urine...Yeasts"
## [43] "Urine...Color"
## [44] "Partial.thromboplastin.time..PTT.."
## [45] "Relationship..Patient.Normal."
## [46] "International.normalized.ratio..INR."
## [47] "Lactic.Dehydrogenase"
## [48] "Prothrombin.time..PT...Activity"
## [49] "Vitamin.B12"
## [50] "Creatine.phosphokinase..CPK.."
## [51] "Ferritin"
## [52] "Arterial.Lactic.Acid"
## [53] "Lipase.dosage"
## [54] "D.Dimer"
## [55] "Albumin"
## [56] "Hb.saturation..arterial.blood.gases."
## [57] "pCO2..arterial.blood.gas.analysis."
## [58] "Base.excess..arterial.blood.gas.analysis."
## [59] "pH..arterial.blood.gas.analysis."
## [60] "Total.CO2..arterial.blood.gas.analysis."
## [61] "HCO3..arterial.blood.gas.analysis."
## [62] "pO2..arterial.blood.gas.analysis."
## [63] "Arteiral.Fio2"
## [64] "Phosphor"
## [65] "ctO2..arterial.blood.gas.analysis."
```

And we keep the remaning variables for the prediction model:

```r
print(names(data))
```

```
##  [1] "Patient.age.quantile"
##  [2] "SARS.Cov.2.exam.result"
##  [3] "Patient.addmited.to.regular.ward..1.yes..0.no."
##  [4] "Patient.addmited.to.semi.intensive.unit..1.yes..0.no."
##  [5] "Patient.addmited.to.intensive.care.unit..1.yes..0.no."
##  [6] "Hematocrit"
##  [7] "Hemoglobin"
##  [8] "Platelets"
##  [9] "Mean.platelet.volume"
## [10] "Red.blood.Cells"
## [11] "Lymphocytes"
## [12] "Mean.corpuscular.hemoglobin.concentration..MCHC."
## [13] "Leukocytes"
## [14] "Basophils"
## [15] "Mean.corpuscular.hemoglobin..MCH."
## [16] "Eosinophils"
## [17] "Mean.corpuscular.volume..MCV."
## [18] "Monocytes"
## [19] "Red.blood.cell.distribution.width..RDW."
## [20] "Respiratory.Syncytial.Virus"
## [21] "Influenza.A"
## [22] "Influenza.B"
## [23] "Parainfluenza.1"
## [24] "CoronavirusNL63"
## [25] "Rhinovirus.Enterovirus"
## [26] "Coronavirus.HKU1"
## [27] "Parainfluenza.3"
## [28] "Chlamydophila.pneumoniae"
## [29] "Adenovirus"
## [30] "Parainfluenza.4"
## [31] "Coronavirus229E"
## [32] "CoronavirusOC43"
## [33] "Inf.A.H1N1.2009"
## [34] "Bordetella.pertussis"
## [35] "Metapneumovirus"
## [36] "Parainfluenza.2"
## [37] "Neutrophils"
## [38] "Urea"
## [39] "Proteina.C.reativa.mg.dL"
## [40] "Creatinine"
## [41] "Potassium"
## [42] "Sodium"
## [43] "Influenza.B..rapid.test"
## [44] "Influenza.A..rapid.test"
## [45] "Strepto.A"
```

Predictive Analysis

Model Training

To predict the likelihood that a patient is infected with the COVID 19, we split the dataset randomly into training and testing tests in a train-to-test split ratio of 4/5.

```r
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
set.seed(10^7)
SPLIT.RATIO <- 4/5
train.index <- createDataPartition(data$SARS.Cov.2.exam.result, p = SPLIT.RATIO , list = FALSE)
train <- data[train.index,]
test <- data[-train.index,]
```

We train a GBM-Gradient Boosting Machine- to produce a prediction model in the form of an ensemble of weak prediction models, typically decision treesmodel using the remaining dataset variables as predictors. By defining a relatively high bag fraction, this introduces randomnesses into the model fit and reduces overfitting.

```r
train.features <- setdiff(names(train), c(outcome.var, "Patient.ID"))
myformula = as.formula(paste0(outcome.var," ~ ", paste0(train.features, collapse="+")))

BAG.FRACTION <- 0.8
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```r
gbm.model = gbm(myformula, data = train,
                n.trees = 500 ,
                bag.fraction = BAG.FRACTION,
                verbose=FALSE)
```

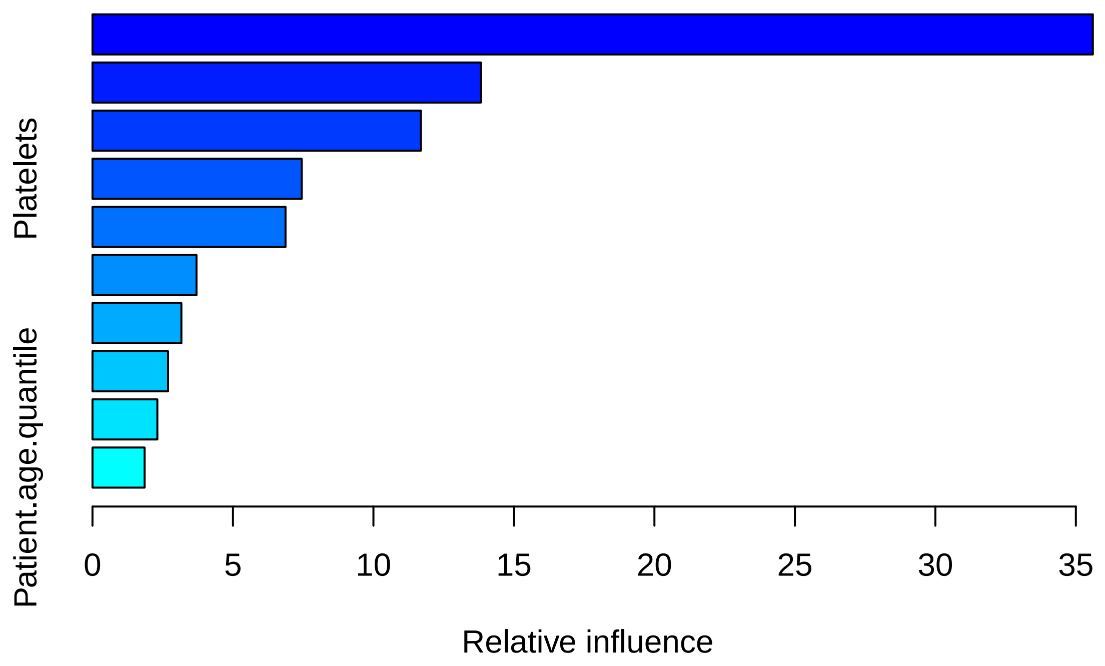
```
## Distribution not specified, assuming bernoulli ...
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution, :
## variable 35: Parainfluenza.2 has no variation.
```

Model Interpretability

The model Interpretability is evaluated by looking at the relative influence of the top 10 most important variables that are normalized and based on the number of times a variable is selected for tree splitting, weighted by the improvement to the model as a result of each split, and averaged over all trees.

```r
model.summary<-summary(gbm.model, cBars=10)
```



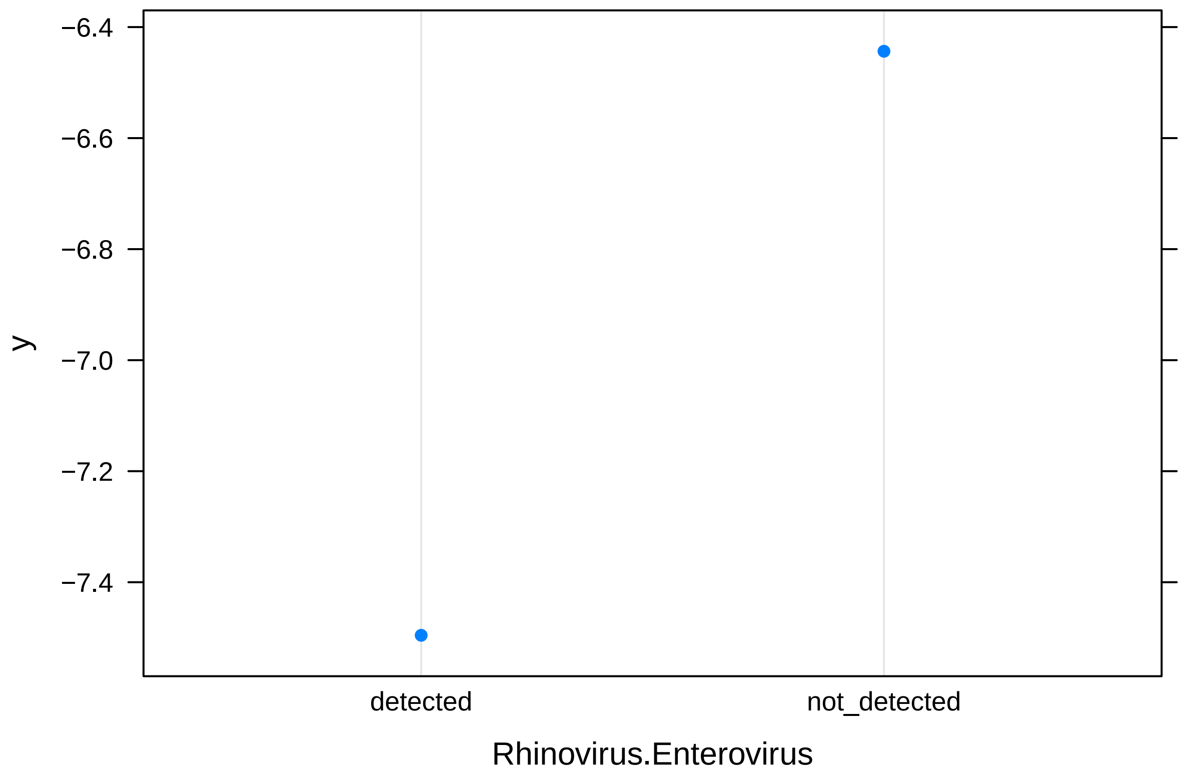
```r
print(model.summary[1:15,])
```

```
##                                                                                    var
## Rhinovirus.Enterovirus                                          Rhinovirus.Enterovirus
## Respiratory.Syncytial.Virus                                Respiratory.Syncytial.Virus
## Leukocytes                                                                  Leukocytes
## Platelets                                                                    Platelets
## Inf.A.H1N1.2009                                                        Inf.A.H1N1.2009
## Monocytes                                                                    Monocytes
## Patient.addmited.to.regular.ward..1.yes..0.no. Patient.addmited.to.regular.ward..1.yes..0.no.
## Eosinophils                                                                Eosinophils
## Influenza.B                                                                Influenza.B
## Patient.age.quantile                                              Patient.age.quantile
## Red.blood.Cells                                                        Red.blood.Cells
## Proteina.C.reativa.mg.dL                                      Proteina.C.reativa.mg.dL
## Sodium                                                                          Sodium
## Urea                                                                              Urea
## Mean.corpuscular.volume..MCV.                            Mean.corpuscular.volume..MCV.
##                                                    rel.inf
## Rhinovirus.Enterovirus                          35.6047711
## Respiratory.Syncytial.Virus                     13.8232937
## Leukocytes                                      11.6870221
## Platelets                                        7.4455164
## Inf.A.H1N1.2009                                  6.8708578
## Monocytes                                        3.7019542
## Patient.addmited.to.regular.ward..1.yes..0.no.  3.1646994
## Eosinophils                                      2.6903773
## Influenza.B                                      2.3094928
## Patient.age.quantile                             1.8545524
## Red.blood.Cells                                  1.1346368
## Proteina.C.reativa.mg.dL                         1.0127916
## Sodium                                           0.9198511
## Urea                                             0.7646602
## Mean.corpuscular.volume..MCV.                    0.7561461
```

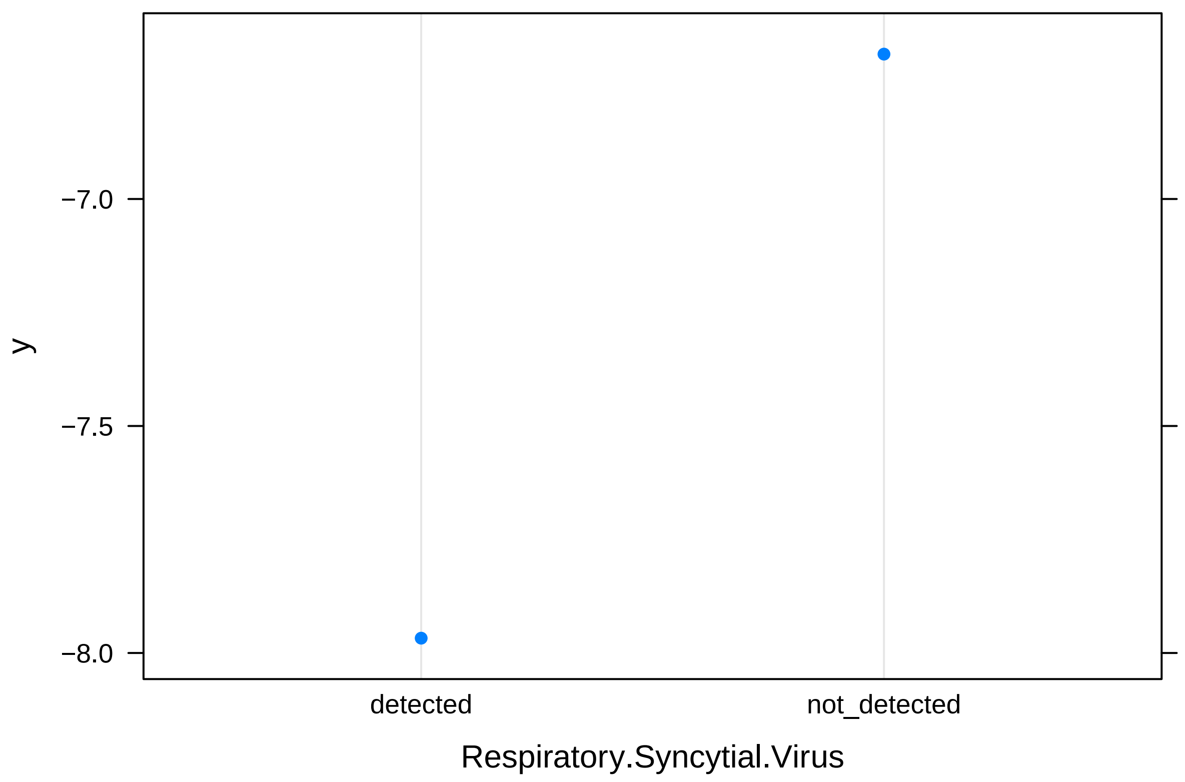
We also analyze the conditional probability plots of the top 5 most important variables below, where the x-axis represents the predictor and the y-axis represents the likelihood of infection .

```
lapply(as.character(model.summary$var[1:5]), plot.gbm, x=gbm.model)
```

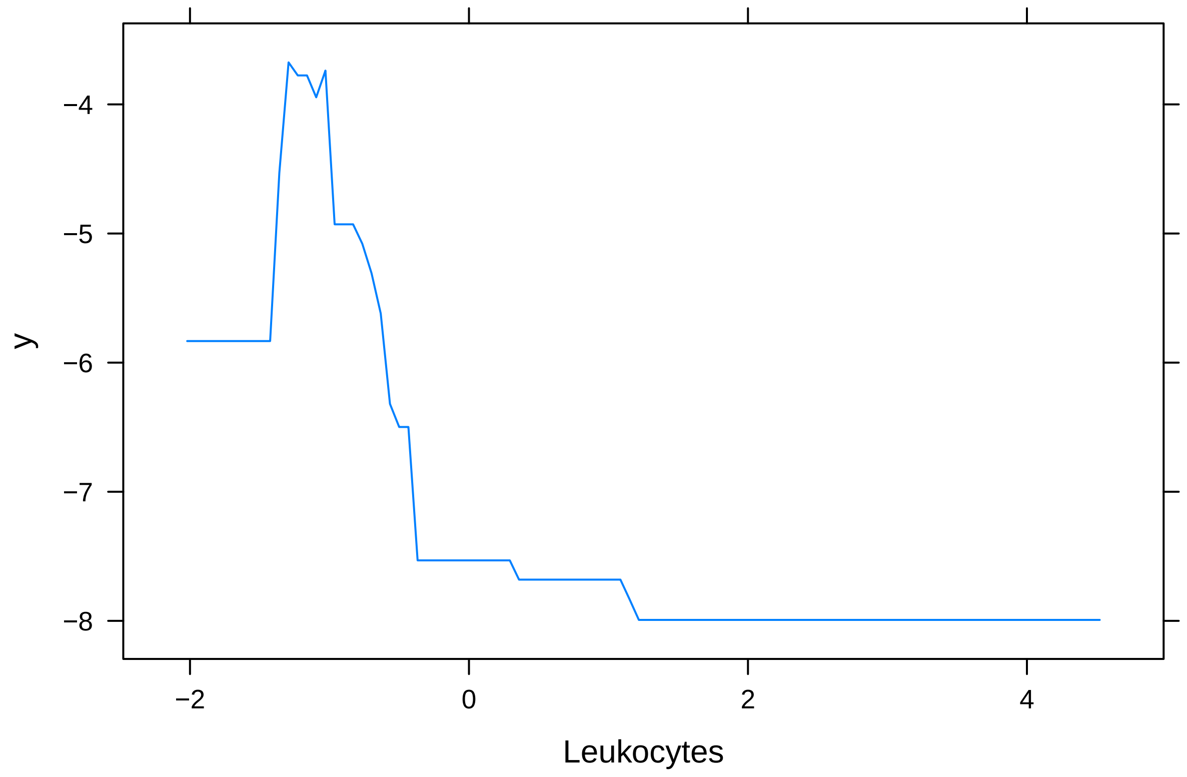
```
## [[1]]
```



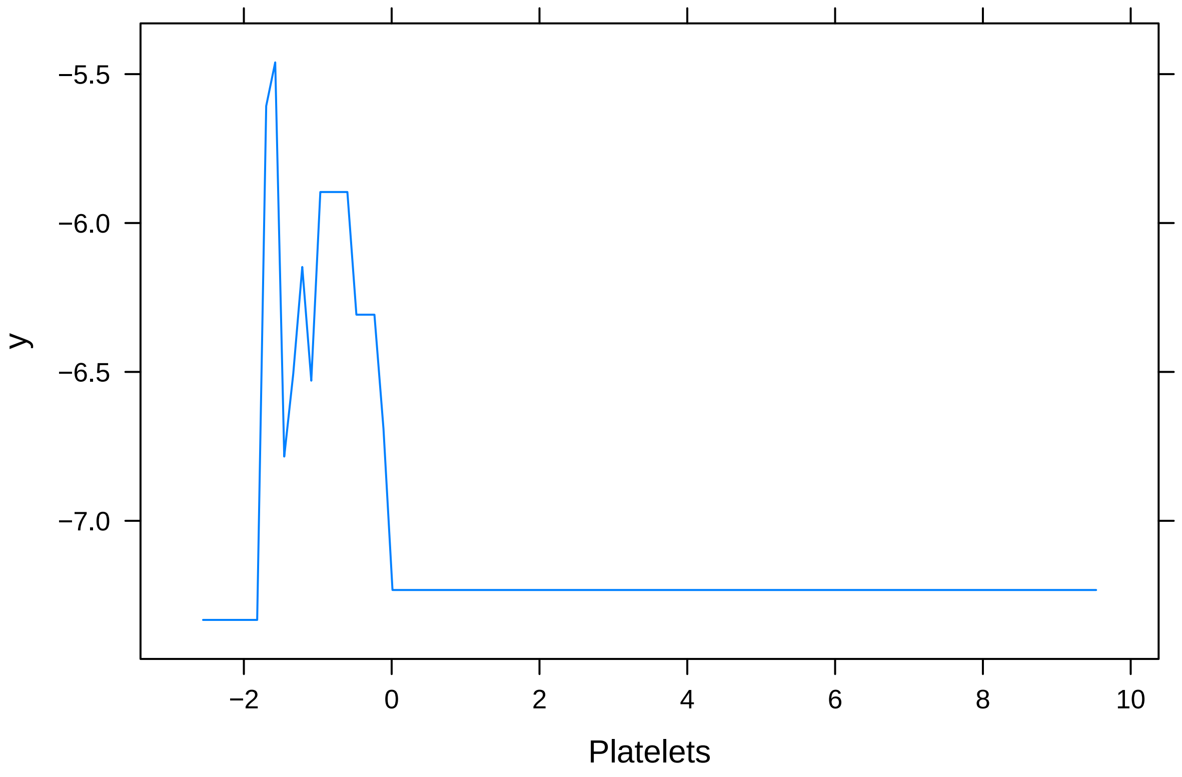
```
##
## [[2]]
```


−7.0
−7.5
−8.0
y
detected
not_detected
Respiratory.Syncytial.Virus

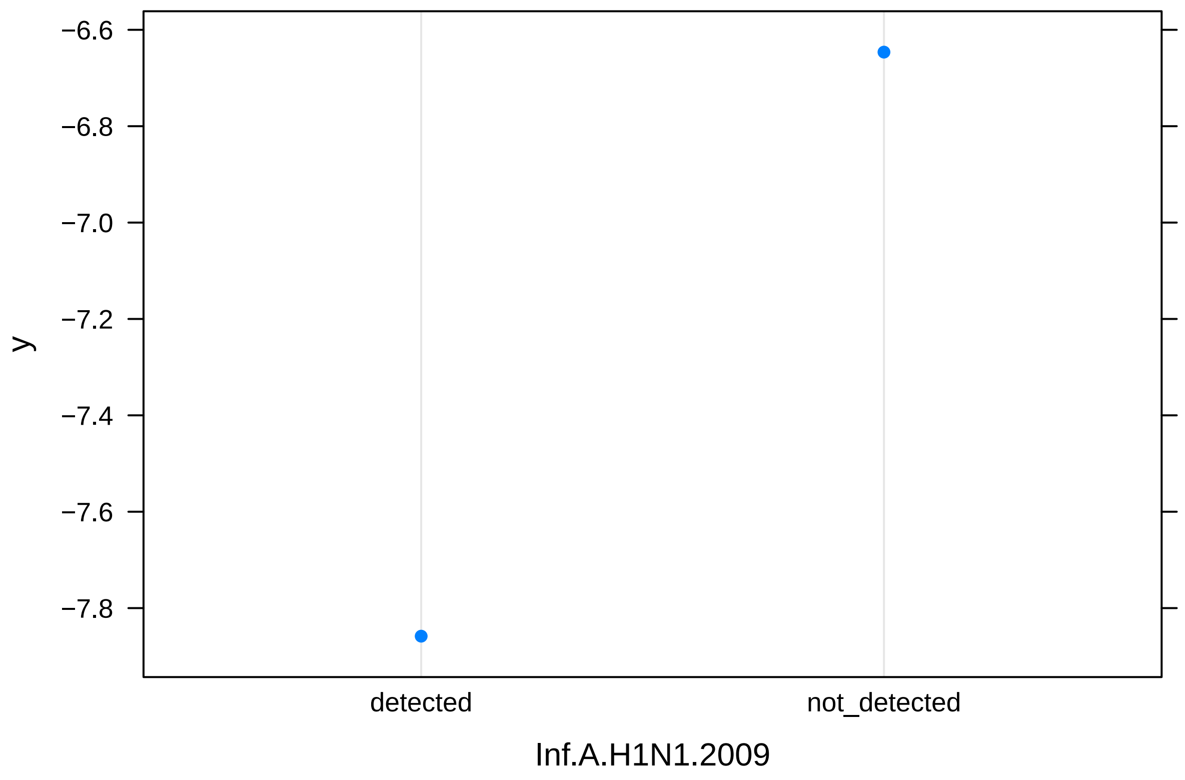
```
##
## [[3]]
```



```
##
## [[4]]
```



```
##
## [[5]]
```



We observe the following:

- Patients are more likely to test positive for COVID 19 when Rhinovirus.Enterovirus, Influenza.B or Inf.A.H1N1.2009 are not detected
- Patients with low Leukocytes or Platelets are more likely to test positive for COVID 19

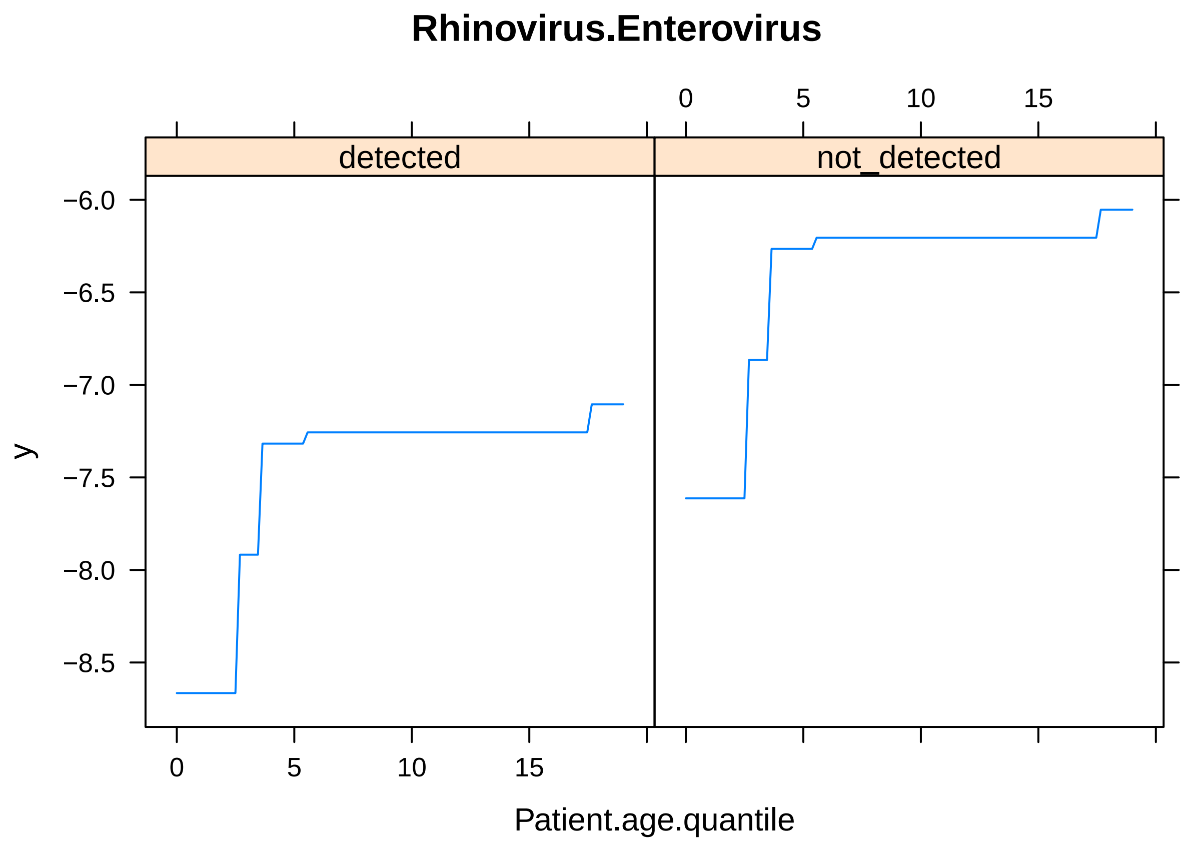
Age is widely discussed as a leading indicator of severe COVID-19 cases. Therefore, we analyze the correlationship between the variable *age_quantile* and the top 5 most important variables discussed previously.

We observe that a patient's age quantile can increase the likelihood of COVID 19 infection regarding top 5 variables.

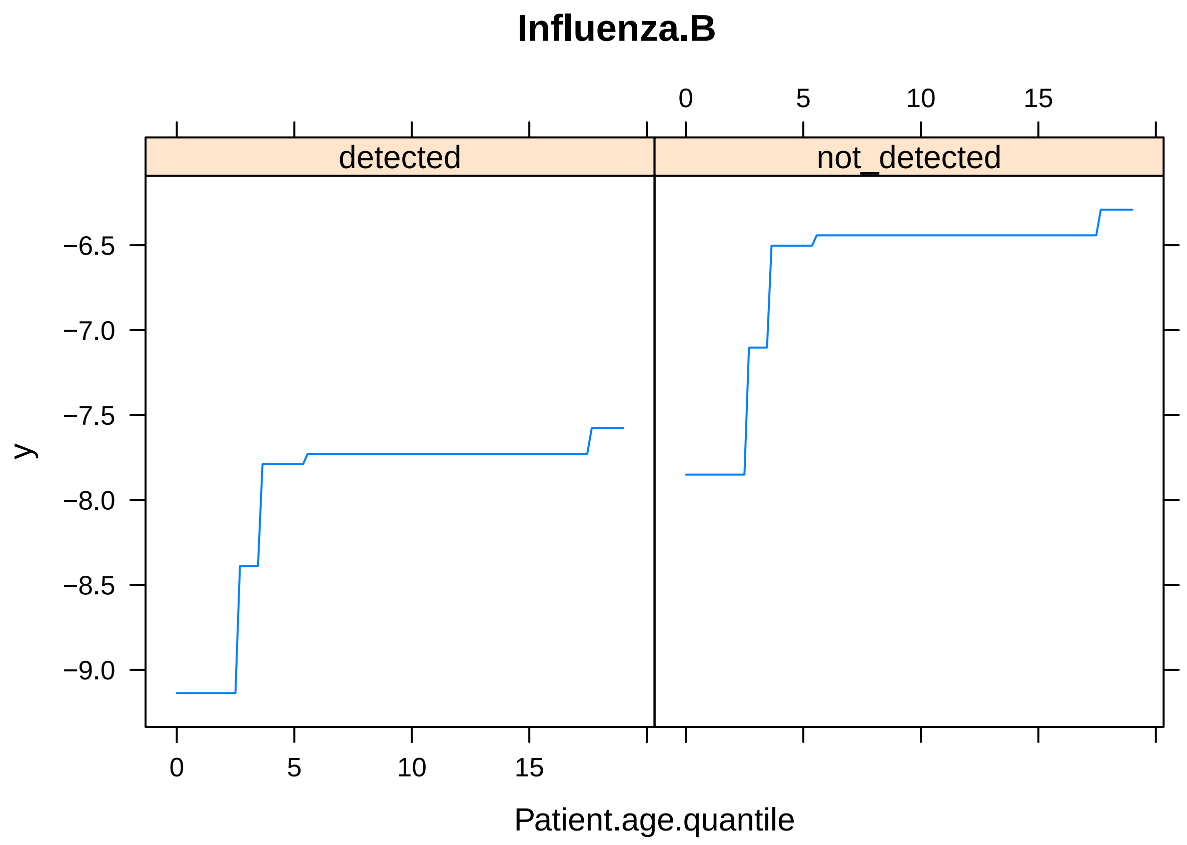
```
library(viridis)
```

```
## Loading required package: viridisLite
```

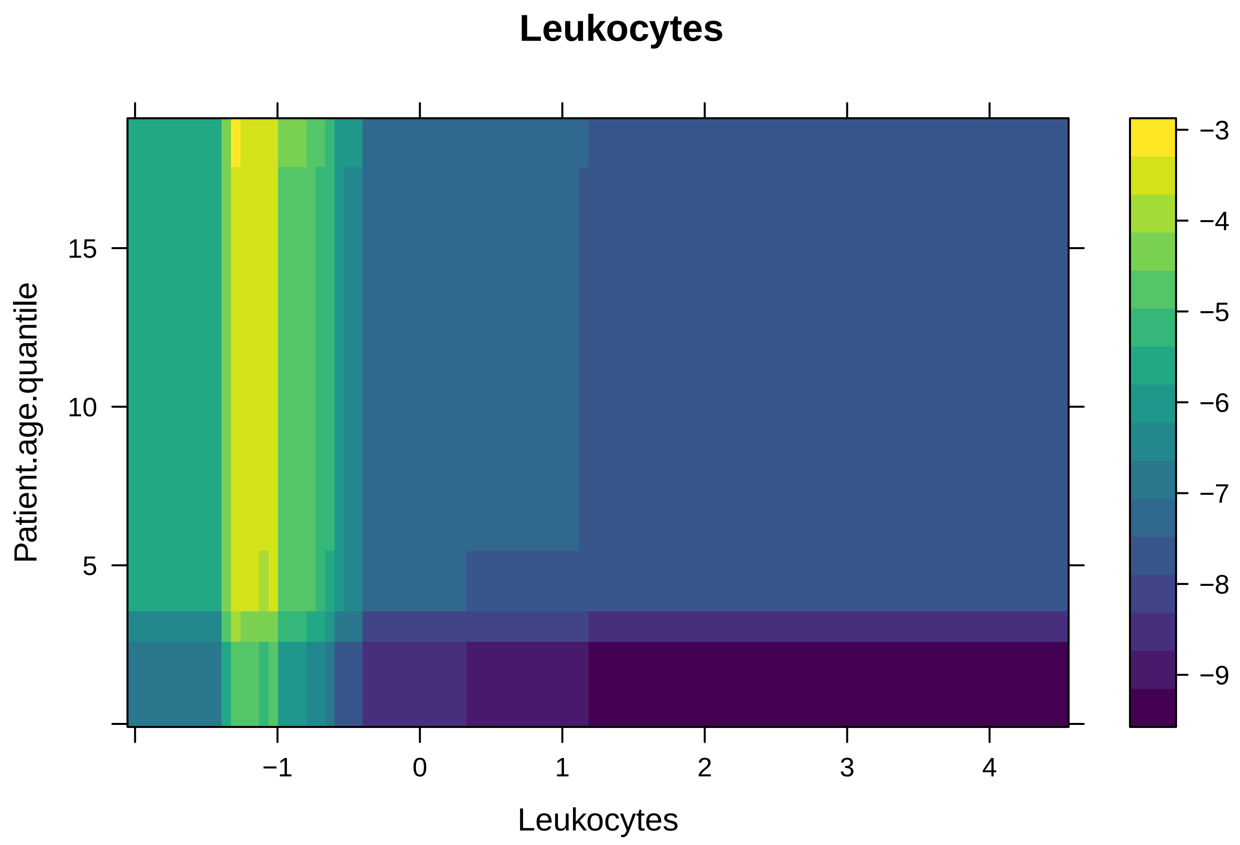
```
plot.gbm(gbm.model, i.var = c(as.character(model.summary$var[1]), 'Patient.age.quantile'),  main="Rhino
```



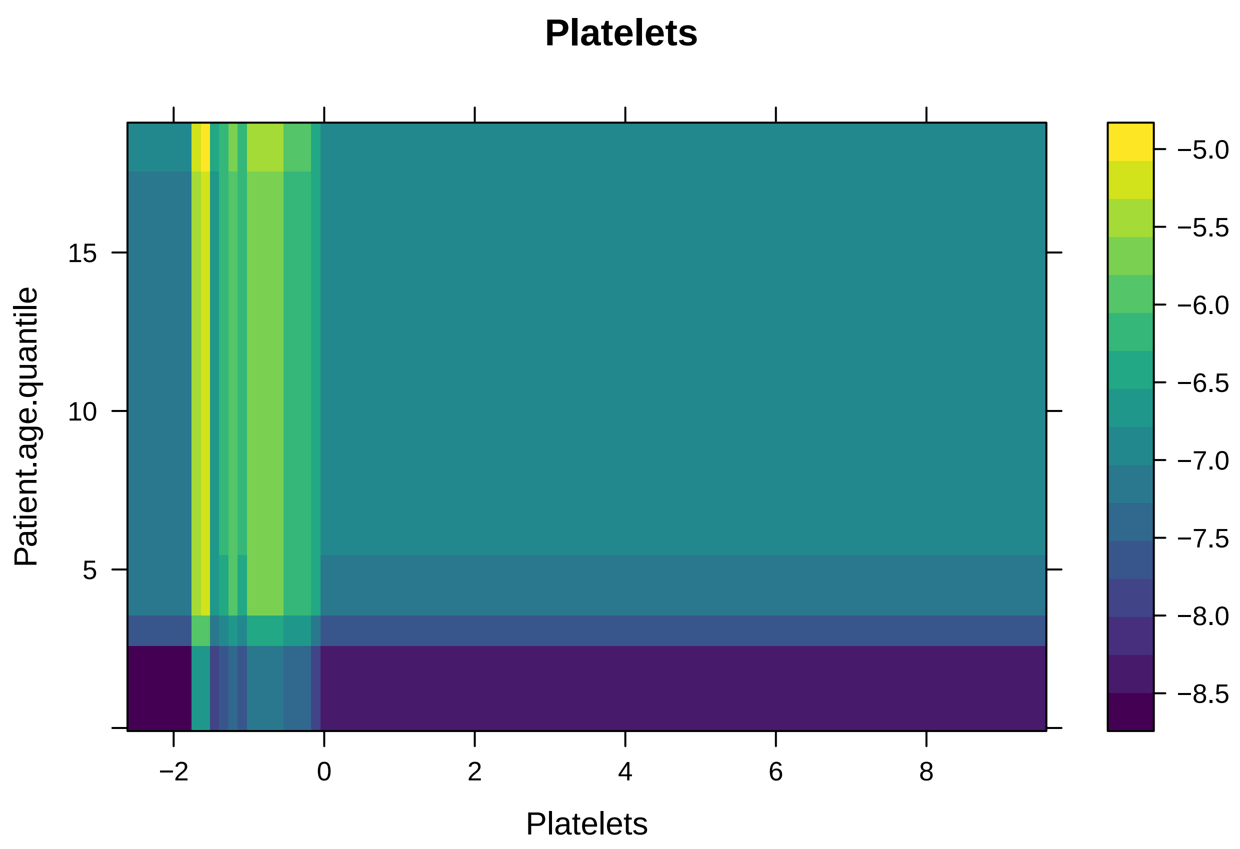
```
plot.gbm(gbm.model, i.var = c(as.character(model.summary$var[2]), 'Patient.age.quantile'),  main="Influ
```



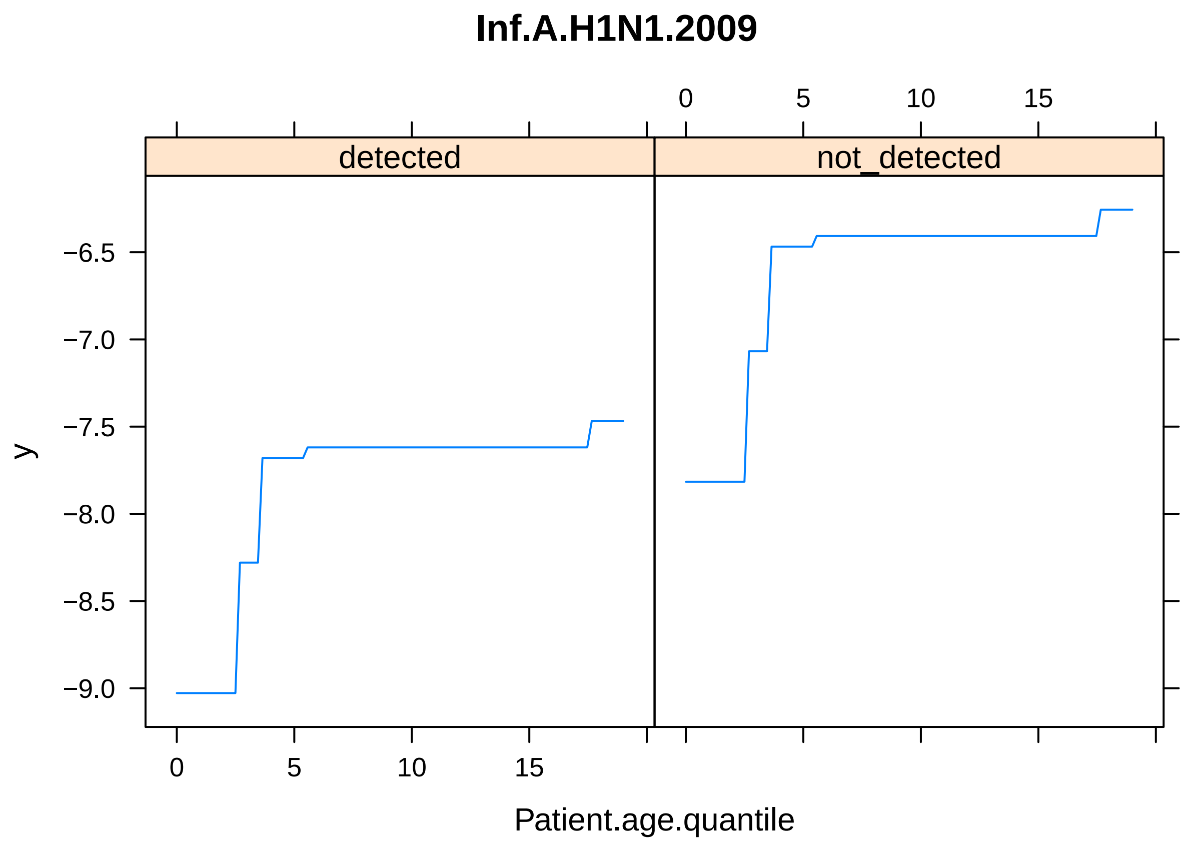
```
plot.gbm(gbm.model, i.var = c(as.character(model.summary$var[3]), 'Patient.age.quantile'),  main="Leuko
```



```
plot.gbm(gbm.model, i.var = c(as.character(model.summary$var[4]), 'Patient.age.quantile'),  main="Platel
```



```
plot.gbm(gbm.model, i.var = c(as.character(model.summary$var[5]), 'Patient.age.quantile'),  main="Inf.A
```



Prediction

when we apply the trained model to the test dataset, the model turn out to perform very well with an AUC of 94%. However, the determination of model's specificity and sensitivity relies on the definition of a likelihood threshold to determine patients that will be considered as likely positive COVID-19 cases among suspected cases.

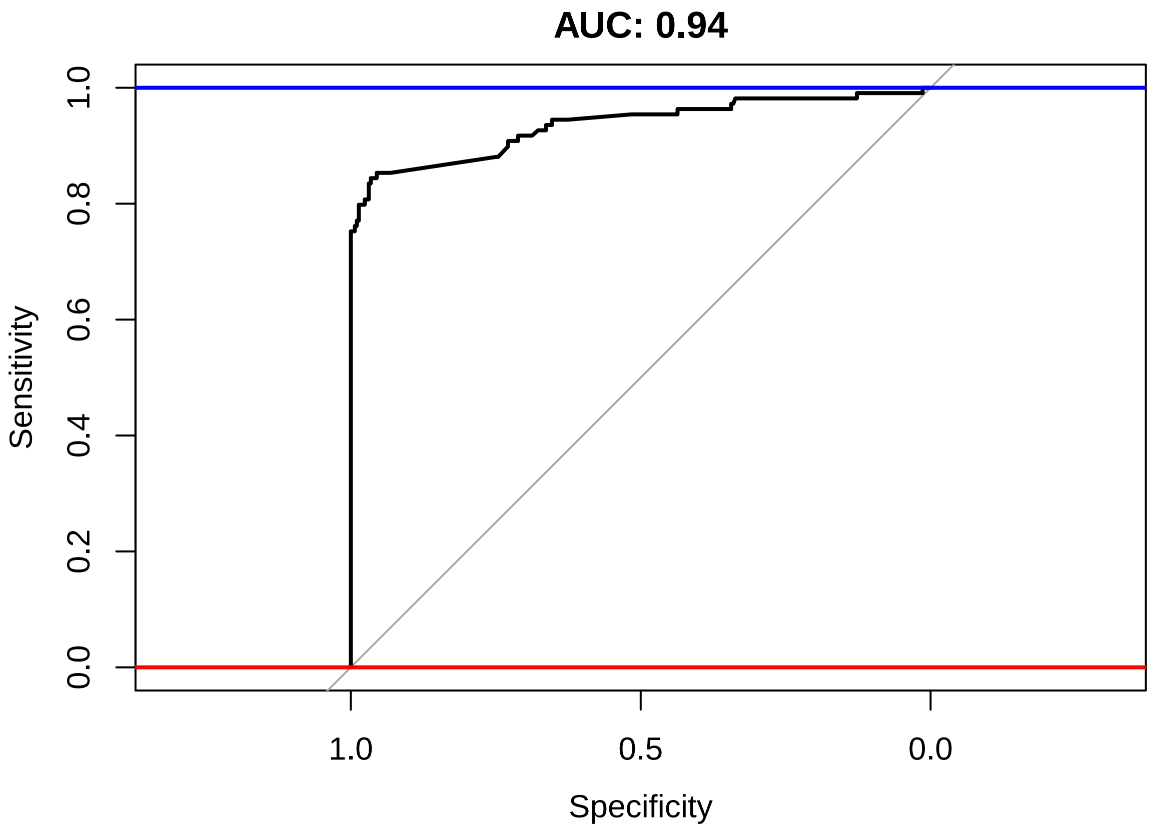
```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
test.current.prediction <-predict(gbm.model, newdata = test, n.trees = 500,
                                   type="response")

x.roc<-roc(response=test$SARS.Cov.2.exam.result, predictor=test.current.prediction)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
plot(x.roc, ylim=c(0,1),
     main=paste('AUC:',round(x.roc$auc[[1]],2)))
abline(h=1,col='blue',lwd=2)
abline(h=0,col='red',lwd=2)
```



A model with high sensitivity achieves good results in finding positive patients among those true positive patients. However, the number of patients predicted to be positive can be too high and impact the model's specificity.

Moreover, the hospital may not have enough resources to apply the necessary procedures for all patients assigned with a positive label if that number is too high. Hence, an ideal model is one that is well-balanced, i.e., one that has high sensitivity but it does not over-assign patients with positive labels.

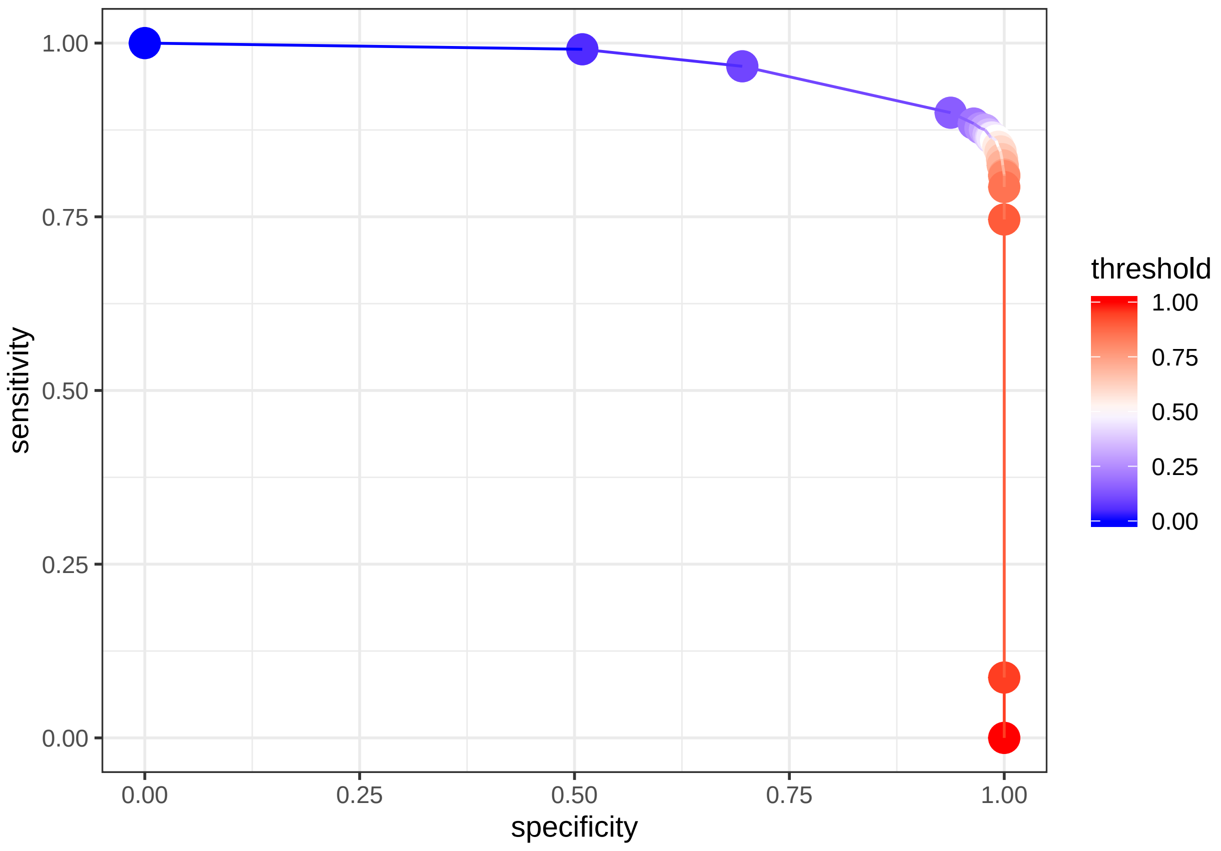
```r
train.current.prediction <-predict(gbm.model, newdata = train, n.trees = 500,
                                   type="response")
x.roc<-roc(response=train$SARS.Cov.2.exam.result, predictor=train.current.prediction)
```

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

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

```r
cc <- coords(x.roc, seq(from = 0, to = 1, by = 0.05), ret=c("sensitivity", "specificity", "threshold"),
library(ggplot2)
library(ggthemes)
mid<-median(cc$threshold)
```

```
ggplot(cc, aes(x=specificity, y=sensitivity,
               color=threshold,
               fill=threshold)) + geom_point(size = 5) + geom_line() +
  theme_bw() +
  scale_color_gradient2(midpoint=mid, low="blue", mid="white", high="red", space ="Lab" ) +
  scale_fill_gradient2(midpoint=mid, low="blue", mid="white", high="red", space ="Lab" )
```



Scenario 1: High availability of resources

In Scenario 1, we assume that the hospital has high availability of resources. In that way, the model can be relaxed and over-estimate the number of positive cases. Hence, our objective function is one that maximizes sensitivity.

We use the train data to select the threshold that maximizes model's sensitivity. We then apply this threshold in the predicted probabilities in the test set. The procedure returns a probability threshold of 5.8% and the model presents a high sensitivity value of 98%, as intended.

However, the high recall comes at the cost of specificity, which presents a low value of 21%. Moreover, about 79% of the patients from the test set were labeled as positive, hence the model has limited usage as a prioritization tool.

```
library(pROC)
train.current.prediction <-predict(gbm.model, newdata = train, n.trees = 500,
                            type="response")


best.th<-coords(roc=x.roc, x=1, input="sensitivity", transpose = FALSE)$threshold
print(paste0("Optimal threshold = ", best.th))
```

```
## [1] "Optimal threshold = 0.00587137563843436"
```

```
oos.current.prediction <-predict(gbm.model, newdata = test, n.trees = 500,
                                 type="response")

print(paste0("Pct patients predicted as infected = ", sum(oos.current.prediction > best.th) / length(oo
```

```
## [1] "Pct patients predicted as infected = 0.8375"
```

```
oos.x.roc<-roc(test$SARS.Cov.2.exam.result, predictor=oos.current.prediction)
```

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

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

```
BinModelPerformance(oos.current.prediction, best.th, test$SARS.Cov.2.exam.result)
```

```
##             Reference
## Prediction positive negative
##    positive      107      228
##    negative        2       63
```

```
##       ROC      Sens      Spec
## 0.9358586 0.9816514 0.2164948
```

Scenario 2: Limited resources

In Scenario 2, we assume an environment with limited resources and hence a reduction in model's sensitivity is acceptable if we can obtain a well-balanced model, overall. For that purpose, we choose as objective function one that maximizes the Youden J's statistic defined as $max(sensitivity + specificy)$.

After making a prediction on the test set, we will then choose a threshold from the train set that maximizes the Youden J's statistic to achieve a well-balanced model. We observe that the model under Scenario 2 now delivers a Sensitivity of 82% compared to 98% from Scenario 1. However, it returns a Specificity of 97% while maintaining a high AUC of 94% (as the choice of threshold does not influence the AUC), hence delivering a more well-balanced model as expected. Moreover, now the model only assigns 28% of the test set with positive labels, showing to be useful as a potential patient prioritization tool.

```
oos.current.prediction <-predict(gbm.model, newdata = test, n.trees = 500,
                                 type="response")
```

```r
#obtain optimum threshold
best.th<-coords(x.roc, "best", ret="threshold", transpose = FALSE,
                best.method="youden")$threshold
print(paste0("Optimal threshold = ", best.th))
```

```
## [1] "Optimal threshold = 0.30444438520008"
```

```r
print(paste0("Pct patients predicted as infected = ",
             sum(oos.current.prediction > best.th) / length(oos.current.prediction)))
```

```
## [1] "Pct patients predicted as infected = 0.2475"
```

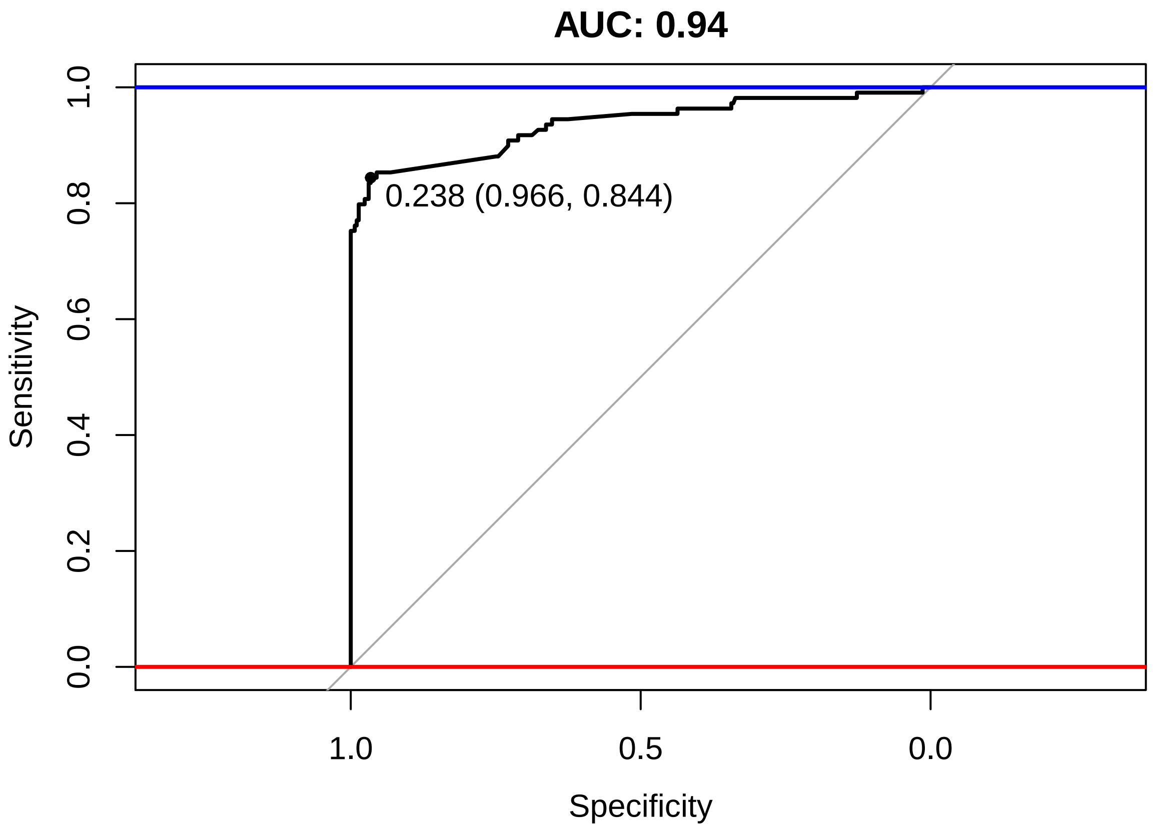
```r
BinModelPerformance(oos.current.prediction, best.th,  test$SARS.Cov.2.exam.result)
```

```
##            Reference
## Prediction positive negative
##   positive       90        9
##   negative       19      282

##       ROC      Sens      Spec
## 0.9358586 0.8256881 0.9690722
```

```r
oos.x.roc<-roc(test$SARS.Cov.2.exam.result, predictor=oos.current.prediction)
```

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

## Setting direction: controls < cases
```

```r
# OUT-OF-SAMPLE ROC
plot(oos.x.roc, ylim=c(0,1), print.thres="best", print.thres.best.method="youden",
     main=paste('AUC:',round(oos.x.roc$auc[[1]],2)))
abline(h=1,col='blue',lwd=2)
abline(h=0,col='red',lwd=2)
```



Conclusion

The model's output can be used as a tool for prioritization and to support further medical decision making processes. The model has high interpretability further showing that patients admitted with COVID-19 symptoms who tested negative for Rhinovirus Enterovirus, Influenza B and Inf.A.H1N1.2009 and presented low levels of Leukocytes and Platelets were more likely to test positive for SARS-CoV-2.