Explainable Artificial Intelligence (AI): Local Explanation with DALEX (SHAP-like explanations)

Lumumba Wandera Victor

2024-11-30

EXPLAINABLE AI FOR MALARIA MODELING: Local Explanation with DALEX

INSTRUCTOR: LUMUMBA WANDERA VICTOR

AMMnet ML GROUP

30TH NOVEMBER, 2024

Definition:

Explainable AI (XAI)refers to a set of tools and techniques that make the behavior of machine learning models more interpretable andunderstandable to humans.

Goals

The goal of XAI is to clarify how and why AI models make certain predictions ensuring that both experts and non-experts can trust and effectively use these insights in real-world applications.

In malaria modeling, for instance, XAI can explain why an AI model predicts high malaria severity in revealing which symptoms (like vomiting, jundice, or fever) are influencing that prediction.

By increasing model interpretability, XAI helps bridge the gap between complex AI technologies and actionable insights for policymakers, healthcare workers, and researchers, allowing them to use AI insights more confidently in planning and intervention.

TYpes

- SHapleY Additive Model exPlanations (SHAP)
- Local Interpretable Model-Agnostic Explanations (LIME)
- Local Explanation with DALEX (SHAP-like explanations)

These methods help identify which features (or variables) contribute most to a model's predictions.

Load necessary packages

```
#install.packages("caret")  # If not already installed
#install.packages("iml")
#install.packages("lime")
#install.packages('tictoc')
#install.packages("ggplot2")
#install.packages("dplyr")
```

Load LIBRARIES

```
library(caret)
library(iml)
library(lime)
library(tictoc)
library(ggplot2)
library(dplyr)
```

Load the Malaria dataset

Source: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7093799/

```
mdata = read.csv("Malaria-Data.csv", header = TRUE)
attach(mdata)
```

```
dim(mdata)
```

```
[1] 337  18
```

```
head(mdata)
```

```
  age sex fever cold rigor fatigue headace bitter_tongue vomitting diarrhea
1   3   1     1    1     0       1       1             1         0        1
2   3   0     1    1     1       1       1             1         0        1
3   3   0     1    1     1       1       1             0         0        1
4   4   1     1    1     0       1       0             0         0        0
5   4   0     1    1     1       0       1             0         0        0
6   4   1     0    0     0       1       1             0         0        1
  Convulsion Anemia jundice cocacola_urine hypoglycemia prostraction
1          1      0       1              1            1            0
2          0      0       0              1            1            0
3          1      0       0              1            1            0
4          0      0       1              0            1            0
5          1      0       1              1            1            0
6          0      1       0              0            0            0
  hyperpyrexia severe_maleria
1            0              0
2            0              0
3            0              1
4            1              0
5            0              0
6            0              1
```

View the Column Names in the Data set

```
names(mdata)
```

```
 [1] "age"            "sex"            "fever"          "cold"
 [5] "rigor"          "fatigue"        "headace"        "bitter_tongue"
 [9] "vomitting"      "diarrhea"       "Convulsion"     "Anemia"
[13] "jundice"        "cocacola_urine" "hypoglycemia"   "prostraction"
[17] "hyperpyrexia"   "severe_maleria"
```

View the Structure of the Dataset

```
str(mdata)
```

```
'data.frame':   337 obs. of  18 variables:
 $ age           : int  3 3 3 4 4 4 4 5 5 8 ...
 $ sex           : int  1 0 0 1 0 1 0 1 0 0 ...
 $ fever         : int  1 1 1 1 1 0 1 1 1 1 ...
 $ cold          : int  1 1 1 1 1 0 1 0 0 1 ...
 $ rigor         : int  0 1 1 0 1 0 1 1 1 1 ...
 $ fatigue       : int  1 1 1 1 0 1 1 1 1 0 ...
 $ headace       : int  1 1 1 0 1 1 0 0 1 1 ...
 $ bitter_tongue : int  1 1 0 0 0 0 0 0 1 1 ...
 $ vomitting     : int  0 0 0 0 0 0 0 1 0 0 ...
 $ diarrhea      : int  1 1 1 0 0 1 0 1 0 1 ...
 $ Convulsion    : int  1 0 1 0 1 0 0 0 1 0 ...
 $ Anemia        : int  0 0 0 0 0 1 0 1 0 0 ...
 $ jundice       : int  1 0 0 1 1 0 0 1 0 1 ...
 $ cocacola_urine: int  1 1 1 0 1 0 0 0 0 0 ...
 $ hypoglycemia  : int  1 1 1 1 1 0 0 0 1 1 ...
 $ prostraction  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ hyperpyrexia  : int  0 0 0 1 0 0 0 0 0 0 ...
 $ severe_maleria: int  0 0 1 0 0 1 0 0 0 0 ...
```

Descriptive Statistics

```
summary(mdata) ###Descriptive Statistics
```

```
      age             sex              fever             cold
 Min.   : 3.00   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:19.00   1st Qu.:0.0000   1st Qu.:1.0000   1st Qu.:0.0000
 Median :29.00   Median :1.0000   Median :1.0000   Median :1.0000
 Mean   :30.35   Mean   :0.5341   Mean   :0.7507   Mean   :0.5668
 3rd Qu.:38.00   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
 Max.   :77.00   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
     rigor            fatigue           headace       bitter_tongue
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :0.0000   Median :0.0000   Median :1.0000   Median :0.0000  
 Mean   :0.3412   Mean   :0.4837   Mean   :0.7003   Mean   :0.4036  
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
   vomitting          diarrhea        Convulsion         Anemia      
 Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000  
 Median :0.00000   Median :0.0000   Median :0.0000   Median :0.0000  
 Mean   :0.07418   Mean   :0.3383   Mean   :0.3442   Mean   :0.3501  
 3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000  
 Max.   :1.00000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
    jundice       cocacola_urine    hypoglycemia     prostraction   
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:1.0000   1st Qu.:0.0000  
 Median :1.0000   Median :1.0000   Median :1.0000   Median :0.0000  
 Mean   :0.6588   Mean   :0.5401   Mean   :0.8576   Mean   :0.2196  
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.0000  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
  hyperpyrexia    severe_maleria  
 Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.0000   1st Qu.:0.0000  
 Median :0.0000   Median :0.0000  
 Mean   :0.1395   Mean   :0.3442  
 3rd Qu.:0.0000   3rd Qu.:1.0000  
 Max.   :1.0000   Max.   :1.0000  
```

```r
library(psych)
describe(mdata)###Descriptive Statistics
```

```
               vars   n  mean    sd median trimmed   mad min max range  skew
age               1 337 30.35 14.72     29   29.22 14.83   3  77    74  0.75
sex               2 337  0.53  0.50      1    0.54  0.00   0   1     1 -0.14
fever             3 337  0.75  0.43      1    0.81  0.00   0   1     1 -1.15
cold              4 337  0.57  0.50      1    0.58  0.00   0   1     1 -0.27
rigor             5 337  0.34  0.47      0    0.30  0.00   0   1     1  0.67
fatigue           6 337  0.48  0.50      0    0.48  0.00   0   1     1  0.07
headace           7 337  0.70  0.46      1    0.75  0.00   0   1     1 -0.87
bitter_tongue     8 337  0.40  0.49      0    0.38  0.00   0   1     1  0.39
vomitting         9 337  0.07  0.26      0    0.00  0.00   0   1     1  3.24
diarrhea         10 337  0.34  0.47      0    0.30  0.00   0   1     1  0.68
Convulsion       11 337  0.34  0.48      0    0.31  0.00   0   1     1  0.65
Anemia           12 337  0.35  0.48      0    0.31  0.00   0   1     1  0.63
jundice          13 337  0.66  0.47      1    0.70  0.00   0   1     1 -0.67
cocacola_urine   14 337  0.54  0.50      1    0.55  0.00   0   1     1 -0.16
hypoglycemia     15 337  0.86  0.35      1    0.94  0.00   0   1     1 -2.04
prostraction     16 337  0.22  0.41      0    0.15  0.00   0   1     1  1.35
hyperpyrexia     17 337  0.14  0.35      0    0.05  0.00   0   1     1  2.07
severe_maleria   18 337  0.34  0.48      0    0.31  0.00   0   1     1  0.65
               kurtosis   se
age                0.49 0.80
sex               -1.99 0.03
fever             -0.67 0.02
cold              -1.93 0.03
rigor             -1.56 0.03
```

```
fatigue           -2.00 0.03
headace           -1.25 0.02
bitter_tongue     -1.85 0.03
vomitting          8.49 0.01
diarrhea          -1.54 0.03
Convulsion        -1.58 0.03
Anemia            -1.61 0.03
jundice           -1.56 0.03
cocacola_urine    -1.98 0.03
hypoglycemia       2.16 0.02
prostraction      -0.18 0.02
hyperpyrexia       2.30 0.02
severe_maleria    -1.58 0.03
```

Check the Number of Missing Values

```r
sum(is.na(mdata))###Check for missing data
```

```
[1] 0
```

Exclude Age from the dataset

```r
mdata=mdata[,-1] ##Exclude Age
names(mdata)
```

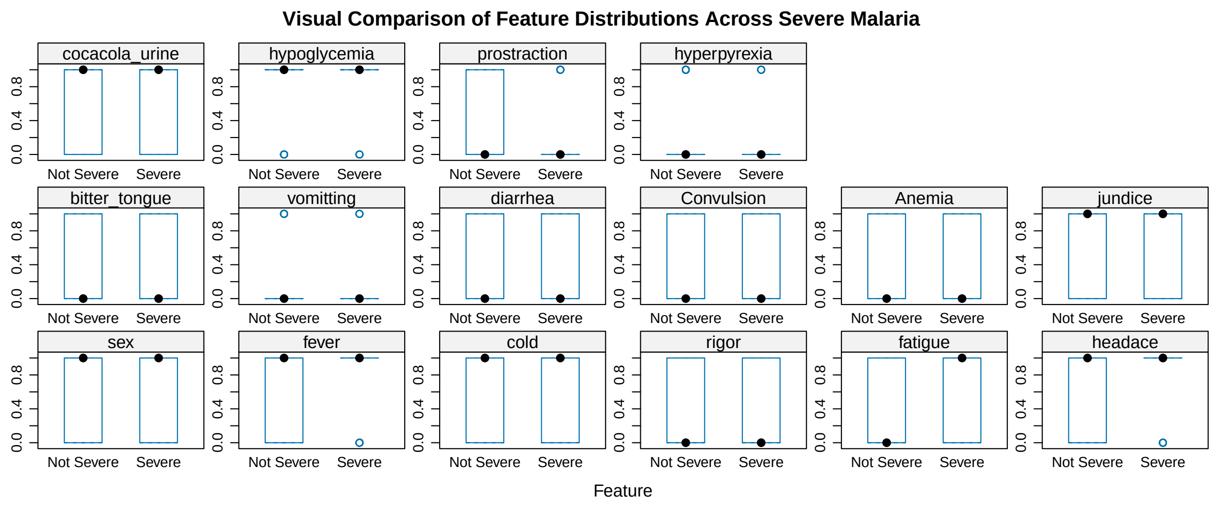
```
 [1] "sex"            "fever"          "cold"           "rigor"
 [5] "fatigue"        "headace"        "bitter_tongue"  "vomitting"
 [9] "diarrhea"       "Convulsion"     "Anemia"         "jundice"
[13] "cocacola_urine" "hypoglycemia"   "prostraction"   "hyperpyrexia"
[17] "severe_maleria"
```

Rename the classes of the Target variable and plot it to determine imbalance

```r
mdata$severe_maleria <- factor(mdata$severe_maleria,
                               levels = c(0,1),
                               labels = c('Not Severe', 'Severe'))
```

Perform Featureplot to see the data distribution at a glance

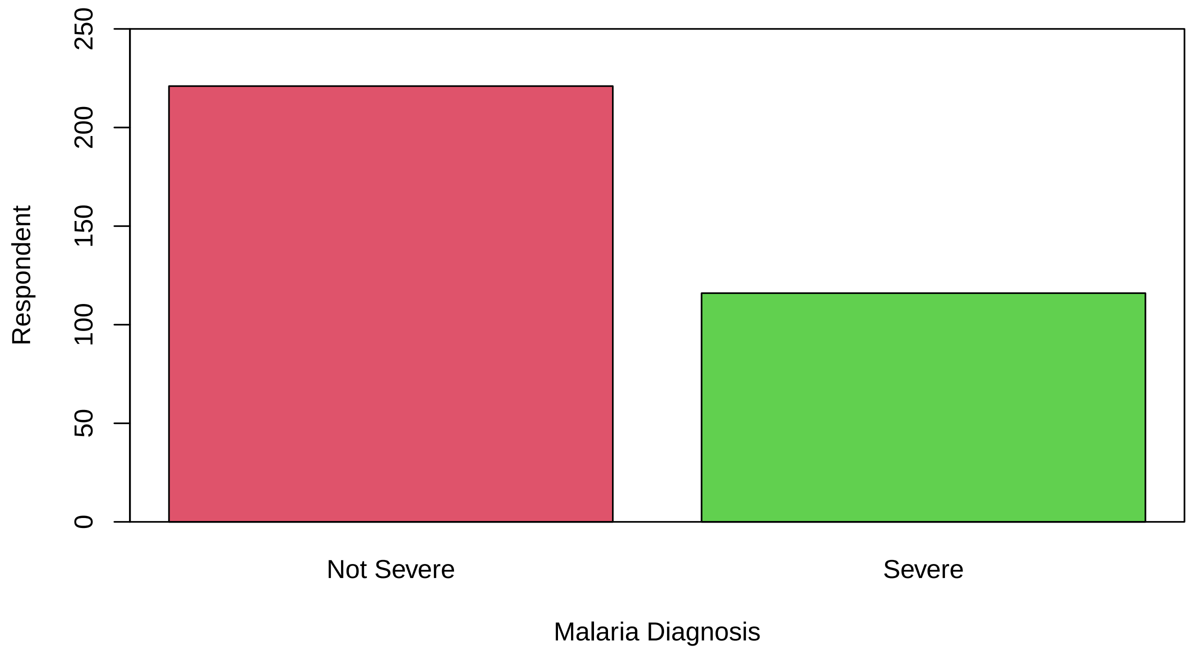
```r
#png("Features Plot.png", height = 800, width = 1000)
featurePlot(x = mdata[, -which(names(mdata) == "severe_maleria")],   # Predictors
            y = mdata$severe_maleria,                                # Target variable
            plot = "box",                                        # Type of plot (e.g., "box", "density",
            #strip = strip.custom(strip.names = TRUE),             # Add strip labels
            scales = list(x = list(relation = "free"),            # Scales for x-axis
                          y = list(relation = "free")), main = "Visual Comparison of Feature Distributi
```



```r
#dev.off()
```

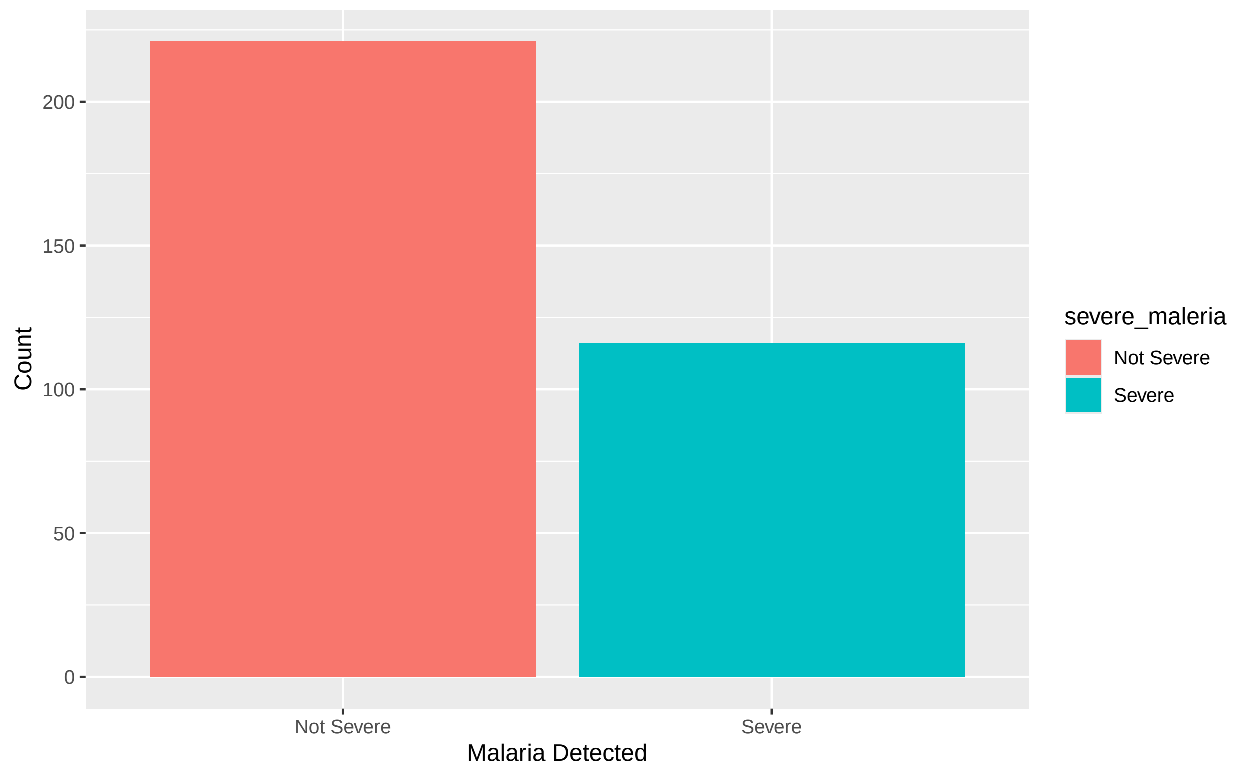
Plot Target Variable

```r
plot(factor(severe_maleria), names= c('Not Severe', 'Severe'),
     col=c(2,3),
     ylim=c(0, 250),
     ylab='Respondent',
     xlab='Malaria Diagnosis')
box()
```



#Or use ggplot

```
ggplot(mdata, aes(x = factor(severe_maleria), fill = severe_maleria)) +
  geom_bar() +
  labs(x = "Malaria Detected",
       y = "Count")
```



VIEW THE AVAILABLE MODELS IN CARET

```
models = getModelInfo()
names(models)
```

```
  [1] "ada"                  "AdaBag"               "AdaBoost.M1"
  [4] "adaboost"             "amdai"                "ANFIS"
  [7] "avNNet"               "awnb"                 "awtan"
 [10] "bag"                  "bagEarth"             "bagEarthGCV"
 [13] "bagFDA"               "bagFDAGCV"            "bam"
 [16] "bartMachine"          "bayesglm"             "binda"
 [19] "blackboost"           "blasso"               "blassoAveraged"
 [22] "bridge"               "brnn"                 "BstLm"
 [25] "bstSm"                "bstTree"              "C5.0"
 [28] "C5.0Cost"             "C5.0Rules"            "C5.0Tree"
 [31] "cforest"              "chaid"                "CSimca"
 [34] "ctree"                "ctree2"               "cubist"
 [37] "dda"                  "deepboost"            "DENFIS"
 [40] "dnn"                  "dwdLinear"            "dwdPoly"
 [43] "dwdRadial"            "earth"                "elm"
 [46] "enet"                 "evtree"               "extraTrees"
 [49] "fda"                  "FH.GBML"              "FIR.DM"
 [52] "foba"                 "FRBCS.CHI"            "FRBCS.W"
 [55] "FS.HGD"               "gam"                  "gamboost"
 [58] "gamLoess"             "gamSpline"            "gaussprLinear"
 [61] "gaussprPoly"          "gaussprRadial"        "gbm_h2o"
 [64] "gbm"                 "gcvEarth"            "GFS.FR.MOGUL"
 [67] "GFS.LT.RS"           "GFS.THRIFT"          "glm.nb"
 [70] "glm"                 "glmboost"            "glmnet_h2o"
 [73] "glmnet"              "glmStepAIC"          "gpls"
 [76] "hda"                 "hdda"                "hdrda"
 [79] "HYFIS"               "icr"                 "J48"
 [82] "JRip"                "kernelpls"           "kknn"
 [85] "knn"                 "krlsPoly"            "krlsRadial"
 [88] "lars"                "lars2"               "lasso"
 [91] "lda"                 "lda2"                "leapBackward"
 [94] "leapForward"         "leapSeq"             "Linda"
 [97] "lm"                  "lmStepAIC"           "LMT"
[100] "loclda"              "logicBag"            "LogitBoost"
[103] "logreg"              "lssvmLinear"         "lssvmPoly"
[106] "lssvmRadial"         "lvq"                 "M5"
[109] "M5Rules"             "manb"                "mda"
[112] "Mlda"                "mlp"                 "mlpKerasDecay"
[115] "mlpKerasDecayCost"   "mlpKerasDropout"     "mlpKerasDropoutCost"
[118] "mlpML"               "mlpSGD"              "mlpWeightDecay"
[121] "mlpWeightDecayML"    "monmlp"              "msaenet"
[124] "multinom"            "mxnet"               "mxnetAdam"
[127] "naive_bayes"         "nb"                  "nbDiscrete"
[130] "nbSearch"            "neuralnet"           "nnet"
[133] "nnls"                "nodeHarvest"         "null"
[136] "OneR"                "ordinalNet"          "ordinalRF"
[139] "ORFlog"              "ORFpls"              "ORFridge"
[142] "ORFsvm"              "ownn"                "pam"
[145] "parRF"               "PART"                "partDSA"
[148] "pcaNNet"             "pcr"                 "pda"
[151] "pda2"                "penalized"           "PenalizedLDA"
[154] "plr"                 "pls"                 "plsRglm"
[157] "polr"                "ppr"                 "pre"
[160] "PRIM"                "protoclass"          "qda"
[163] "QdaCov"              "qrf"                 "qrnn"
[166] "randomGLM"           "ranger"              "rbf"
[169] "rbfDDA"              "Rborist"             "rda"
[172] "regLogistic"         "relaxo"              "rf"
[175] "rFerns"              "RFlda"               "rfRules"
[178] "ridge"               "rlda"                "rlm"
[181] "rmda"                "rocc"                "rotationForest"
[184] "rotationForestCp"    "rpart"               "rpart1SE"
[187] "rpart2"              "rpartCost"           "rpartScore"
[190] "rqlasso"             "rqnc"                "RRF"
[193] "RRFglobal"           "rrlda"               "RSimca"
[196] "rvmLinear"           "rvmPoly"             "rvmRadial"
[199] "SBC"                 "sda"                 "sdwd"
[202] "simpls"              "SLAVE"               "slda"
[205] "smda"                "snn"                 "sparseLDA"
[208] "spikeslab"           "spls"                "stepLDA"
[211] "stepQDA"             "superpc"             "svmBoundrangeString"
[214] "svmExpoString"       "svmLinear"           "svmLinear2"
[217] "svmLinear3"          "svmLinearWeights"    "svmLinearWeights2"
[220] "svmPoly"             "svmRadial"           "svmRadialCost"
[223] "svmRadialSigma"      "svmRadialWeights"    "svmSpectrumString"
```

```
[226] "tan"                  "tanSearch"            "treebag"
[229] "vbmpRadial"           "vglmAdjCat"           "vglmContRatio"
[232] "vglmCumulative"       "widekernelpls"        "WM"
[235] "wsrf"                 "xgbDART"              "xgbLinear"
[238] "xgbTree"              "xyf"
```

TODAY we are going to train the following machine learning models:

- LR
- SVM
- RANDOM FOREST
- NAIVE BAYES
- KNN
- LDA
- NNET/mlp
- LVQ
- Bagging
- Boosting
- DT

STEPS

1. Data Preparation and Preprocessing, Cleaning, Feature Engineering,Visualization, Data Splitting, etc
2. Define the Training Control- Set up cross validation
3. Train the Models- Select the ML models you want to train
4. Evaluate your model using test data
5. Tune the hyperparameters and Resample the data (optional)
6. Implement XAI

DATA PARTITION FOR MACHINE LEARNING

caret can also be used for data partition

```r
set.seed(123)
trainIndex <- createDataPartition(mdata$severe_maleria, p = 0.7, list = FALSE)
train <- mdata[trainIndex, ]
test <- mdata[-trainIndex, ]
dim(train)
```

```
[1] 237  17
```

```r
dim(test)
```

```
[1] 100  17
```

Set seed for reproducibility

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

Define control for training

```r
#control1 <- trainControl(method = "cv", number = 10)
control1 <- trainControl(method ="repeatedcv",
                         number=10, repeats=5,
                         sampling='smote',
                         search='random')## For tuning
```

The R code above defines a trainControl object named control1, used to configure the training process for machine learning models within the caret package. The method = "repeatedcv" specifies that repeated k-fold cross-validation will be applied, ensuring robust model performance evaluation. Specifically, number = 10 sets the number of folds to 10, and repeats = 5 means this cross-validation will be repeated five times, reducing the variability in performance metrics. The sampling = 'smote' parameter implements Synthetic Minority Over-sampling Technique (SMOTE) during training, a technique that addresses class imbalance by generating synthetic examples in the minority class. Lastly, search = 'random' indicates that hyperparameter tuning will be performed using random search rather than a grid search, making it more efficient by sampling a random subset of the hyperparameter space. This setup improves the model's accuracy and generalizability, particularly for imbalanced datasets, by preventing overfitting and ensuring reliable hyperparameter selection.

1. Train the Logistic Regression Model

```r
tic()
lrModel <- train(severe_maleria ~ .,
                 data = train,
                 method = "glm",
                 trControl = control1)
toc()
```

```
1.61 sec elapsed
```

Predict on the test set

```r
lrpred <- predict(lrModel, newdata = test)
```

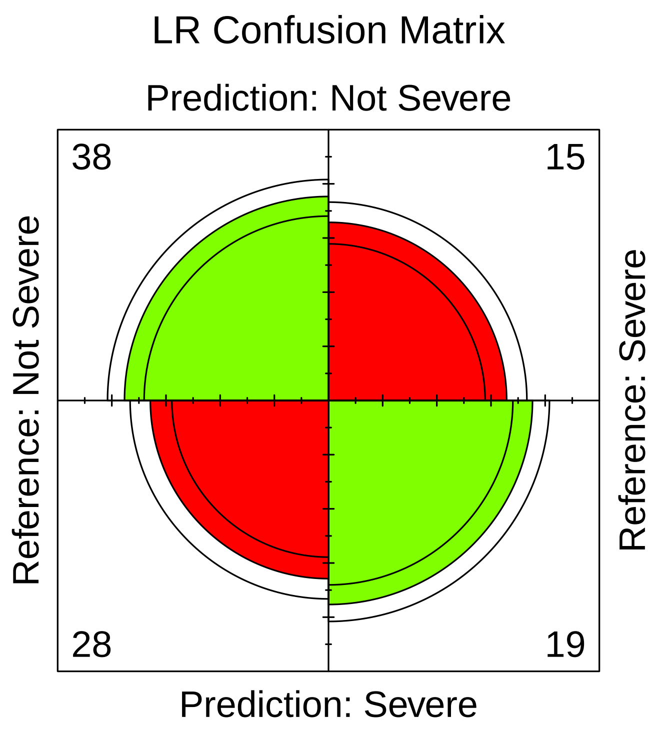
Evaluate with Confusion Matrix

```r
lr.cM <- confusionMatrix(lrpred, test$severe_maleria,
                         positive = "Severe",
                         mode = "everything")
print(lr.cM)
```

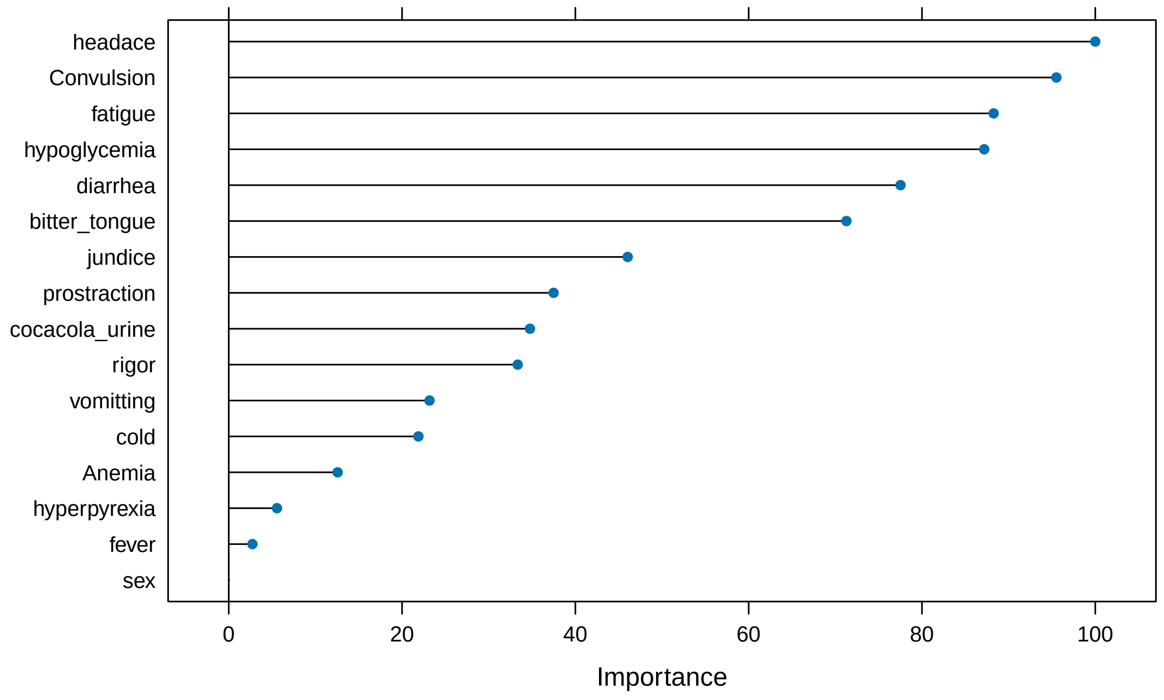
```
Confusion Matrix and Statistics

            Reference
Prediction   Not Severe Severe
  Not Severe         38     15
  Severe             28     19

               Accuracy : 0.57
                 95% CI : (0.4671, 0.6686)
    No Information Rate : 0.66
    P-Value [Acc > NIR] : 0.97603

                  Kappa : 0.1232

 Mcnemar's Test P-Value : 0.06725

            Sensitivity : 0.5588
            Specificity : 0.5758
         Pos Pred Value : 0.4043
         Neg Pred Value : 0.7170
              Precision : 0.4043
                 Recall : 0.5588
                     F1 : 0.4691
             Prevalence : 0.3400
         Detection Rate : 0.1900
   Detection Prevalence : 0.4700
      Balanced Accuracy : 0.5673

       'Positive' Class : Severe
```

Plotting confusion matrix

```
fourfoldplot(lr.cM$table,
             col = rainbow(4),
             main = "LR Confusion Matrix")
```



```r
plot(varImp(lrModel, scale = TRUE))
```



SHAP (SHapley Additive exPlanations)

The Shapley value helps explain how much each feature contributes to the prediction made by a machine learning model. It provides a way to fairly distribute the "credit" for the model's output across all input features. By visualizing the SHAP plot, you can understand not only which features are important,but also how specific feature values that are driving predictions for individual cases.

Set seed for reproducibility

```r
set.seed(456)
```

Assuming lrModel is already trained : Convert the caret model to a Predictor object, separating the target variable

```r
predictorlr <- Predictor$new(lrModel,
                             data = train[, -which(names(train) == "severe_maleria")],
                             y = train$severe_maleria)
```

Select a single instance from the test set to explain Replace '1' with the index of any other instance if desired

```r
x_interest <- test[1, -which(names(test) == "severe_maleria")]
```

Compute SHAP values for the specific instance

```r
shapleylr <- Shapley$new(predictorlr, x.interest = x_interest)
```

View the SHAP Values

```r
shapleylr$results
```

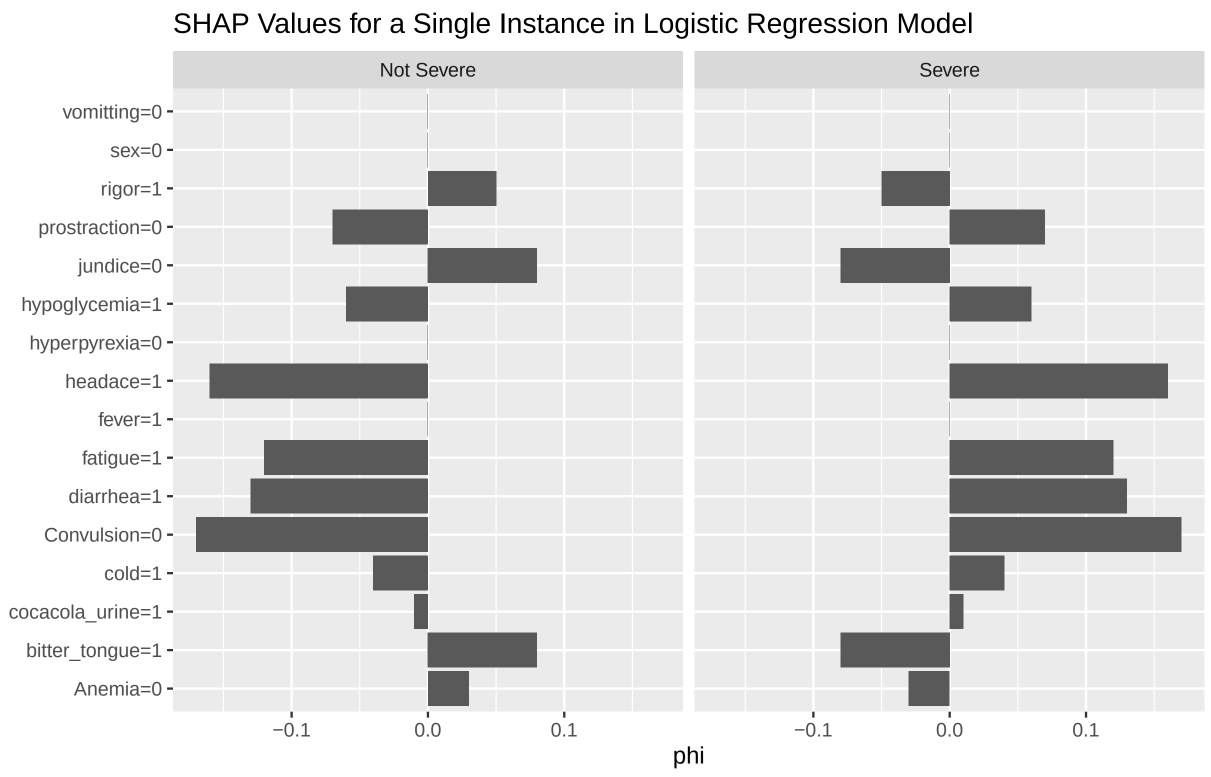
```
         feature      class   phi    phi.var   feature.value
1            sex Not Severe  0.00 0.00000000           sex=0
2          fever Not Severe  0.00 0.00000000         fever=1
3           cold Not Severe -0.04 0.03878788          cold=1
4          rigor Not Severe  0.05 0.04797980         rigor=1
5        fatigue Not Severe -0.12 0.10666667       fatigue=1
6        headace Not Severe -0.16 0.13575758       headace=1
7   bitter_tongue Not Severe  0.08 0.07434343 bitter_tongue=1
8       vomitting Not Severe  0.00 0.00000000     vomitting=0
9       diarrhea Not Severe -0.13 0.11424242      diarrhea=1
10    Convulsion Not Severe -0.17 0.14252525    Convulsion=0
11        Anemia Not Severe  0.03 0.02939394        Anemia=0
12       jundice Not Severe  0.08 0.07434343       jundice=0
13 cocacola_urine Not Severe -0.01 0.01000000 cocacola_urine=1
14   hypoglycemia Not Severe -0.06 0.05696970  hypoglycemia=1
15   prostraction Not Severe -0.07 0.06575758  prostraction=0
16   hyperpyrexia Not Severe  0.00 0.00000000  hyperpyrexia=0
```

```
17            sex     Severe  0.00 0.00000000            sex=0
18          fever     Severe  0.00 0.00000000          fever=1
19           cold     Severe  0.04 0.03878788           cold=1
20          rigor     Severe -0.05 0.04797980          rigor=1
21        fatigue     Severe  0.12 0.10666667        fatigue=1
22        headace     Severe  0.16 0.13575758        headace=1
23  bitter_tongue     Severe -0.08 0.07434343  bitter_tongue=1
24      vomitting     Severe  0.00 0.00000000      vomitting=0
25       diarrhea     Severe  0.13 0.11424242       diarrhea=1
26     Convulsion     Severe  0.17 0.14252525     Convulsion=0
27         Anemia     Severe -0.03 0.02939394         Anemia=0
28        jundice     Severe -0.08 0.07434343        jundice=0
29 cocacola_urine     Severe  0.01 0.01000000 cocacola_urine=1
30   hypoglycemia     Severe  0.06 0.05696970   hypoglycemia=1
31   prostraction     Severe  0.07 0.06575758   prostraction=0
32   hyperpyrexia     Severe  0.00 0.00000000   hyperpyrexia=0
```

```
#View(shapleylr$results)
```

Plot the SHAP values for this instance

```
shapleylr$plot() +
  ggtitle("SHAP Values for a Single Instance in Logistic Regression Model")
```



INTERPRETATION

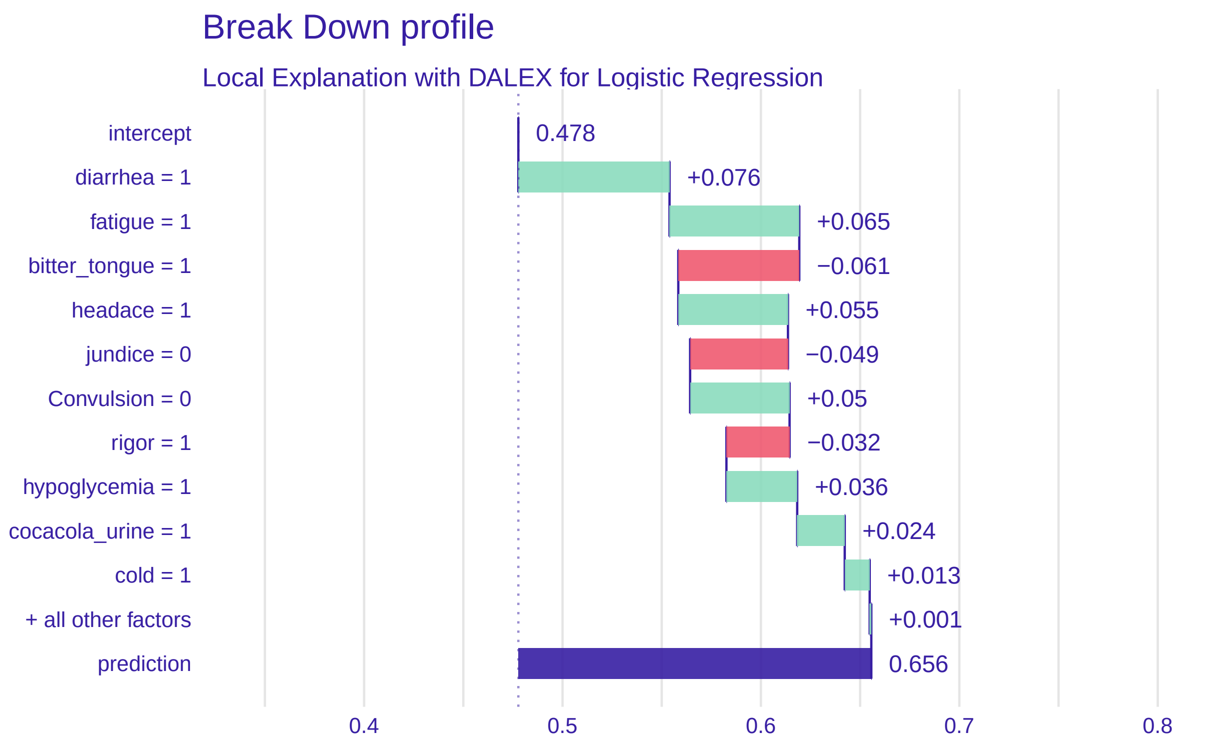
Each feature has its own SHAP value, calculated in the context of all other features. The direction and length of the bar indicate the magnitude and impact on the prediction. Rightward (positive): Indicates the feature is pushing the model prediction towards a positive class (e.g., "Severe" if that is the positive label).

Local Explanation with DALEX (SHAP-like explanations)

```r
#install.packages("DALEX")
library(DALEX)

# Create the explainer object
explainer <- explain(lrModel,
                     data = test[, -ncol(test)],  # Exclude the outcome column
                     y = as.numeric(as.character(test$severe_maleria)),
                     label = "Local Explanation with DALEX for Logistic Regression")
```

```
Preparation of a new explainer is initiated
  -> model label       :  Local Explanation with DALEX for Logistic Regression
  -> data              :  100  rows  16  cols
  -> target variable   :  100  values
  -> predict function  :  yhat.train  will be used (  default  )
  -> predicted values  :  No value for predict function target column. (  default  )
  -> model_info        :  package caret , ver. 6.0.94 , task classification (  default  )
  -> predicted values  :  numerical, min =  0.1097885 , mean =  0.4777402 , max =  0.7690111
  -> residual function :  difference between y and yhat (  default  )
  -> residuals         :  numerical, min =  NA , mean =  NA , max =  NA
  A new explainer has been created!
```

```r
# Select an instance to explain (e.g., first row in test data set)
instance <- test[1, -ncol(test)]  # Exclude the outcome column for prediction

# Generate explanations for the instance
local_explanation <- predict_parts(explainer, new_observation = instance)

# Plot local explanation
plot(local_explanation)
```



Overview

The graph presents a local explanation of a logistic regression model using the DALEX package. It visualizes how different predictor variables contribute to the prediction for a specific instance. The prediction is represented by the bar on the right, and the contributions of each variable are shown as horizontal bars.

Breakdown of Contributions

- Intercept: This baseline value represents the model's prediction when all predictor variables are zero or absent. In this case, the intercept is 0.478.
- Predictor Variables: Each predictor variable's contribution is shown as a bar. The color indicates the direction of the contribution:
- Green: Positive contribution, meaning the variable increases the prediction.
- Red: Negative contribution, meaning the variable decreases the prediction.

The length of the bar represents the magnitude of the contribution.

Interpretation of the Specific Variables

Diarrhea = 1: Having diarrhea positively contributes to the prediction, with a value of +0.076. Fatigue = 1: Fatigue also has a positive impact, contributing +0.065. Bitter_tongue = 1: A bitter tongue negatively contributes to the prediction, with a value of -0.061. Headache = 1: Having a headache positively contributes +0.055. Jundice = 0: Not having jaundice negatively contributes -0.049. Convulsion = 0: Not having convulsions positively contributes +0.05. Rigor = 1: Rigor negatively contributes -0.032. Hypoglycemia

= 1: Hypoglycemia positively contributes +0.036. Cocacola_urine = 1: Having coca-cola colored urine positively contributes +0.024. Cold = 1: Having a cold positively contributes +0.013. All other factors: The remaining factors not explicitly shown contribute a small positive value of +0.001.

Overall Prediction

Summing up all the contributions (intercept + predictor variables), we arrive at the final prediction of 0.656. This value represents the probability of a certain outcome, as logistic regression models typically output probabilities.

This graph provides a valuable tool for understanding how a logistic regression model arrives at a specific prediction. It highlights the relative importance of different predictor variables and their impact on the final outcome. However, it is essential to consider the limitations and interpret the results in conjunction with other model evaluation metrics.

Additional Explanation (More simpler explanation)

The plot explains how different health symptoms contribute to a prediction made by a logistic regression model. Let's break it down step by step in simple terms. The model predicts severe malaria occurrence. The prediction value shown at the bottom is 0.656 (or 65.6%), meaning the model is moderately confident about this outcome. The model starts with a base value, the "intercept," which is 0.478. This represents the starting point for the prediction, assuming no additional information about symptoms.

Adding symptoms:

The model adjusts the base value based on the presence or absence of various symptoms.

For example, diarrhea = 1 (having diarrhea) adds 0.076 to the base, increasing the prediction. Similarly, fatigue = 1 adds 0.065, and rigor = 1 (shivering) adds 0.036.

Subtracting symptoms:

Some symptoms decrease the prediction value.

For instance, bitter_tongue = 1 (a bitter taste in the mouth) subtracts 0.061, and jundice = 0 (absence of jaundice) subtracts 0.049.

The final prediction:

By adding and subtracting these values step by step, the model arrives at the final prediction of 0.656.

What does this mean

The plot shows how much each symptom influenced the prediction. Positive bars (green) pushed the prediction higher, while negative bars (red) pulled it down. This can help doctors or experts understand what factors were most important in the decision. For example, diarrhea and fatigue played a significant role in increasing the prediction, while a bitter tongue reduced it.

RANDOM FOREST

2. Train the Random Forest Classifier

```
tic()
rfModel <- train(factor(severe_maleria) ~ .,
                 data = train,
                 method = "rf", trControl = control1)
toc()
```

```
27.51 sec elapsed
```

Predict on the test set

```
rfpred <- predict(rfModel, newdata = test)
```

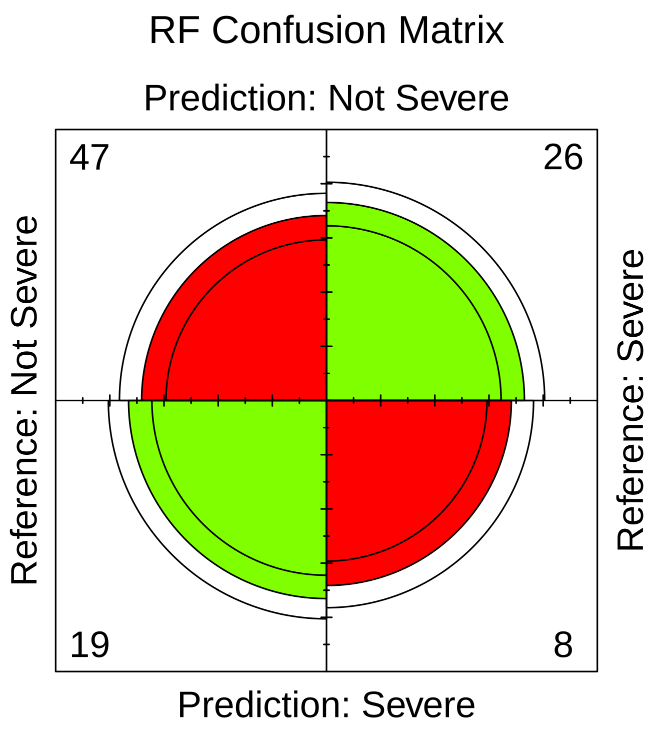
Evaluate with Confusion Matrix

```
rf.cM <- confusionMatrix(rfpred,
                         as.factor(test$severe_maleria),
                         positive = "Severe",
                         mode = "everything")
print(rf.cM)
```

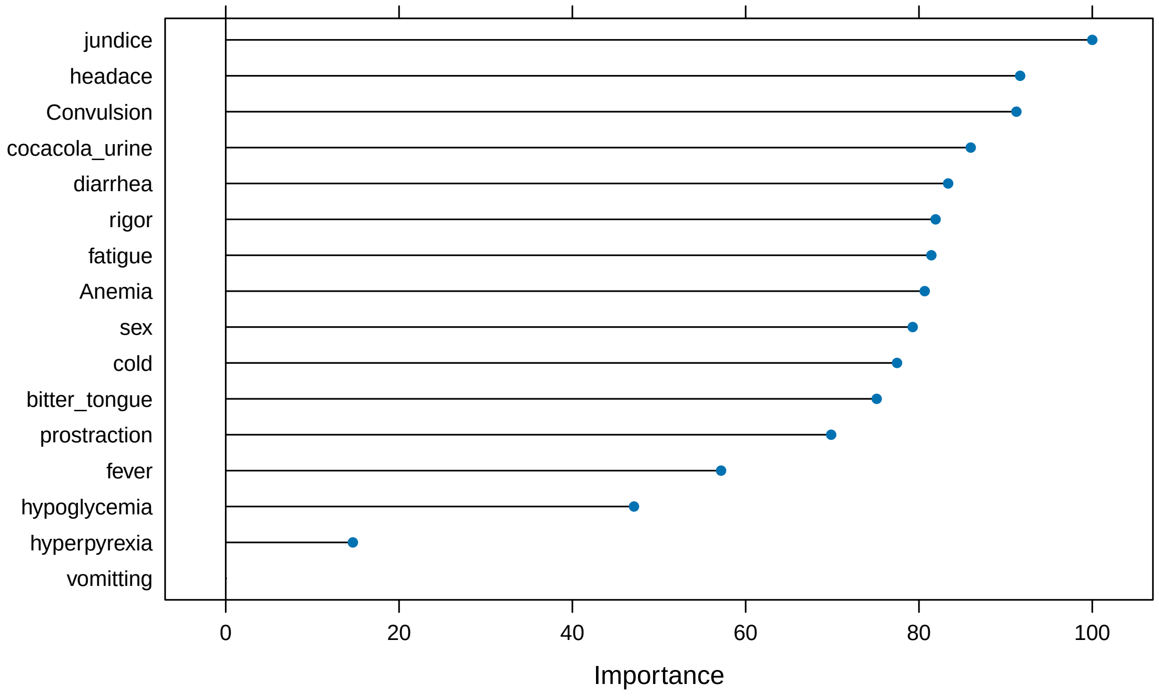
```
Confusion Matrix and Statistics

            Reference
Prediction   Not Severe Severe
  Not Severe         47     26
  Severe             19      8

               Accuracy : 0.55
                 95% CI : (0.4473, 0.6497)
    No Information Rate : 0.66
    P-Value [Acc > NIR] : 0.9915

                  Kappa : -0.0553

 Mcnemar's Test P-Value : 0.3711

            Sensitivity : 0.2353
            Specificity : 0.7121
         Pos Pred Value : 0.2963
         Neg Pred Value : 0.6438
              Precision : 0.2963
                 Recall : 0.2353
                     F1 : 0.2623
             Prevalence : 0.3400
         Detection Rate : 0.0800
   Detection Prevalence : 0.2700
      Balanced Accuracy : 0.4737

       'Positive' Class : Severe
```

Plotting confusion matrix

```r
fourfoldplot(rf.cM$table, col = rainbow(4), main = "RF Confusion Matrix")
```



```r
plot(varImp(rfModel, scale = TRUE))
```



SHAP (SHapley Additive exPlanations)

The Shapley value helps explain how much each feature contributes to the prediction made by a machine learning model. It provides a way to fairly distribute the "credit" for the model's output across all input features. By visualizing the SHAP plot, you can understand not only which features are important, but also how specific feature values that are driving predictions for individual cases.

Set seed for reproducibility

```r
set.seed(456)
```

Assuming lrModel is already trained : Convert the caret model to a Predictor object, separating the target variable

```r
predictorrf <- Predictor$new(rfModel,
                             data = train[, -which(names(train) == "severe_maleria")],
                             y = train$severe_maleria)
```

Select a single instance from the test set to explain Replace '1' with the index of any other instance if desired

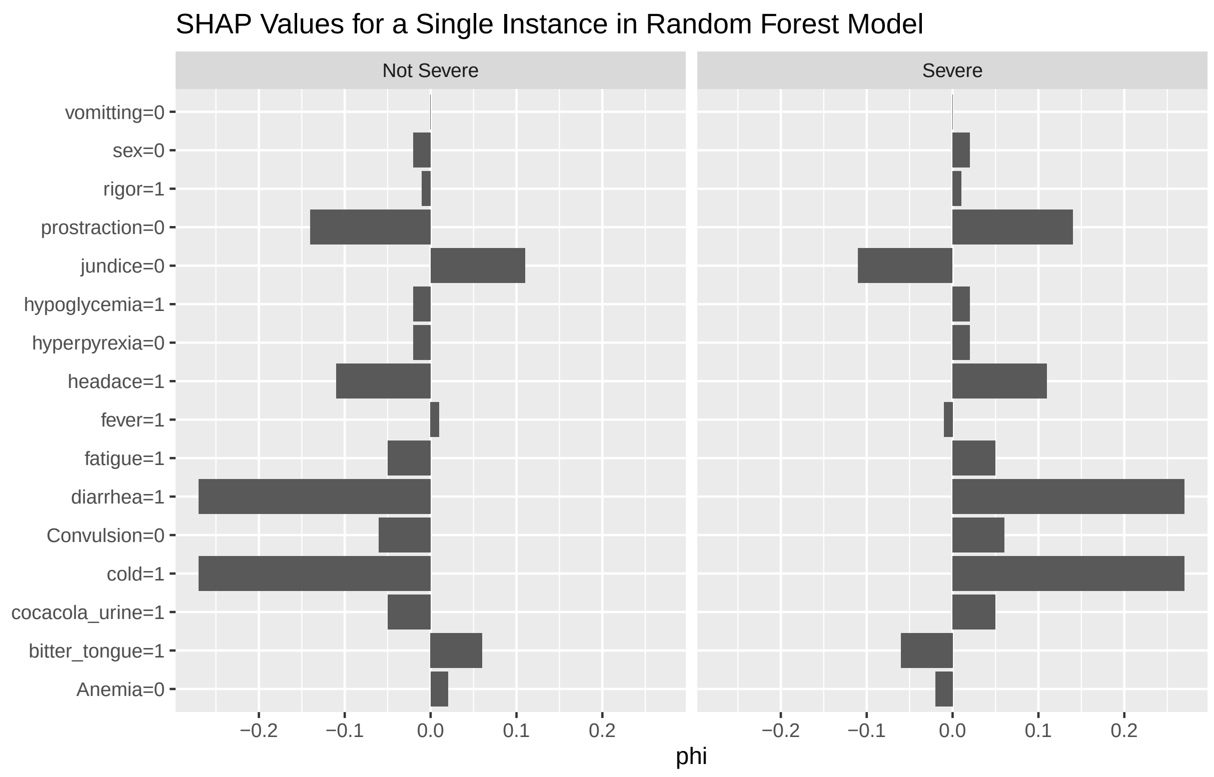
```r
x_interest <- test[1, -which(names(test) == "severe_maleria")]
```

Compute SHAP values for the specific instance

```r
shapleyrf <- Shapley$new(predictorrf, x.interest = x_interest)
```

Plot the SHAP values for this instance

```r
shapleyrf$plot() + ggtitle("SHAP Values for a Single Instance in Random Forest Model")
```



Leftward (negative): Indicates the feature is pushing the model prediction towards a negative class (e.g., “Not Severe”). Larger absolute SHAP values mean a feature has a stronger influence on the prediction. Smaller SHAP values (close to zero) indicate that a feature has minimal influence on the model’s output for that instance

TRY FOR OTHER MODELS

NAIVE BAYES

3. Train the Naive Bayes Classifier

```r
tic()
nbModel <- train(severe_maleria ~ .,
                 data = train,
                 method = "nb", trControl = control1)
toc()
```

```
6.17 sec elapsed
```

Predict on the test set

```r
nbpred <- predict(nbModel, newdata = test)
```

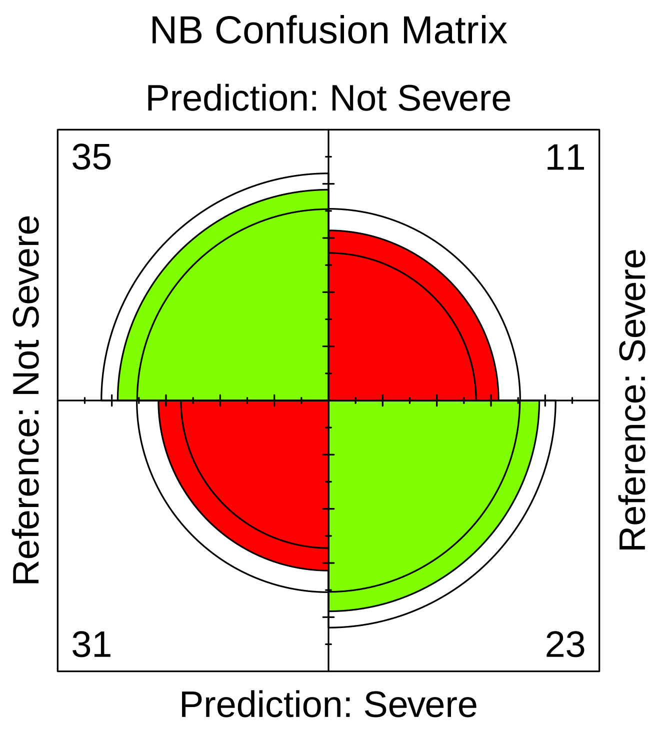
Evaluate with Confusion Matrix

```r
nb.cM <- confusionMatrix(nbpred,
                         as.factor(test$severe_maleria),
                         positive = "Severe",
                         mode = "everything")
print(nb.cM)
```

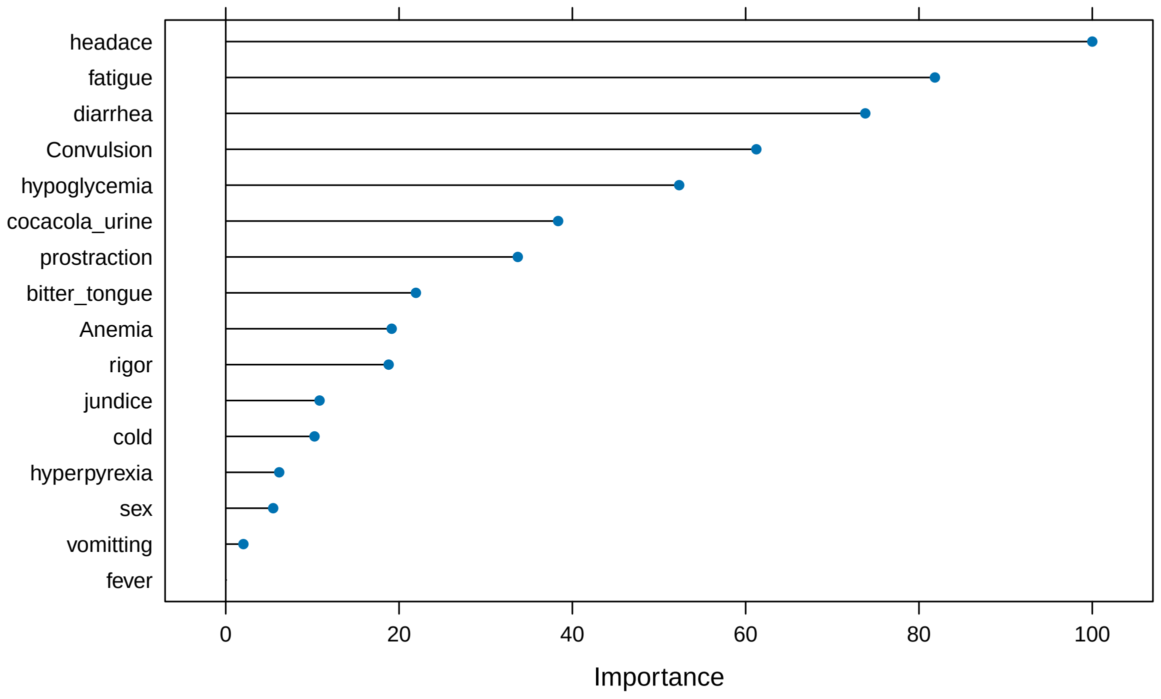
```
Confusion Matrix and Statistics

            Reference
Prediction   Not Severe Severe
  Not Severe         35     11
  Severe             31     23

               Accuracy : 0.58
                 95% CI : (0.4771, 0.678)
    No Information Rate : 0.66
    P-Value [Acc > NIR] : 0.96195

                  Kappa : 0.181

 Mcnemar's Test P-Value : 0.00337

            Sensitivity : 0.6765
            Specificity : 0.5303
         Pos Pred Value : 0.4259
         Neg Pred Value : 0.7609
              Precision : 0.4259
                 Recall : 0.6765
                     F1 : 0.5227
             Prevalence : 0.3400
         Detection Rate : 0.2300
   Detection Prevalence : 0.5400
      Balanced Accuracy : 0.6034

       'Positive' Class : Severe
```

Plotting confusion matrix

```r
fourfoldplot(nb.cM$table, col = rainbow(4),
             main = "NB Confusion Matrix")
```



```r
plot(varImp(nbModel, scale = TRUE))
```



SHAP (SHapley Additive exPlanations)

The Shapley value helps explain how much each feature contributes to the prediction made by a machine learning model. It provides a way to fairly distribute the "credit" for the model's output across all input features. By visualizing the SHAP plot, you can understand not only which features are important, but also how specific feature values that are driving predictions for individual cases.

Set seed for reproducibility

```r
set.seed(456)
```

Assuming lrModel is already trained : Convert the caret model to a Predictor object, separating the target variable

```r
predictornb <- Predictor$new(nbModel,
                             data = train[, -which(names(train) == "severe_maleria")],
                             y = train$severe_maleria)
```

Select a single instance from the test set to explain. Replace '1' with the index of any other instance if desired

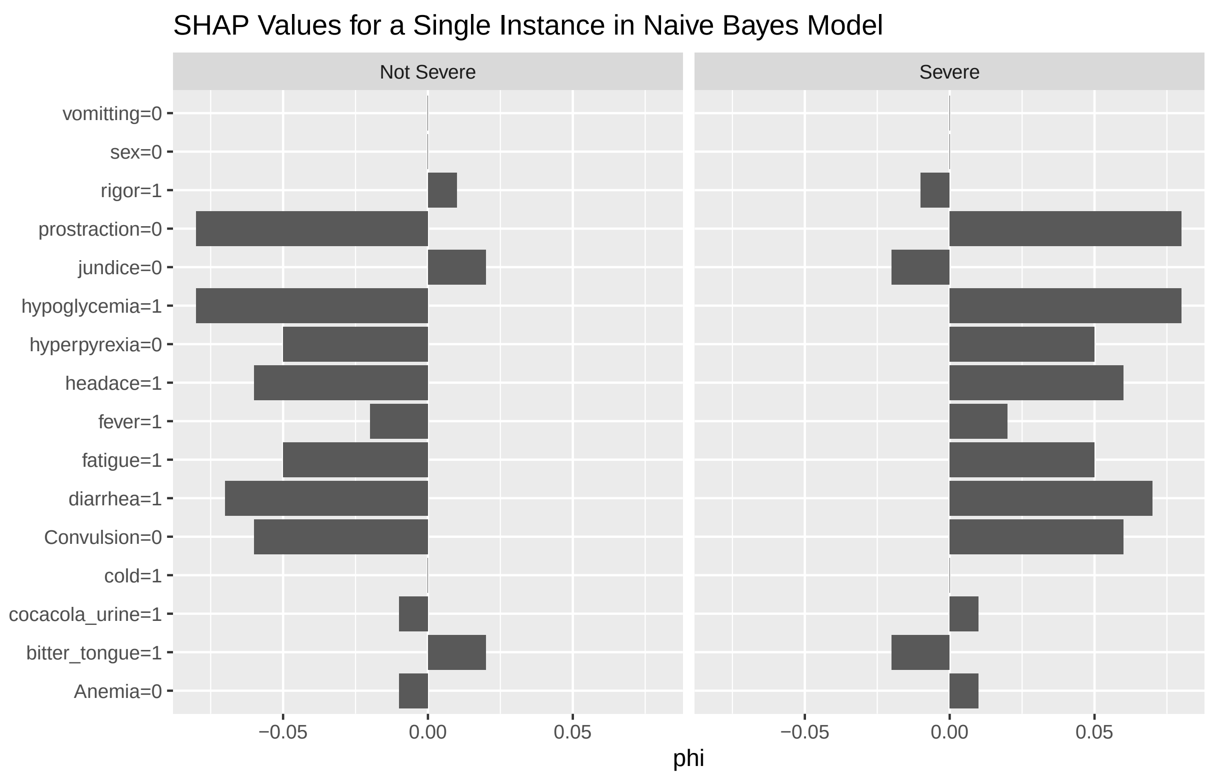
```r
x_interest <- test[1, -which(names(test) == "severe_maleria")]
```

Compute SHAP values for the specific instance

```r
shapleynb <- Shapley$new(predictornb, x.interest = x_interest)
```

Plot the SHAP values for this instance

```r
shapleynb$plot() +
  ggtitle("SHAP Values for a Single Instance in Naive Bayes Model")
```



Leftward (negative): Indicates the feature is pushing the model prediction towards a negative class (e.g., "Not Severe"). Larger absolute SHAP values mean a feature has a stronger influence on the prediction. Smaller SHAP values (close to zero) indicate that a feature has minimal influence on the model's output for that instance

DECISION TREE

3. Train the Decision Tree Classifier

```
tic()
DTModel <- train(factor(severe_maleria) ~ .,
                 data = train,

                 method = "rpart", trControl = control1)
toc()
```

```
1.97 sec elapsed
```

Predict on the test set

```
DTpred <- predict(DTModel, newdata = test)
```

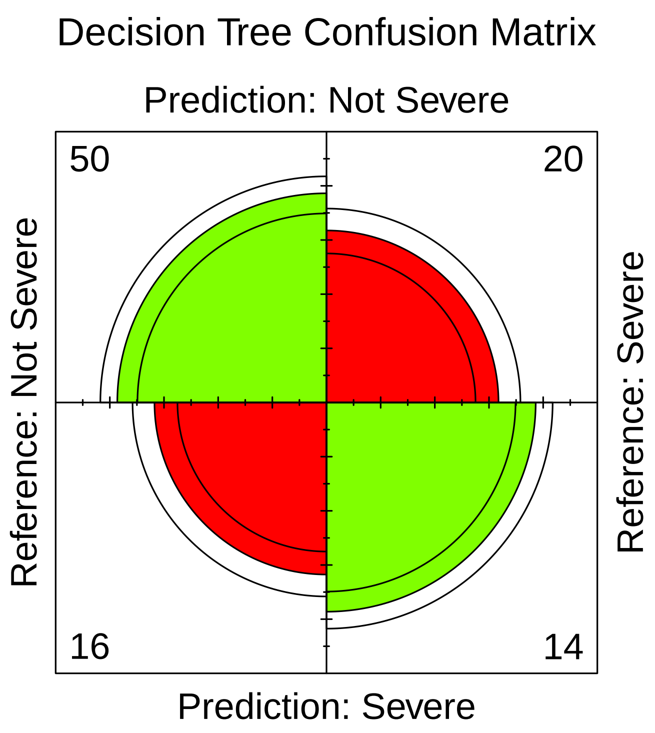
Evaluate with Confusion Matrix

```
DT.cM <- confusionMatrix(DTpred,
                         as.factor(test$severe_maleria),
                         positive = "Severe",
                         mode = "everything")
print(DT.cM)
```

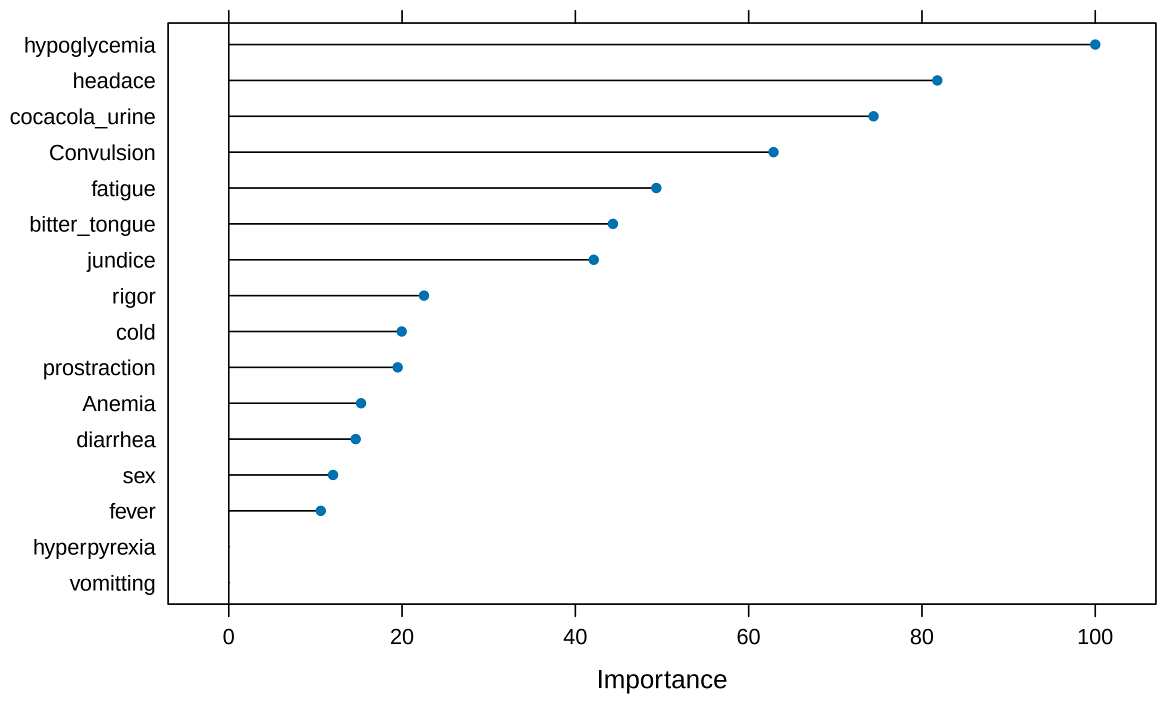
```
Confusion Matrix and Statistics

            Reference
Prediction   Not Severe Severe
  Not Severe         50     20
  Severe             16     14

               Accuracy : 0.64
                 95% CI : (0.5379, 0.7336)
    No Information Rate : 0.66
    P-Value [Acc > NIR] : 0.7039

                  Kappa : 0.1743

 Mcnemar's Test P-Value : 0.6171

            Sensitivity : 0.4118
            Specificity : 0.7576
         Pos Pred Value : 0.4667
         Neg Pred Value : 0.7143
              Precision : 0.4667
                 Recall : 0.4118
                     F1 : 0.4375
             Prevalence : 0.3400
         Detection Rate : 0.1400
   Detection Prevalence : 0.3000
      Balanced Accuracy : 0.5847

       'Positive' Class : Severe
```

Plotting confusion matrix

```
fourfoldplot(DT.cM$table, col = rainbow(4),
             main = "Decision Tree Confusion Matrix")
```



```r
plot(varImp(DTModel, scale = TRUE))
```



SHAP (SHapley Additive exPlanations)

The Shapley value helps explain how much each feature contributes to the prediction made by a machine learning model. It provides a way to fairly distribute the "credit" for the model's output across all input features. By visualizing the SHAP plot, you can understand not only which features are important, but also how specific feature values that are driving predictions for individual cases.

Set seed for reproducibility

```
set.seed(456)
```

Assuming lrModel is already trained : Convert the caret model to a Predictor object, separating the target variable

```
predictorDT <- Predictor$new(DTModel,
                             data = train[, -which(names(train) == "severe_maleria")],
                             y = train$severe_maleria)
```

Select a single instance from the test set to explain Replace '1' with the index of any other instance if desired

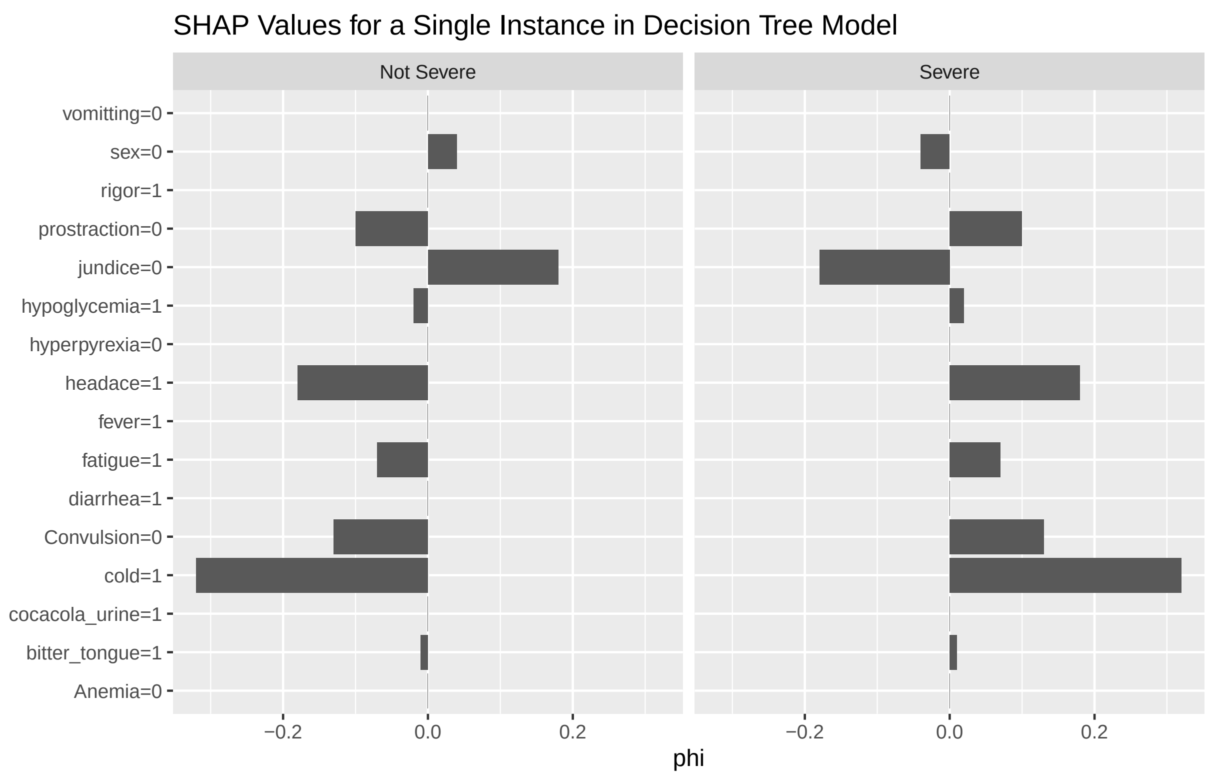
```
x_interest <- test[1, -which(names(test) == "severe_maleria")]
```

Compute SHAP values for the specific instance

```
shapleyDT <- Shapley$new(predictorDT, x.interest = x_interest)
```

Plot the SHAP values for this instance

```
shapleyDT$plot() +
  ggtitle("SHAP Values for a Single Instance in Decision Tree Model")
```



Leftward (negative): Indicates the feature is pushing the model prediction towards a negative class (e.g., "Not Severe"). Larger absolute SHAP values mean a feature has a stronger influence on the prediction. Smaller SHAP values (close to zero) indicate that a feature has minimal influence on the model's output for that instance

KNN

3. Train the K-Nearest Neighbors Classifier

```r
tic()
knnModel <- train(factor(severe_maleria) ~ .,
                 data = train,

                 method = "knn", trControl = control1)
toc()
```

```
2.35 sec elapsed
```

Predict on the test set

```r
knnpred <- predict(knnModel, newdata = test)
```

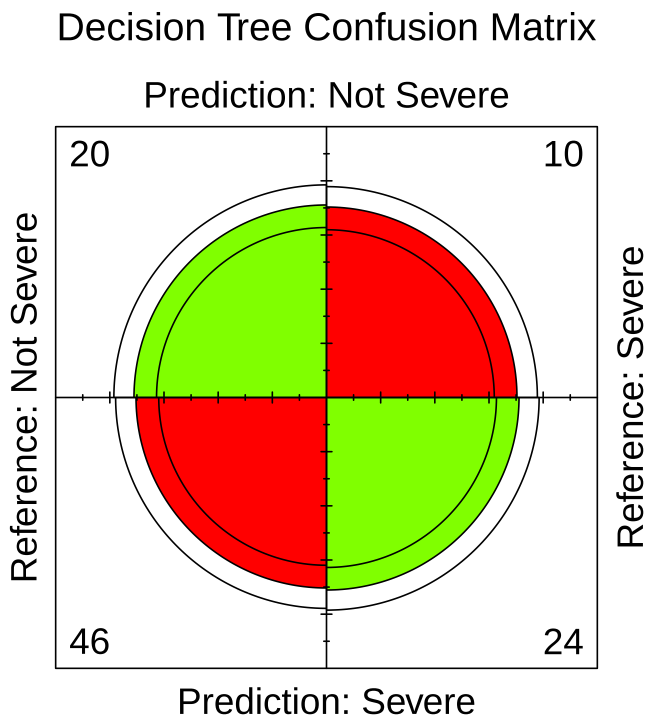
Evaluate with Confusion Matrix

```r
knn.cM <- confusionMatrix(knnpred,
                         as.factor(test$severe_maleria),
                         positive = "Severe",
                         mode = "everything")
print(knn.cM)
```

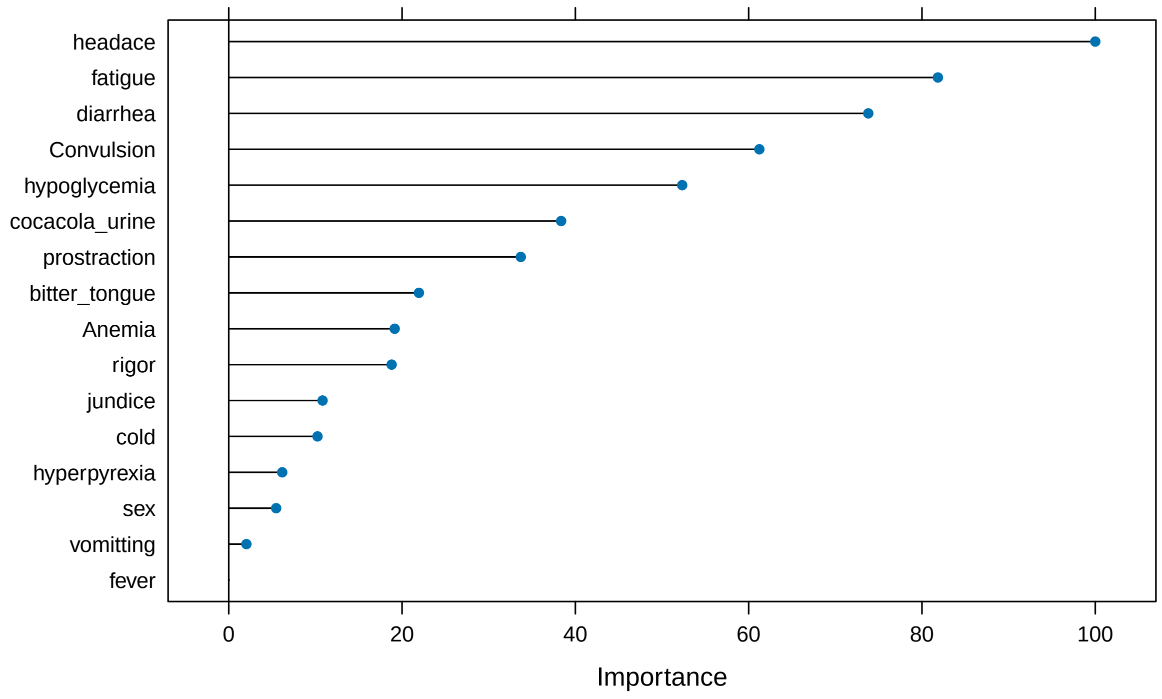
```
Confusion Matrix and Statistics

            Reference
Prediction   Not Severe Severe
  Not Severe         20     10
  Severe             46     24

               Accuracy : 0.44
                 95% CI : (0.3408, 0.5428)
    No Information Rate : 0.66
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0071

 Mcnemar's Test P-Value : 2.91e-06

            Sensitivity : 0.7059
            Specificity : 0.3030
         Pos Pred Value : 0.3429
         Neg Pred Value : 0.6667
              Precision : 0.3429
                 Recall : 0.7059
                     F1 : 0.4615
             Prevalence : 0.3400
         Detection Rate : 0.2400
   Detection Prevalence : 0.7000
      Balanced Accuracy : 0.5045

       'Positive' Class : Severe
```

Plotting confusion matrix

```r
fourfoldplot(knn.cM$table, col = rainbow(4),
             main = "Decision Tree Confusion Matrix")
```



```r
plot(varImp(knnModel, scale = TRUE))
```



SHAP (SHapley Additive exPlanations)

The Shapley value helps explain how much each feature contributes to the prediction made by a machine learning model. It provides a way to fairly distribute the "credit" for the model's output across all input features. By visualizing the SHAP plot, you can understand not only which features are important,but also how specific feature values that are driving predictions for individual cases.

Set seed for reproducibility

```r
set.seed(456)
```

Assuming lrModel is already trained : Convert the caret model to a Predictor object, separating the target variable

```r
predictorknn <- Predictor$new(knnModel,
                              data = train[, -which(names(train) == "severe_maleria")],
                              y = train$severe_maleria)
```

Select a single instance from the test set to explain. Replace '1' with the index of any other instance if desired

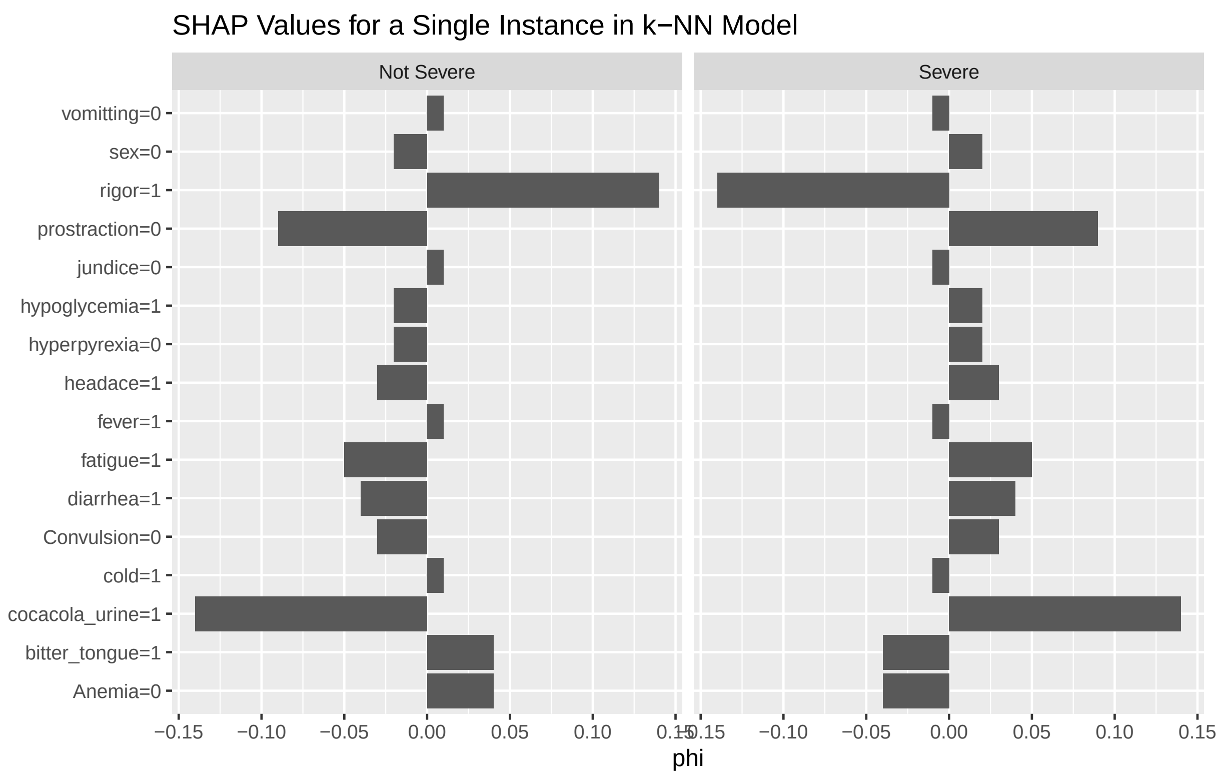
```r
x_interest <- test[1, -which(names(test) == "severe_maleria")]
```

Compute SHAP values for the specific instance

```r
shapleyknn <- Shapley$new(predictorknn, x.interest = x_interest)
```

Plot the SHAP values for this instance

```r
shapleyknn$plot() +
  ggtitle("SHAP Values for a Single Instance in k-NN Model")
```



Leftward (negative): Indicates the feature is pushing the model prediction towards a negative class (e.g., "Not Severe"). Larger absolute SHAP values mean a feature has a stronger influence on the prediction. Smaller SHAP values (close to zero) indicate that a feature has minimal influence on the model's output for that instance

SVM

3. Train the Support Vector Machines Classifier

```r
tic()
svmModel <- train(factor(severe_maleria) ~ .,
                  data = train,

                  method = "svmRadial", trControl = control1)
toc()
```

```
7.07 sec elapsed
```

Predict on the test set

```r
svmpred <- predict(svmModel, newdata = test)
```

Evaluate with Confusion Matrix

```
svm.cM <- confusionMatrix(svmpred,
                          as.factor(test$severe_maleria),
                          positive = "Severe",
                          mode = "everything")
print(svm.cM)
```

```
Confusion Matrix and Statistics

            Reference
Prediction   Not Severe Severe
  Not Severe         47     23
  Severe             19     11

               Accuracy : 0.58
                 95% CI : (0.4771, 0.678)
    No Information Rate : 0.66
    P-Value [Acc > NIR] : 0.9620

                  Kappa : 0.0367

 Mcnemar's Test P-Value : 0.6434

            Sensitivity : 0.3235
            Specificity : 0.7121
         Pos Pred Value : 0.3667
         Neg Pred Value : 0.6714
              Precision : 0.3667
                 Recall : 0.3235
                     F1 : 0.3438
             Prevalence : 0.3400
         Detection Rate : 0.1100
   Detection Prevalence : 0.3000
      Balanced Accuracy : 0.5178

       'Positive' Class : Severe
```

Plotting confusion matrix

fourfoldplot(svm.cM$table, col = rainbow(4), main = "Decision Tree Confusion Matrix") plot(varImp(svmModel, scale = TRUE))

SHAP (SHapley Additive exPlanations)

The Shapley value helps explain how much each feature contributes to the prediction made by a machine learning model. It provides a way to fairly distribute the "credit" for the model's output across all input features. By visualizing the SHAP plot, you can understand not only which features are important, but also how specific feature values that are driving predictions for individual cases.

Set seed for reproducibility

```r
set.seed(456)
```

Assuming lrModel is already trained : Convert the caret model to a Predictor object, separating the target variable

```r
predictorsvm <- Predictor$new(svmModel,
                              data = train[, -which(names(train) == "severe_maleria")],
                              y = train$severe_maleria)
```

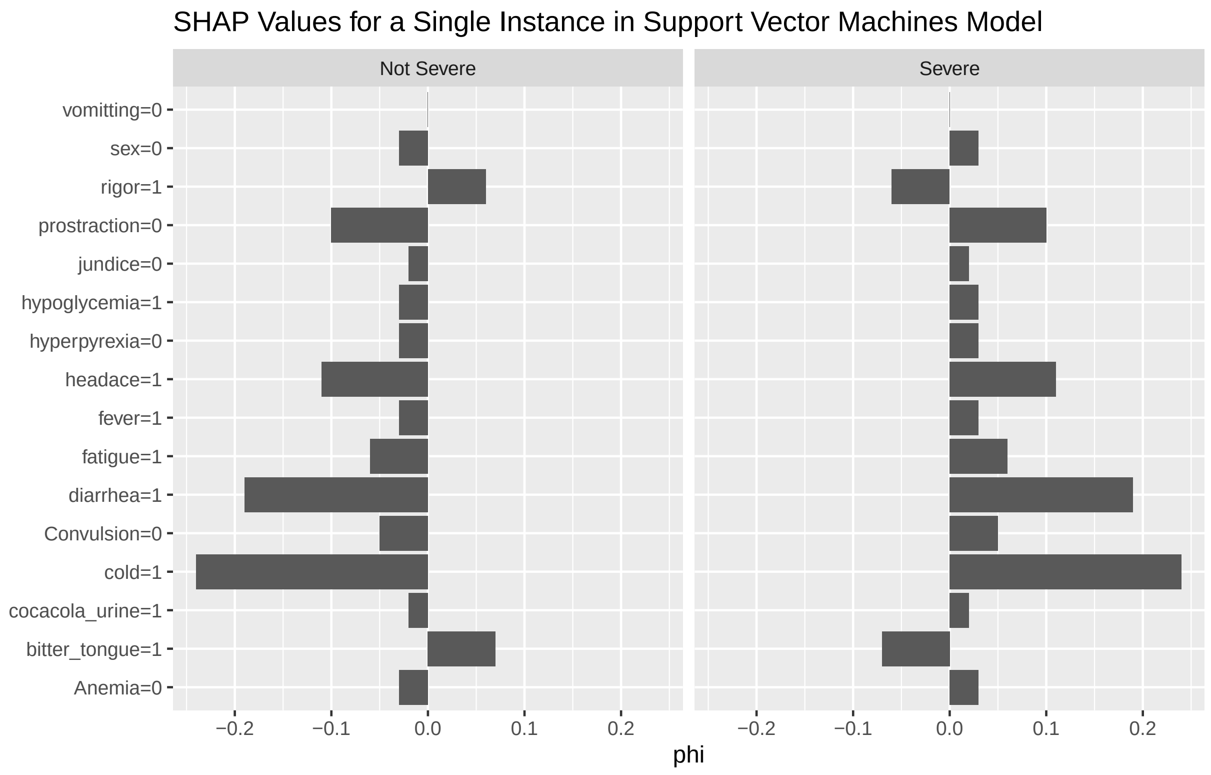
Select a single instance from the test set to explain. Replace '1' with the index of any other instance if desired x_interest <- test[1, -which(names(test) == "severe_maleria")]

Compute SHAP values for the specific instance

```r
shapleysvm <- Shapley$new(predictorsvm, x.interest = x_interest)
```

Plot the SHAP values for this instance

```r
shapleysvm$plot() +
  ggtitle("SHAP Values for a Single Instance in Support Vector Machines Model")
```



Leftward (negative): Indicates the feature is pushing the model prediction towards a negative class (e.g., "Not Severe"). Larger absolute SHAP values mean a feature has a stronger influence on the prediction. Smaller SHAP values (close to zero) indicate that a feature has minimal influence on the model's output for that instance

TREE BAG MODEL

3. Train the Tree Bag Classifier

```r
tic()
TbagModel <- train(factor(severe_maleria) ~ .,
                 data = train,
                 method = "treebag", trControl = control1)
toc()
```

```
10.5 sec elapsed
```

Predict on the test set

```r
Tbagpred <- predict(TbagModel, newdata = test)
```

Evaluate with Confusion Matrix

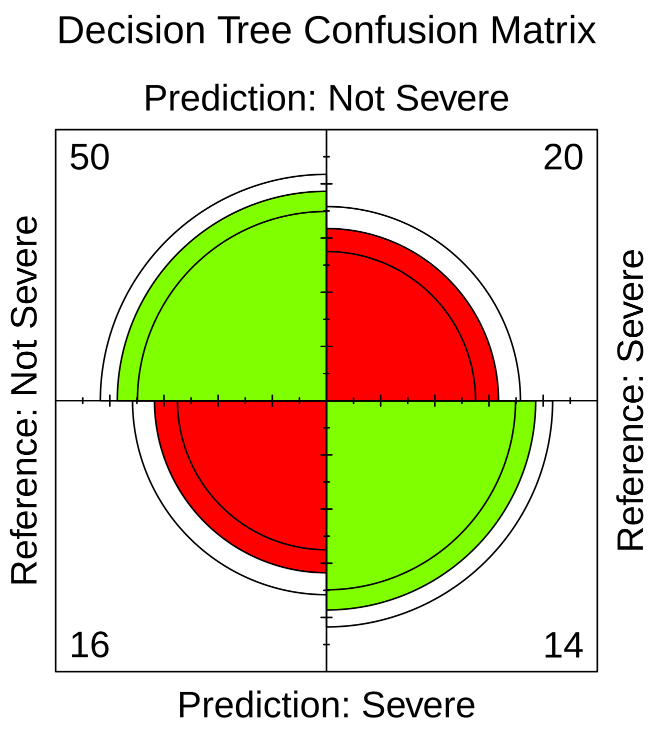
```r
Tbag.cM <- confusionMatrix(Tbagpred,
                         as.factor(test$severe_maleria),
                         positive = "Severe",
                         mode = "everything")
print(Tbag.cM)
```

```
Confusion Matrix and Statistics

            Reference
Prediction   Not Severe Severe
  Not Severe         48     25
  Severe             18      9

               Accuracy : 0.57
                 95% CI : (0.4671, 0.6686)
    No Information Rate : 0.66
    P-Value [Acc > NIR] : 0.9760

                  Kappa : -0.0084

 Mcnemar's Test P-Value : 0.3602

            Sensitivity : 0.2647
            Specificity : 0.7273
```

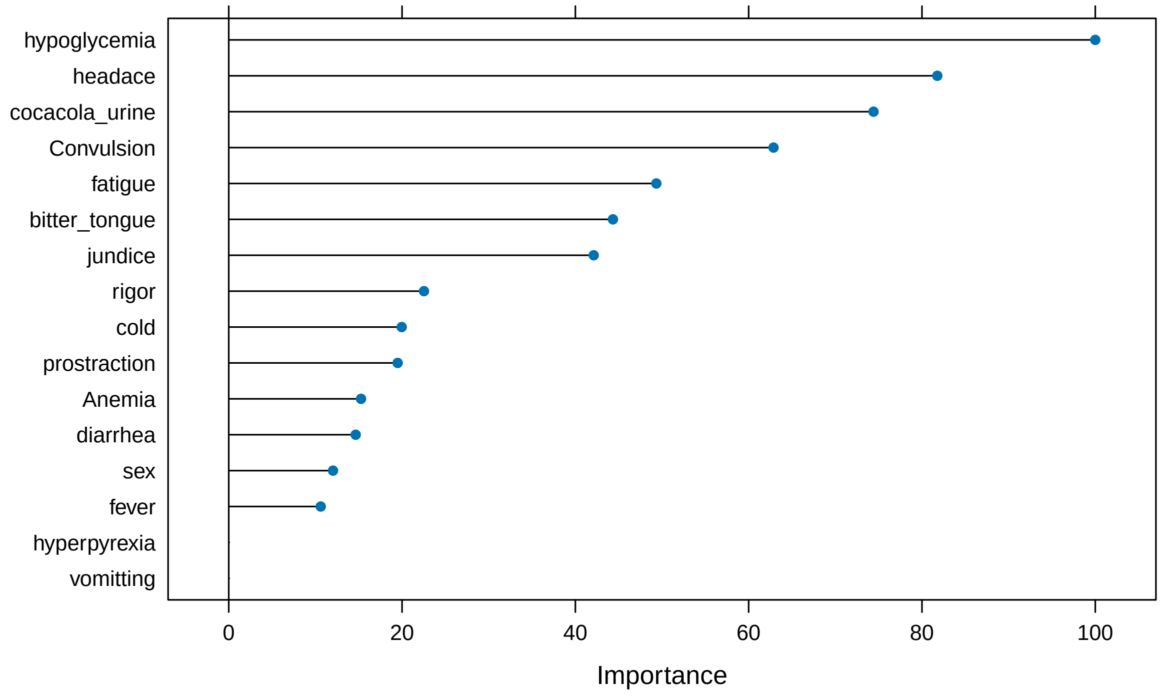
```
         Pos Pred Value : 0.3333
         Neg Pred Value : 0.6575
              Precision : 0.3333
                 Recall : 0.2647
                     F1 : 0.2951
             Prevalence : 0.3400
         Detection Rate : 0.0900
   Detection Prevalence : 0.2700
      Balanced Accuracy : 0.4960

       'Positive' Class : Severe
```

Plotting confusion matrix

```r
fourfoldplot(DT.cM$table, col = rainbow(4),
             main = "Decision Tree Confusion Matrix")
```



```r
plot(varImp(DTModel, scale = TRUE))
```



SHAP (SHapley Additive exPlanations)

The Shapley value helps explain how much each feature contributes to the prediction made by a machine learning model. It provides a way to fairly distribute the "credit" for the model's output across all input features. By visualizing the SHAP plot, you can understand not only which features are important, but also how specific feature values that are driving predictions for individual cases.

Set seed for reproducibility

```r
set.seed(456)
```

Assuming lrModel is already trained : Convert the caret model to a Predictor object, separating the target variable

```r
predictorTbag <- Predictor$new(TbagModel,
                               data = train[, -which(names(train) == "severe_maleria")],
                               y = train$severe_maleria)
```

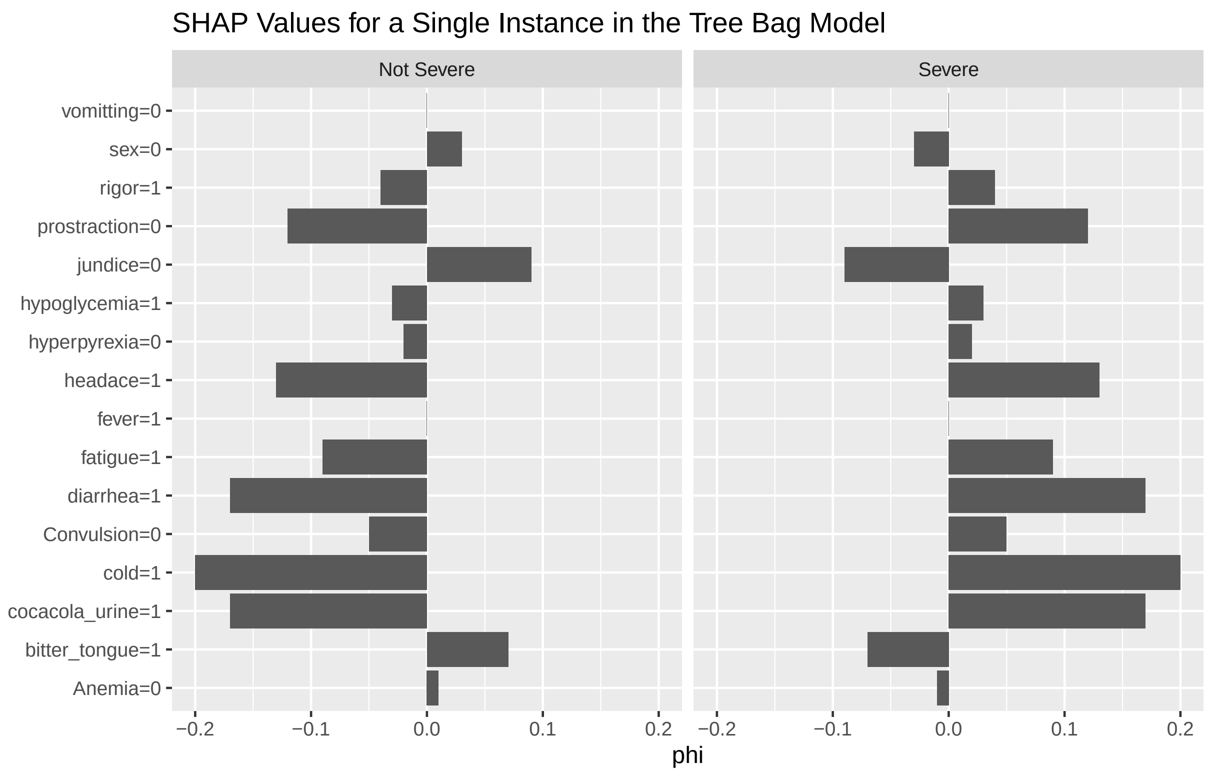
Select a single instance from the test set to explain. Replace '1' with the index of any other instance if desired x_interest <- test[1, -which(names(test) == "severe_maleria")]

Compute SHAP values for the specific instance

```r
shapleyTbag <- Shapley$new(predictorTbag, x.interest = x_interest)
```

Plot the SHAP values for this instance

```r
shapleyTbag$plot() +
  ggtitle("SHAP Values for a Single Instance in the Tree Bag Model")
```



Leftward (negative): Indicates the feature is pushing the model prediction towards a negative class (e.g., “Not Severe”). Larger absolute SHAP values mean a feature has a stronger influence on the prediction. Smaller SHAP values (close to zero) indicate that a feature has minimal influence on the model’s output for that instance