Tree & Forest

Mth 4330

Christopher Guzman, Dereck Lin, Haris Nioulikos, Samiha Uddin

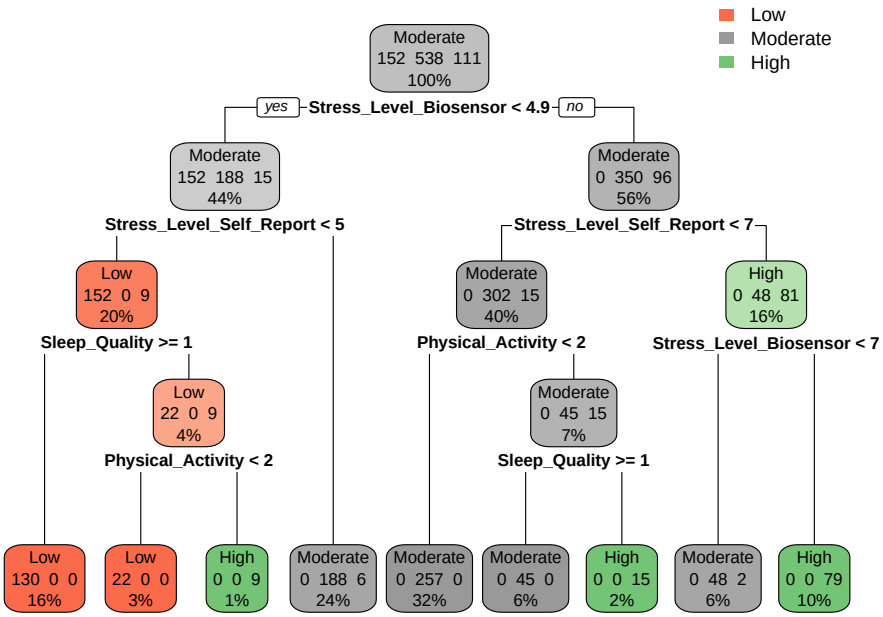
2025-12-13

```r
# Data Processing, same as from intermed rep
student_health_data <- read.csv('Data/student_health_data.csv')
#convert response label to an ordered factor
student_health_data$Health_Risk_Level <- factor(student_health_data$Health_Risk_Level,levels = c("Low", "Moderate", "High"),orde

#Encoding ordinal features, and the nominal feature gender
#cat features:  Gender, Physical_Activity, Sleep_Quality, Mood
student_health_data <- student_health_data %>%
  mutate(
    Gender = case_when(
      Gender == "M" ~ 1,
      Gender == "F" ~ 0
      ),
    Physical_Activity = case_when(
      Physical_Activity == "High" ~ 2,
      Physical_Activity == "Moderate" ~ 1,
      Physical_Activity == "Low" ~ 0
      ),
    Sleep_Quality = case_when(
      Sleep_Quality == "Good" ~ 2,
      Sleep_Quality == "Moderate" ~ 1,
      Sleep_Quality == "Poor" ~ 0
      ),
    Mood = case_when(
      Mood == "Happy" ~ 2,
      Mood == "Neutral" ~ 1,
      Mood == "Stressed" ~ 0
      )
    )

# remove identifier(Student_ID)
student_health_data$Student_ID <- NULL

#OVR encode into separate cols
classes <- levels(student_health_data$Health_Risk_Level)
for(class in classes){
  isClass <- paste0("y_", class)
  student_health_data[[isClass]] <- ifelse(student_health_data$Health_Risk_Level == class, 1, 0)
}


set.seed(1)#for reproducibility

# split into train&test(8:2)
# src: https://scikit-learn.org/stable/common_pitfalls.html
idx <- createDataPartition(student_health_data$Health_Risk_Level, p = 0.8, list = FALSE)
tr <- student_health_data[idx,]
te <- student_health_data[-idx,]

#cv split assignments
v <- sample(1:5, nrow(tr), replace = TRUE)
```

```r
my_folds_list <- lapply(1:5, function(i) which(v != i))
names(my_folds_list) <- paste0("Fold", 1:5)
#full features: ~ Age + Gender + Heart_Rate + Blood_Pressure_Systolic + Blood_Pressure_Diastolic + Stress_Level_Biosensor + Stres
```

Decision tree, tuned and features selected via cv

```r
dt_base <- rpart(
    Health_Risk_Level ~ Age + Gender + Heart_Rate + Blood_Pressure_Systolic + Blood_Pressure_Diastolic + Stress_Level_Biosensor
    data = tr,
    method = "class"
  )

#untuned dt visualized
rpart.plot(dt_base,
           type = 2,
           extra = 101,
           fallen.leaves = TRUE)
```




```r
#tr error
dt_base_tr_pred <-predict(dt_base, newdata = tr, type = "class")
tr_pred_labels <-
dt_base_tr_conf_mat <- table(Predicted = dt_base_tr_pred, Actual = tr$Health_Risk_Level)
dt_base_tr_conf_mat
```

```
##           Actual
## Predicted  Low Moderate High
##   Low      152        0    0
##   Moderate   0      538    8
##   High       0        0  103
```

```r
  #obv overfit, just learned the data

#te error
```

```r
dt_base_te_pred <-predict(dt_base, newdata = te, type = "class")
dt_base_te_conf_mat <- table(Predicted = dt_base_te_pred, Actual = te$Health_Risk_Level)
dt_base_te_conf_mat
```

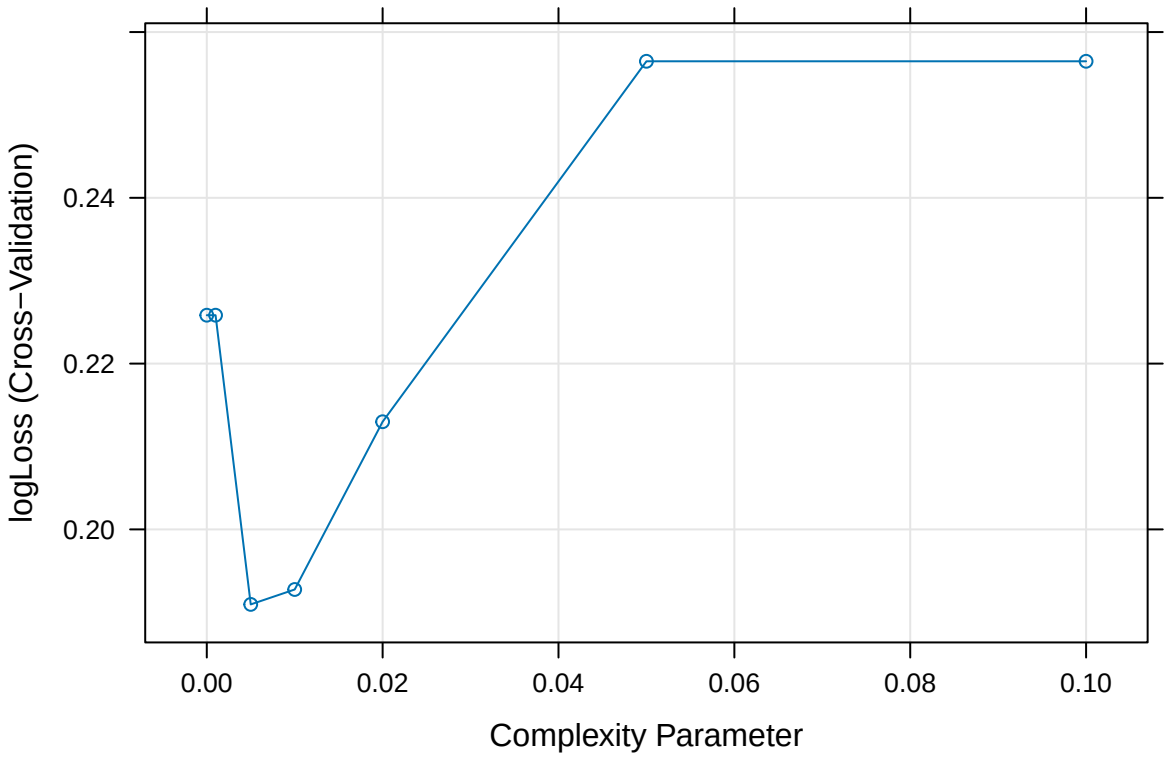
```
##           Actual
## Predicted  Low Moderate High
##   Low       36        0    0
##   Moderate   2      134    1
##   High       0        0   26
```

Decision tree, tuned and features selected via cv

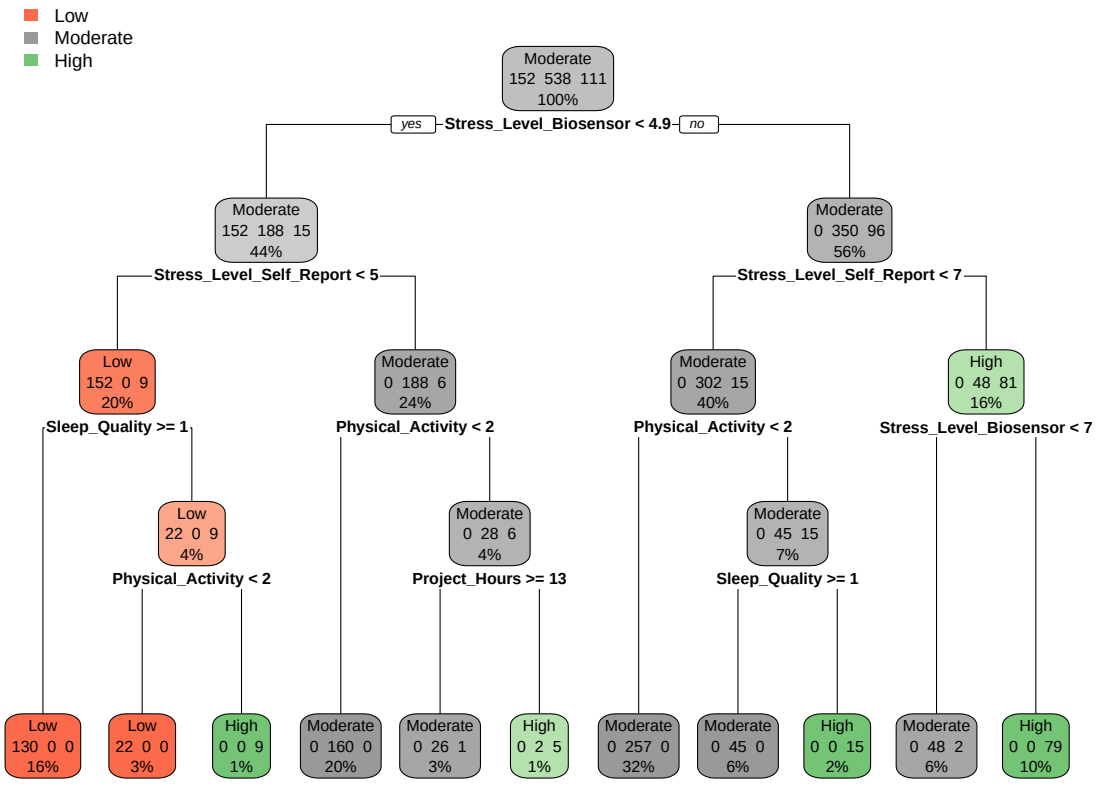
```r
#using v as split indexes for cv
trctrl <- trainControl(method = "cv", number = 5, index = my_folds_list, classProbs = TRUE, summaryFunction = mnLogLoss)
#complexity param from 0 to 1, most -> least complex
grid <- expand.grid(cp = c(0, 0.001, 0.005, 0.01, 0.02, 0.05, 0.1))

dt_cv <- train(Health_Risk_Level ~ Age + Gender + Heart_Rate + Blood_Pressure_Systolic + Blood_Pressure_Diastolic + Stress_Level
```

##sep chunk for vis and data stuff to sep from comp time

```r
#output best tree
#print(dt_cv)
plot(dt_cv)
```



```
rpart.plot(dt_cv$finalModel,
           type = 2,
           extra = 101,
           fallen.leaves = TRUE)
```




```
#convert
y_tr <- tr[,c(14,15,16)]
y_te <- te[,c(14,15,16)]

#tr error
dt_cv_tr_pred <-predict(dt_cv$finalModel, newdata = tr, type = "class")
dt_cv_tr_conf_mat <- table(Predicted = dt_cv_tr_pred, Actual = tr$Health_Risk_Level)
#dt_cv_tr_conf_mat

dt_cv_tr_pred_prob <-predict(dt_cv$finalModel, newdata = tr, type = "prob")
prob_safe <- pmax(as.matrix(dt_cv_tr_pred_prob), 1e-15)
prob_safe <- pmin(prob_safe, 1 - 1e-15)
dt_cv_tr_log_loss <- -mean(rowSums(y_tr * log(prob_safe)))
dt_cv_tr_log_loss
```

```
## [1] 0.02105141
```

```
#te error
dt_cv_te_pred <-predict(dt_cv$finalModel, newdata = te, type = "class")
dt_cv_te_conf_mat <- table(Predicted = dt_cv_te_pred, Actual = te$Health_Risk_Level)
#dt_cv_te_conf_mat

dt_cv_te_pred_prob <-predict(dt_cv$finalModel, newdata = te, type = "prob")
prob_safe <- pmax(as.matrix(dt_cv_te_pred_prob), 1e-15)
prob_safe <- pmin(prob_safe, 1 - 1e-15)
```

```r
dt_cv_te_log_loss <- -mean(rowSums(y_te * log(prob_safe)))
dt_cv_te_log_loss
```

```
## [1] 0.3981614
```

```r
##dt sensitivity on test set pred
matrix_te <- confusionMatrix(data = dt_cv_te_pred, reference = te$Health_Risk_Level)
matrix_te
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Low Moderate High
##   Low       36        0    0
##   Moderate   2      129    1
##   High       0        5   26
##
## Overall Statistics
##
##                Accuracy : 0.9598
##                  95% CI : (0.9223, 0.9825)
##     No Information Rate : 0.6734
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9192
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Low Class: Moderate Class: High
## Sensitivity              0.9474          0.9627      0.9630
## Specificity              1.0000          0.9538      0.9709
## Pos Pred Value           1.0000          0.9773      0.8387
## Neg Pred Value           0.9877          0.9254      0.9940
## Prevalence               0.1910          0.6734      0.1357
## Detection Rate           0.1809          0.6482      0.1307
## Detection Prevalence     0.1809          0.6633      0.1558
## Balanced Accuracy        0.9737          0.9583      0.9669
```

```r
mean(dt_cv$results$logLoss)
```

```
## [1] 0.2230442
```

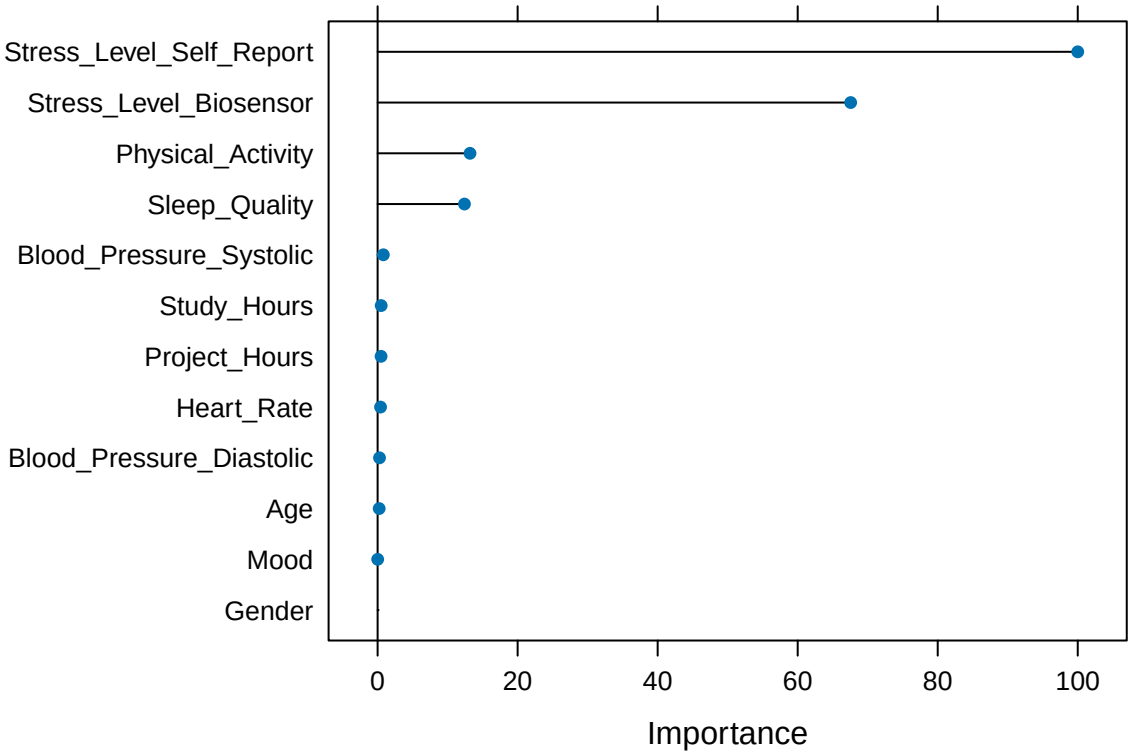
Rand forest

```r
rf_cv <- train(Health_Risk_Level ~ Age + Gender + Heart_Rate + Blood_Pressure_Systolic + Blood_Pressure_Diastolic + Stress_Level
```

```r
# rf_pred_tr <- predict(rf_cv, newdata = tr)
# confusionMatrix(data = rf_pred_tr, reference = tr$Health_Risk_Level)

rf_cv_logloss <-mean(rf_cv$results$logLoss)
rf_cv_logloss
```

```
## [1] 0.1296348
```

```r
rf_pre_te_prob <- predict(rf_cv, newdata = te, type = "prob")
rf_pre_te <- predict(rf_cv, newdata = te)

plot(varImp(rf_cv))
```




```
prob_safe <- pmax(as.matrix(rf_pre_te_prob), 1e-15)
prob_safe <- pmin(prob_safe, 1 - 1e-15)
rf_te_logloss <- -mean(rowSums(y_te * log(prob_safe)))
rf_te_logloss
```

```
## [1] 0.03953676
```

```
confusionMatrix(data = rf_pre_te, reference = te$Health_Risk_Level)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Low Moderate High
##   Low       36        0    0
##   Moderate   2      134    0
##   High       0        0   27
##
## Overall Statistics
##
##                Accuracy : 0.9899
##                  95% CI : (0.9642, 0.9988)
##     No Information Rate : 0.6734
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9794
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Low Class: Moderate Class: High
```

```
## Sensitivity             0.9474          1.0000      1.0000
## Specificity             1.0000          0.9692      1.0000
## Pos Pred Value          1.0000          0.9853      1.0000
## Neg Pred Value          0.9877          1.0000      1.0000
## Prevalence              0.1910          0.6734      0.1357
## Detection Rate          0.1809          0.6734      0.1357
## Detection Prevalence    0.1809          0.6834      0.1357
## Balanced Accuracy       0.9737          0.9846      1.0000
```