# 

# 

# **Advanced Topics in Health Care Data Analytics and Data Mining**

# 

# 

# 

# **GROUP 1**

# Peter Mukiibi

Yash Puri

Puja Yadav

Biniyam Abebe

Doreen Accorley

# 

# 

# 

# 

# 

# 

# 

# **Random Forest Model Tuning for Hospital Performance Classification**

# **Introduction**

The objective of this analysis was to build and optimize a Random Forest classification model to predict hospital performance status specifically identifying whether a hospital is high-performing (Y) or not (N). The predictor variables were derived from the Hospital Value-Based Purchasing (HVBP) program data. To improve model accuracy and reliability, hyperparameter tuning was conducted on three key parameters: mtry, maxnodes, and ntree.

## **Methodology**

### Data Preparation

• Dataset size: 589 observations  
 • Predictor variables: 6  
 • Target variable: HighPerf (binary: Y = High Performing, N = Not High Performing)  
 • The dataset was split into training and test sets to facilitate model evaluation after tuning.

### Step 1: Baseline Model and mtry Tuning

An initial Random Forest model was trained using default parameters. A grid search over mtry values from 1 to 4 was performed using 10-fold cross-validation.  
  
 • Best mtry: 2  
 • Cross-validated accuracy: 0.8136

### Step 2: Tuning maxnodes

Using the best mtry value (2), the maxnodes parameter was tuned from 30 to 100. A new model was trained at each value.  
  
 • Best maxnodes: 37  
 • Mean Accuracy: 0.8794  
 • Rationale: Chosen based on the highest balanced accuracy

### Step 3: Tuning ntree

With mtry = 2 and maxnodes = 37, the number of trees (ntree) was tuned across a range of values: 50 to 2000.  
  
 • Best ntree: 550  
 • Mean Accuracy: 0.8136

### **Final Model Training and Evaluation**

The final optimized model was trained with the following parameters:  
 • mtry = 2, maxnodes = 37, ntree = 550  
  
 Test Set Performance:  
 • Accuracy: 0.9253  
 • Kappa: 0.8432  
 • Sensitivity: 0.9574  
 • Specificity: 0.8776  
 • Balanced Accuracy: 0.9175

### **Correlation Analysis**

Pearson correlation coefficients were calculated between numerically encoded HighPerf and each predictor variable:

| **Predictor** | **Correlation** | **Interpretation** |
| --- | --- | --- |
| **EfficiencyCost** | **0.496** | **Strongest positive correlation** |
| **PatientExperience** | **0.425** | **Moderate positive correlation** |
| **ClinicalOutcome** | **0.387** | **Moderate positive correlation** |
| **ClinicalProcess** | **0.061** | **Weak correlation (not significant)** |
| **StructuralMsrs** | **-0.007** | **No correlation** |
| **HospOwnership (NonPub)** | **0.042** | **Very weak correlation** |

## 

## **Results and Interpretation**

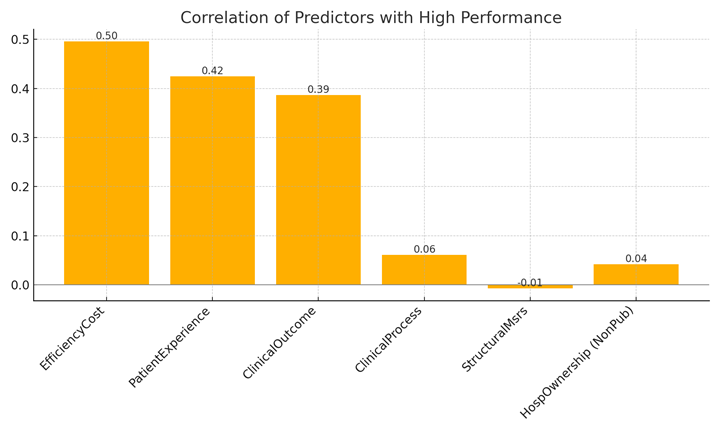
The step-by-step hyperparameter tuning process notably enhanced the model's classification performance. The final model achieved high accuracy and strong balance between sensitivity and specificity, making it suitable for identifying high-performing hospitals. Among the predictors, EfficiencyCost, PatientExperience, and ClinicalOutcome were most influential in predicting performance.

## **Conclusion**

A robust Random Forest model was successfully developed and optimized to classify hospitals based on HVBP metrics. Through systematic tuning of mtry, maxnodes, and ntree, we maximized model performance. Correlation analysis highlighted which features most strongly influenced classification, offering interpretability alongside predictive power.

## 

## **Correlation Chart**

****

**Figure: Correlation of individual predictors with hospital performance.**

## 

## **Appendix: R Code Snippets Used**

# Initial setup  
 set.seed(1234)  
 tuneGrid <- expand.grid(.mtry = c(1:4))  
 rf\_mtry <- train(HighPerf~., data = data\_train, method = "rf", metric = "Accuracy", tuneGrid = tuneGrid,  
 trControl = trControl, importance = TRUE, nodesize = 14, ntree = 500)  
  
 # Tuning maxnodes  
 store\_maxnode <- list()  
 tuneGrid <- expand.grid(.mtry = 2)  
 for (maxnodes in c(30:100)) {  
 rf\_maxnode <- train(HighPerf~., data = data\_train, method = "rf", metric = "Accuracy",  
 tuneGrid = tuneGrid, trControl = trControl, importance = TRUE,  
 nodesize = 14, maxnodes = maxnodes, ntree = 500)  
 store\_maxnode[[toString(maxnodes)]] <- rf\_maxnode  
 }  
  
 # Tuning ntree  
 store\_maxtrees <- list()  
 for (ntree in c(50, 100, 150, 250, 300, 350, 400, 450, 500, 550, 600, 800, 1000, 2000)) {  
 rf\_maxtrees <- train(HighPerf~., data = data\_train, method = "rf", metric = "Accuracy",  
 tuneGrid = tuneGrid, trControl = trControl, importance = TRUE,  
 nodesize = 14, maxnodes = 37, ntree = ntree)  
 store\_maxtrees[[toString(ntree)]] <- rf\_maxtrees  
 }  
  
 # Final model  
 fit\_rf <- train(HighPerf~., data = data\_train, method = "rf", metric = "Accuracy", tuneGrid = tuneGrid,  
 trControl = trControl, importance = TRUE, nodesize = 14, ntree = 550, maxnodes = 37)  
  
 # Predictions and evaluation  
 mypredicts <- predict(fit\_rf, data\_test)  
 confusionMatrix(mypredicts, factor(data\_test$HighPerf))  
  
 # Correlation analysis  
 data\_train$HighPerf\_Num <- ifelse(data\_train$HighPerf == "Y", 1, 0)  
 cor.test(data\_train$EfficiencyCost, data\_train$HighPerf\_Num)  
 cor.test(data\_train$PatientExperience, data\_train$HighPerf\_Num)  
 cor.test(data\_train$ClinicalOutcomeMeasures, data\_train$HighPerf\_Num)  
 cor.test(data\_train$ClinicalProcessMeasures, data\_train$HighPerf\_Num)  
 cor.test(data\_train$StructuralMsrs\_Yes\_Count, data\_train$HighPerf\_Num)  
 data\_train$HospOwnership\_Num <- ifelse(data\_train$HospOwnership == "NonPub", 1, 0)  
 cor.test(data\_train$HospOwnership\_Num, data\_train$HighPerf\_Num)

# Final model

fit\_rf <- train(HighPerf~., data = data\_train, method = "rf", metric = "Accuracy", tuneGrid = tuneGrid,

trControl = trControl, importance = TRUE, nodesize = 14, ntree = 550, maxnodes = 37)

# Predictions and evaluation

mypredicts <- predict(fit\_rf, data\_test)

confusionMatrix(mypredicts, factor(data\_test$HighPerf))

# Correlation

data\_train$HighPerf\_Num <- ifelse(data\_train$HighPerf == "Y", 1, 0)

cor.test(data\_train$EfficiencyCost, data\_train$HighPerf\_Num)