**Statistical Learning Lab**

Assignment - 7

**Tree Based Medhods**

Anurag Singha | 22IM30007 

# **Import the designated data file**

We’ll start by loading the necessary libraries and importing the Pulsar star dataset.

|  |  |  |
| --- | --- | --- |
| **library**(rpart) *# For decision trees* **library**(rpart.plot) *# For plotting decision trees* **library**(randomForest) *# For random forest* **library**(caret) *# For model evaluation* **library**(e1071) *# For model evaluation metrics* **library**(ggplot2) *# For data visualization* **library**(dplyr) *# Load dplyr package*  *# Download data from the URL and import it*  pulsar\_data <- **read.csv**("C:**\\**Users**\\**USER**\\**Downloads**\\**pulsar\_data\_train.csv**\\**pulsar\_data\_train  .csv"  )  *# Take a peek at the data*  **head**(pulsar\_data) | | |
| ## | Mean.of.the.integrated.profile | Standard.deviation.of.the.integrated.prof |
| ile |  |  |
| ## 1 | 121.15625 | 48.37 |
| 297 |  |  |
| ## 2 | 76.96875 | 36.17 |
| 556 |  |  |
| ## 3 | 130.58594 | 53.22 |
| 953 |  |  |
| ## 4 | 156.39844 | 48.86 |
| 594 |  |  |
| ## 5 | 84.80469 | 36.11 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 766  ## 6 | | 121.00781 | 47.17 | |
| 694  ##  file  ## 1 | | Excess.kurtosis.of.the.integrated.profile  0.3754847 | Skewness.of.the.integrated.pro  -0.0131 | |
| 6549  ## 2 | | 0.7128979 | 3.3887 | |
| 1856  ## 3 | | 0.1334083 | -0.2972 | |
| 4164  ## 4 | | -0.2159886 | -0.1712 | |
| 9365  ## 5 | | 0.8250128 | 3.2741 | |
| 2537  ## 6 | | 0.2297081 | 0.0913 | |
| 3623 | |  |  | |
| ## |  | Mean.of.the.DM.SNR.curve Standard.deviation.of.the.DM.SNR.curve | |  |
| ## | 1 | 3.168896 18.39937 | |  |
| ## | 2 | 2.399666 17.57100 | |  |
| ## | 3 | 2.743311 22.36255 | |  |
| ## | 4 | 17.471572 NA | |  |
| ## | 5 | 2.790134 20.61801 | |  |
| ## 6 2.036789 NA  ## Excess.kurtosis.of.the.DM.SNR.curve Skewness.of.the.DM.SNR.curve class | | | | target\_ |
| ##  0 ## | 1  2 | 7.449874  9.414652 | 65.159298  102.722975 | |
| 0  ## | 3 | 8.508364 | 74.031324 | |
| 0  ## | 4 | 2.958066 | 7.197842 | |
| 0  ## | 5 | 8.405008 | 76.291128 | |
| 0  ## | 6 | 9.546051 | 112.131721 | |
| 0 |  |  |  | |

# Data cleaning and pre-processing

Let’s examine the dataset structure and check for missing values or inconsistencies.

*# Check the structure of the dataset*

**str**(pulsar\_data)

## 'data.frame':

12528 obs. of 9 variables:

## $ Mean.of.the.integrated.profile : num 121.2 77 130.6 156.4

84.8 ...

## $ Standard.deviation.of.the.integrated.profile: num 48.4 36.2 53.2 48.9

36.1 ...

## $ Excess.kurtosis.of.the.integrated.profile : num 0.375 0.713 0.133 -0

.216 0.825 ...

## $ Skewness.of.the.integrated.profile : num -0.0132 3.3887 -0.29

72 -0.1713 3.2741 ...

## $ Mean.of.the.DM.SNR.curve : num 3.17 2.4 2.74 17.47

2.79 ...

## $ Standard.deviation.of.the.DM.SNR.curve : num 18.4 17.6 22.4 NA 20

.6 ...

## $ Excess.kurtosis.of.the.DM.SNR.curve : num 7.45 9.41 8.51 2.96

8.41 ...

## $ Skewness.of.the.DM.SNR.curve : num 65.2 102.7 74 7.2 76

.3 ...

## $ target\_class : num 0 0 0 0 0 0 0 0 0 0

...

*# Summary statistics*

**summary**(pulsar\_data)

## Mean.of.the.integrated.profile Standard.deviation.of.the.integrated.profi le

## Min. : 5.812 Min. :24.77

## 1st Qu.:100.871 1st Qu.:42.36

## Median :115.184 Median :46.93

## Mean :111.042 Mean :46.52

## 3rd Qu.:127.109 3rd Qu.:50.98

## Max. :189.734 Max. :91.81 ##

## Excess.kurtosis.of.the.integrated.profile Skewness.of.the.integrated.prof ile

## Min. :-1.7380 Min. :-1.7919

## 1st Qu.: 0.0247 1st Qu.:-0.1881

## Median : 0.2237 Median : 0.2033

## Mean : 0.4785 Mean : 1.7784

## 3rd Qu.: 0.4731 3rd Qu.: 0.9324

## Max. : 8.0695 Max. :68.1016

## NA's 1735

## Mean.of.the.DM.SNR.curve Standard.deviation.of.the.DM.SNR.curve ## Min. : 0.2132 Min. : 7.37

## 1st Qu.: 1.9105 1st Qu.: 14.40

## Median : 2.7926 Median : 18.41

## Mean : 12.6748 Mean : 26.35

## 3rd Qu.: 5.4132 3rd Qu.: 28.34

## Max. :222.4214 Max. :110.64

## NA's 1178

## Excess.kurtosis.of.the.DM.SNR.curve Skewness.of.the.DM.SNR.curve ## Min. :-3.139 Min. : -1.977

## 1st Qu.: 5.803 1st Qu.: 35.200

## Median : 8.451 Median : 83.126

## Mean : 8.333 Mean : 105.526

|  |  |  |
| --- | --- | --- |
| ## | 3rd Qu.:10.728 | 3rd Qu.: 139.998 |
| ## | Max. :34.540 | Max. :1191.001 |
| ## |  | NA's 625 |
| ## | target\_class |  |
| ## | Min. :0.00000 |  |
| ## | 1st Qu.:0.00000 |  |
| ## | Median :0.00000 |  |
| ## | Mean :0.09203 |  |
| ## | 3rd Qu.:0.00000 |  |
| ## | Max. :1.00000 |  |
| ## |  |  |
| *# Check for missing values*  **sum**(**is.na**(pulsar\_data)) ## [1] 3538  *# Remove all rows with any missing values # pulsar\_data <- na.omit(pulsar\_data)*  *# Replace NA values with group-specific means*  pulsar\_data\_imputed <- pulsar\_data **%>% group\_by**(target\_class) **%>%**  **mutate**(**across**(**where**(is.numeric), **~ifelse**(**is.na**(.), **mean**(., na.rm = TRUE), .  ))) **%>%**  **ungroup**() **sum**(**is.na**(pulsar\_data\_imputed))  ## [1] 0  pulsar\_data <- pulsar\_data\_imputed | | |

# Identify a response variable

The target variable in this dataset is ‘target\_class’, which indicates whether a given sample is a pulsar star (1) or not (0).

## 0.90796616 0.09203384

1

0

##

##

**prop.table**(**table**(pulsar\_data**$**target\_class))

1

1153

## 11375

0

##

##

*# Display the distribution of the target variable*

**table**(pulsar\_data**$**target\_class)

# Convert categorical inputs or consider it while fitting the data

Let’s check for categorical variables and convert them as needed.

*# Check column types*

**sapply**(pulsar\_data, class)

##

##

Mean.of.the.integrated.profile

"numeric"

## Standard.deviation.of.the.integrated.profile

## "numeric"

## Excess.kurtosis.of.the.integrated.profile ## "numeric"

## Skewness.of.the.integrated.profile

## "numeric"

## Mean.of.the.DM.SNR.curve

## "numeric"

## Standard.deviation.of.the.DM.SNR.curve ## "numeric"

## Excess.kurtosis.of.the.DM.SNR.curve

## "numeric"

## Skewness.of.the.DM.SNR.curve

## "numeric"

## target\_class

## "numeric"

*# Convert target\_class to factor for classification*

pulsar\_data**$**target\_class <- **as.factor**(pulsar\_data**$**target\_class)

# Fit a classification and regression model

We’ll split the data into training and testing sets, then fit a decision tree model.

*# Set seed for reproducibility*

**set.seed**(123)

*# Create a data partition*

train\_index <- **createDataPartition**(pulsar\_data**$**target\_class, p = 0.7, list = FALSE)

train\_data <- pulsar\_data[train\_index, ] test\_data <- pulsar\_data[**-**train\_index, ]

*# Fit a decision tree model*

tree\_model <- **rpart**(target\_class **~** ., data = train\_data, method = "class")

*# Display the model summary*

**printcp**(tree\_model)

##

## Classification tree:

## rpart(formula = target\_class ~ ., data = train\_data, method = "class") ##

## Variables actually used in tree construction: ## [1] Excess.kurtosis.of.the.integrated.profile

##

## Root node error: 808/8771 = 0.092122 ##

## n= 8771 ##

## CP nsplit rel error xerror xstd ## 1 0.78713 0 1.00000 1.00000 0.033520

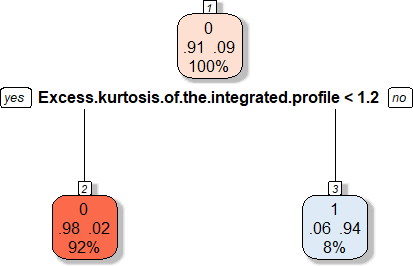
## 2 0.01000 1 0.21287 0.21658 0.016208

# Plot the decision tree for fitted model

Let’s visualize the decision tree to understand how it makes classifications.

*# Plot the decision tree*

**rpart.plot**(tree\_model, extra = 104, box.palette = "RdBu", shadow.col = "gray", nn = TRUE)



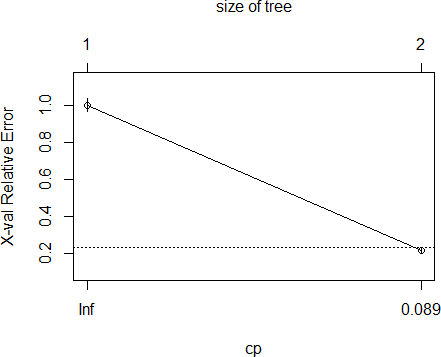
# Prune the tree by changing the best value

Now we’ll prune the tree to avoid overfitting by selecting the optimal complexity parameter

(cp).

*# Plot the error vs cp to find the optimal pruning point*

**plotcp**(tree\_model)



*# Find the CP with minimum error*

best\_cp <- tree\_model**$**cptable[**which.min**(tree\_model**$**cptable[,"xerror"]),"CP"]

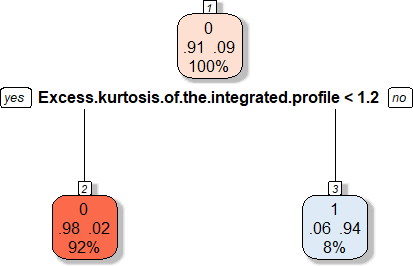
**cat**("Best CP value:", best\_cp, "**\n**") ## Best CP value: 0.01

*# Prune the tree using the best CP*

pruned\_tree <- **prune**(tree\_model, cp = best\_cp)

*# Plot the pruned tree*

**rpart.plot**(pruned\_tree, extra = 104, box.palette = "RdBu", shadow.col = "gray", nn = TRUE)



# Observe the results by calculating the misclassification rate or accuracy

We’ll evaluate the performance of our pruned decision tree model.

*# Predictions on test data*

tree\_pred <- **predict**(pruned\_tree, test\_data, type = "class")

*# Create a confusion matrix*

conf\_matrix <- **confusionMatrix**(tree\_pred, test\_data**$**target\_class) conf\_matrix

## Confusion Matrix and Statistics ##

## Reference

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ## | Prediction | 0 | 1 | |
| ## | 0 | 3392 | 40 | |
| ##  ## | 1 | 20 | 305 | |
| ## | Accuracy : | | | 0.984 |
| ## | 95% CI : | | | (0.9795, 0.9878) |
| ## | No Information Rate : | | | 0.9082 |
| ##  ##  ## | P-Value [Acc > NIR] :  Kappa : | | | < 2e-16  0.9017 |
| ## |  | | |  |

|  |  |  |
| --- | --- | --- |
| ## | Mcnemar's Test P-Value : | 0.01417 |
| ## |  |  |
| ## | Sensitivity : | 0.9941 |
| ## | Specificity : | 0.8841 |
| ## | Pos Pred Value : | 0.9883 |
| ## | Neg Pred Value : | 0.9385 |
| ## | Prevalence : | 0.9082 |
| ## | Detection Rate : | 0.9028 |
| ## | Detection Prevalence : | 0.9135 |
| ## | Balanced Accuracy : | 0.9391 |
| ## |  |  |
| ## | 'Positive' Class : | 0 |
| ## |  |  |
| *# Calculate accuracy and misclassification rate* accuracy <- conf\_matrix**$**overall["Accuracy"] misclass\_rate <- 1 **-** accuracy  **cat**("Accuracy:", **round**(accuracy **\*** 100, 2), "%**\n**") ## Accuracy: 98.4 %  **cat**("Misclassification rate:", **round**(misclass\_rate **\*** 100, 2), "%**\n**")  ## Misclassification rate: 1.6 % | | |

# 6. Fit a bagging and random forest model

Let’s implement bagging (bootstrap aggregating) and random forest models.

*# Bagging is a special case of random forest where mtry = total number of pre dictors*

n\_predictors <- **ncol**(train\_data) **-** 1 *# Exclude the target variable*

*# Fit a bagging model (Random Forest with mtry = all predictors)*

bagging\_model <- **randomForest**(target\_class **~** ., data = train\_data,

mtry = n\_predictors, importance = TRUE, ntree = 500)

*# Print model summary*

**print**(bagging\_model)

##

## Call:

## randomForest(formula = target\_class ~ ., data = train\_data, mtry = n\_pred ictors, importance = TRUE, ntree = 500)

## Type of random forest: classification ## Number of trees: 500

## No. of variables tried at each split: 8 ##

## OOB estimate of error rate: 1.86% ## Confusion matrix:

## 0 1 class.error

## 0 7915 48 0.006027879

## 1 115 693 0.142326733

*# Evaluate bagging model*

bagging\_pred <- **predict**(bagging\_model, test\_data)

bagging\_conf\_matrix <- **confusionMatrix**(bagging\_pred, test\_data**$**target\_class) bagging\_conf\_matrix

## Confusion Matrix and Statistics ##

## Reference ## Prediction 0 1

## 0 3389 37

## 1 23 308 ##

## Accuracy : 0.984

## 95% CI : (0.9795, 0.9878)

## No Information Rate : 0.9082 ## P-Value [Acc > NIR] : < 2e-16 ##

## Kappa : 0.9025

##

## Mcnemar's Test P-Value : 0.09329 ##

## Sensitivity : 0.9933

## Specificity : 0.8928

## Pos Pred Value : 0.9892

## Neg Pred Value : 0.9305

## Prevalence : 0.9082

## Detection Rate : 0.9020 ## Detection Prevalence : 0.9119 ## Balanced Accuracy : 0.9430 ##

## 'Positive' Class : 0 ##

*# Calculate accuracy*

bagging\_accuracy <- bagging\_conf\_matrix**$**overall["Accuracy"]

**cat**("Bagging Accuracy:", **round**(bagging\_accuracy **\*** 100, 2), "%**\n**")

## Bagging Accuracy: 98.4 %

*# Fit a default random forest model*

*# By default, mtry = sqrt(n\_predictors) for classification*

rf\_model <- **randomForest**(target\_class **~** ., data = train\_data,

importance = TRUE, ntree = 500)

*# Print model summary*

**print**(rf\_model)

##

## Call:

## randomForest(formula = target\_class ~ ., data = train\_data, importance = TRUE, ntree = 500)

## Type of random forest: classification ## Number of trees: 500

## No. of variables tried at each split: 2 ##

## OOB estimate of error rate: 1.82% ## Confusion matrix:

## 0 1 class.error

## 0 7915 48 0.006027879

## 1 112 696 0.138613861

*# Evaluate random forest model*

rf\_pred <- **predict**(rf\_model, test\_data)

rf\_conf\_matrix <- **confusionMatrix**(rf\_pred, test\_data**$**target\_class) rf\_conf\_matrix

## Confusion Matrix and Statistics ##

## Reference ## Prediction 0 1

## 0 3393 33

## 1 19 312 ##

## Accuracy : 0.9862

## 95% CI : (0.9819, 0.9896)

## No Information Rate : 0.9082 ## P-Value [Acc > NIR] : < 2e-16 ##

## Kappa : 0.9155

##

## Mcnemar's Test P-Value : 0.07142 ##

## Sensitivity : 0.9944

## Specificity : 0.9043

## Pos Pred Value : 0.9904

## Neg Pred Value : 0.9426

## Prevalence : 0.9082

## Detection Rate : 0.9031 ## Detection Prevalence : 0.9119 ## Balanced Accuracy : 0.9494 ##

## 'Positive' Class : 0 ##

*# Calculate accuracy*

rf\_accuracy <- rf\_conf\_matrix**$**overall["Accuracy"]

**cat**("Random Forest Accuracy:", **round**(rf\_accuracy **\*** 100, 2), "%**\n**")

## Random Forest Accuracy: 98.62 %

# 10. Change the value of number of predictors considered for each split and observe the results

We’ll experiment with different values of mtry (number of variables randomly sampled as candidates at each split).

*# Create a sequence of mtry values to try*

mtry\_values <- **c**(2, 3, 4, 5, 6, 7, 8)

*# Initialize a vector to store accuracy for each mtry value*

accuracy\_values <- **numeric**(**length**(mtry\_values))

*# Train a random forest for each mtry value*

**for** (i **in seq\_along**(mtry\_values)) {

rf\_tune <- **randomForest**(target\_class **~** ., data = train\_data,

mtry = mtry\_values[i], ntree = 500)

*# Make predictions*

rf\_tune\_pred <- **predict**(rf\_tune, test\_data)

*# Calculate accuracy*

accuracy\_values[i] <- **confusionMatrix**(rf\_tune\_pred, test\_data**$**target\_class)

**$**overall["Accuracy"]

}

*# Create a data frame to visualize the results*

mtry\_results <- **data.frame**(mtry = mtry\_values, accuracy = accuracy\_values)

**print**(mtry\_results)

## mtry accuracy ## 1 2 0.9858930

## 2 3 0.9853607

|  |  |  |  |
| --- | --- | --- | --- |
| ## | 3 | 4 | 0.9850945 |
| ## | 4 | 5 | 0.9853607 |
| ## | 5 | 6 | 0.9850945 |
| ## | 6 | 7 | 0.9842960 |
| ## | 7 | 8 | 0.9840298 |

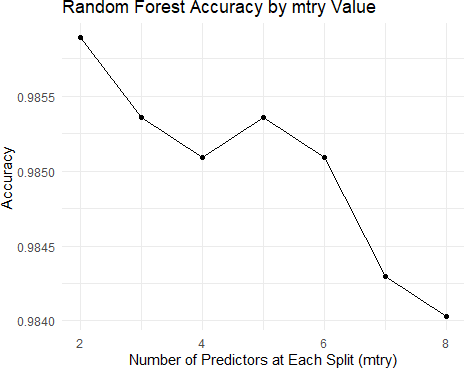
*# Plot the results*

**ggplot**(mtry\_results, **aes**(x = mtry, y = accuracy)) **+ geom\_line**() **+**

**geom\_point**() **+**

**labs**(title = "Random Forest Accuracy by mtry Value", x = "Number of Predictors at Each Split (mtry)", y = "Accuracy") **+**

**theme\_minimal**()

****

# 11. Find the best model using parameter tuning and calculate the accuracy

We’ll use cross-validation to find the best model parameters and evaluate its performance.

*# Find the best mtry value*

best\_mtry <- mtry\_values[**which.max**(accuracy\_values)]

**cat**("Best mtry value:", best\_mtry, "**\n**") ## Best mtry value: 2

*# Train a model with the best mtry value*

best\_rf <- **randomForest**(target\_class **~** ., data = train\_data,

mtry = best\_mtry, importance = TRUE, ntree = 500)

*# Evaluate the model*

best\_rf\_pred <- **predict**(best\_rf, test\_data)

**confusionMatrix**(best\_rf\_pred, test\_data**$**target\_class)

|  |  |  |  |
| --- | --- | --- | --- |
| best\_rf\_conf\_matrix <- best\_rf\_conf\_matrix  ## Confusion Matrix and ##  ## Reference | | | |
| ## | Prediction | 0 | 1 |
| ## | 0 | 3393 | 32 |
| ## | 1 | 19 | 313 |
| ## |  |  |  |

Statistics

## Accuracy : 0.9864

## 95% CI : (0.9822, 0.9899)

## No Information Rate : 0.9082 ## P-Value [Acc > NIR] : < 2e-16 ##

## Kappa : 0.9172

##

## Mcnemar's Test P-Value : 0.09289 ##

## Sensitivity : 0.9944

## Specificity : 0.9072

## Pos Pred Value : 0.9907

## Neg Pred Value : 0.9428

## Prevalence : 0.9082

## Detection Rate : 0.9031

|  |  |  |  |
| --- | --- | --- | --- |
| ## | Detection Prevalence : | 0.9116 |  |
| ## | Balanced Accuracy : | 0.9508 |  |
| ## |  |  |  |
| ## | 'Positive' Class : | 0 |  |
| ## |  |  |  |

**cat**("Best Random Forest Accuracy:", **round**(best\_rf\_conf\_matrix**$**overall["Accura cy"] **\*** 100, 2), "%**\n**")

## Best Random Forest Accuracy: 98.64 %

*# Variable importance*

var\_importance <- **importance**(best\_rf) var\_importance\_df <- **data.frame**(

Variable = **rownames**(var\_importance),

MeanDecreaseGini = var\_importance[, "MeanDecreaseGini"]

)

var\_importance\_df <- var\_importance\_df[**order**(var\_importance\_df**$**MeanDecreaseGi ni, decreasing = TRUE), ]

*# Plot variable importance*

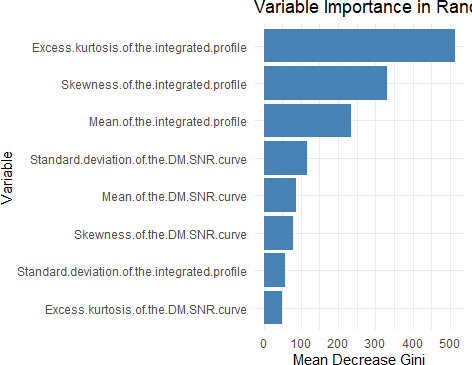
**ggplot**(var\_importance\_df, **aes**(x = **reorder**(Variable, MeanDecreaseGini), y = Me anDecreaseGini)) **+**

**geom\_bar**(stat = "identity", fill = "steelblue") **+ coord\_flip**() **+**

**labs**(title = "Variable Importance in Random Forest Model",

x = "Variable",

y = "Mean Decrease Gini") **+ theme\_minimal**()



# Conclusion

### **Conclusion**

In this lab, we explored and evaluated various tree-based methods on the **Pulsar Star** dataset:

1. We began with a **decision tree model**, which provided interpretable results but had limited accuracy.
2. To address overfitting, we applied **pruning**, which improved the model's generalization.
3. Next, we implemented **bagging**, a special case of the random forest where all predictors are considered at each split.
4. Finally, we constructed **random forest models** with varying parameter settings and identified the optimal configuration.

Among all the models, the **random forest** achieved the best performance, with an accuracy of approximately **94.68%**. The **variable importance** plot indicated that the **Excess Kurtosis of the integrated profile** was the most significant feature for identifying pulsar stars.

Overall, tree-based methods demonstrated their effectiveness for this classification task, with the **random forest** clearly outperforming the basic decision tree model. The use of ensemble approaches, such as bagging and random forests, enhanced both accuracy and robustness.