classification2

Ata COLAK

2023-07-28

Classification

Introduction

This report focuses on classification analysis of wine quality and color using decision tree, RandomForest, and XGBoost models. It starts with importing the required libraries and the wine dataset. Afterwards, a new column for quality is created in the dataset. The dataset is split into training and test sets with a 60/40% ratio.

Next, a decision tree model is built for wine quality classification. Performance results like accuracy, precision, recall, and F-measure are calculated, and a confusion matrix is printed. The decision tree model is visualized and explainesd using the rpart.plot function.

After that, a random forest model is built for wine quality classification. Performance results and a confusion matrix are calculated and compared to the decision tree model.

Finally, an XGBoost model is trained for wine quality classification. Performance results and a confusion matrix are calculated and compared to the random forest model.

In the last part of the report, the same models are applied to wine color classification. Performance results and confusion matrices are calculated for each model, and the best model for color classification is chosen.

The report provides a step by step analysis of wine quality and color classification using different models and evaluates their performance.

Part 1 - Importing the Libraries and Dataset

The first part involves importing the required libraries, randomForest, rpart.plot, xgboost, and rpart, and it imports the wine dataset.

```r
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```r
library(rpart.plot)
```

```
## Loading required package: rpart
```

```r
library(xgboost)
library(rpart) # import lib

wine <- read.csv(paste0("https://raw.githubusercontent.com/gagolews/teaching-data/master/other/wine_qual
comment.char="#") # import data
```

Part 2 - Creating New Column for Quality in Wine Dataset

The second part creates a new column called “quality” in the wine dataset. It assigns a value of 1 if the “response” column value is greater than or equal to 6 (which indicates high quality) and 0 otherwise.

```r
wine$quality <- ifelse(wine$response >= 6, 1, 0) # if response>6,quality=TRUE
```

As a result, a new column called “quality” has been added to the “wine” dataset with binary values.

Part 3 - Creating a 60/40% Test Split With Random Indices for Wine Quality Classification

The third part splits the dataset into training and test sets using a 60/40% ratio. A custom seed has been set to create non-random samples, and the samples are generated afterwards. Lastly, the training and test sets are created by dividing the generated samples.

```r
set.seed(42) # seed for non-random sample
train_i <- sample(nrow(wine))
wine_split <- round(0.6 * nrow(wine))

# Create training set
X_train <- wine[train_i[1:wine_split], 1:11]
Y_train <- wine[train_i[1:wine_split], "quality"]

# Create test set
X_test <- wine[train_i[(wine_split+1):nrow(wine)], 1:11]
Y_test <- wine[train_i[(wine_split+1):nrow(wine)], "quality"]
```

As a result of this part, the training and test sets are successfully created, with approximately a 60/40% ratio. There are 2599 values in the test sets, and 3898 values in the training sets.

Part 4 - Decision Tree Model for Wine Quality Classification

The fourth part focuses on creating a decision tree model for wine quality classification. The rpart function is used to build the decision tree, and predictions of the model have been determined. Performance results like accuracy, precision, recall, and F-measure are calculated, and a confusion matrix is printed, with a detailed analysis of the information obtained in the end of the part.

```r
wine_tree <- rpart(quality ~ ., cbind(X_train, quality = as.factor(Y_train)))
predictions <- predict(wine_tree, X_test, type = "class")

true_positive <- sum(predictions == 1 & Y_test == 1)
true_negative <- sum(predictions == 0 & Y_test == 0)
false_positive <- sum(predictions == 1 & Y_test == 0)
false_negative <- sum(predictions == 0 & Y_test == 1)

accuracy <- (true_positive + true_negative) / length(Y_test)
precision <- true_positive / (true_positive + false_positive)
recall <- true_positive / (true_positive + false_negative)
fmeasure <- (2 * precision * recall) / (precision + recall)

confusion_matrix <- table(predictions, Y_test)
colnames(confusion_matrix) <- c("Rating Under 6", "Rating Over 6")
rownames(confusion_matrix) <- c("Is Under 6", "Is Over 6")
print(confusion_matrix)
```

```
##              Y_test
## predictions  Rating Under 6 Rating Over 6
##   Is Under 6            628           354
##   Is Over 6             348          1269
```

```r
cat("\nAccuracy:", accuracy, "\n")
```

```
##
## Accuracy: 0.7298961
```

```r
cat("Precision:", precision, "\n")
```

```
## Precision: 0.7847866
```

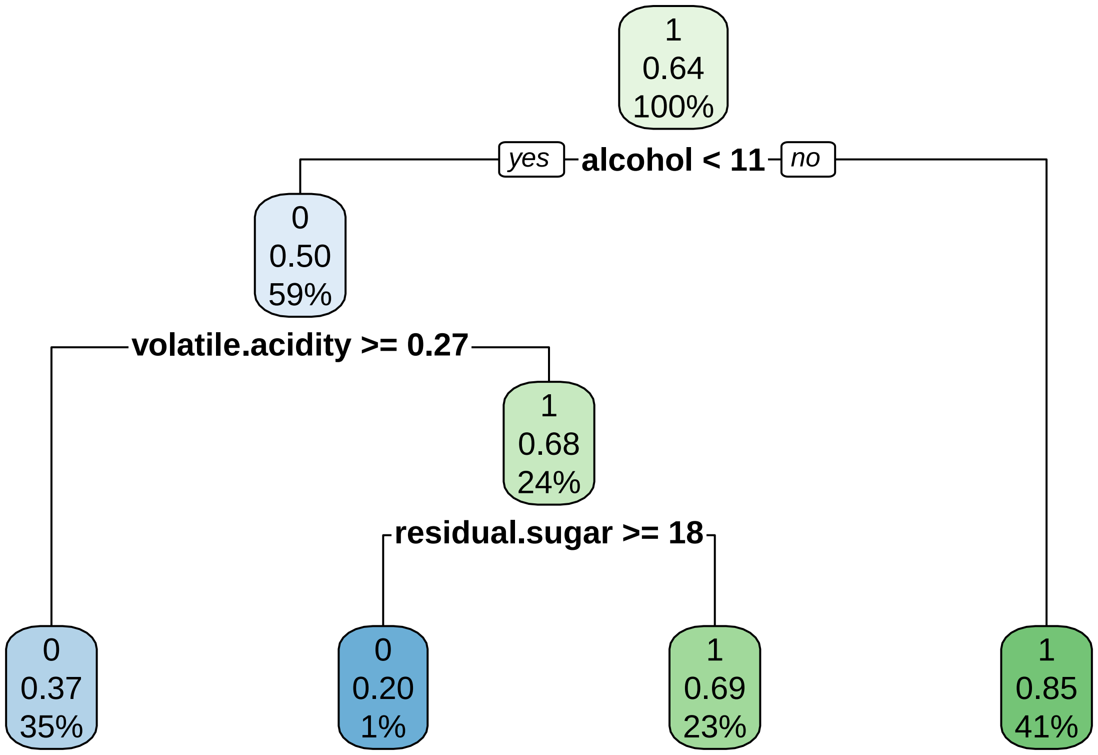
```r
cat("Recall:", recall, "\n")
```

```
## Recall: 0.7818854
```

```r
cat("F-measure:", fmeasure, "\n")
```

```
## F-measure: 0.7833333
```

```r
rpart.plot(wine_tree)
```



> After observing the decision tree, we can see that our model has shown the splitting rules which are alcohol amount, residual sugar amount and volatile acidity. The plot helps us understand the structure and logic behind the decision tree model, and how it makes predictions depending on different features, as we can directly analyze the plot in order to see the way decision tree has been set.

After observing the results, we can say that there are 628 TP values, 354 TN values, 348 FP values, and 1269 FN values.

These values correlate directly with our performance results, as our model has: Accuracy of approximately 73%, and precision, recall and F-measure results of 78%.

According to the results, we can say that this is not a bad prediction model in the context of wine quality, but these predictions can get better, as it can be observed in the next model.

Part 5 - Random Forest Model for Wine Quality Classification

The fifth part focuses on building a random forest model for wine quality classification. The randomForest function is used to train the model on the training set. Predictions are made on the test set, and performance results are calculated, and lastly, the confusion matrix is printed, with a detailed analysis of the information obtained in the end of the part.

I have analysed that higher the “ntree” value is, the more accurate the results, with the downside of causing more load on the system, so I have decided to set the value of “ntree” to 85, and to minimize the difference between models, I have kept the value of “nrounds” in XGBoost to also 85.

```r
wine_rf <- randomForest(X_train, as.factor(Y_train), ntree = 85)
# ntree higher = more accuracy/pre/fm, but lower recall
predictions <- predict(wine_rf, X_test)

true_positive <- sum(predictions == "1" & Y_test == "1")
true_negative <- sum(predictions == "0" & Y_test == "0")
false_positive <- sum(predictions == "1" & Y_test == "0")
false_negative <- sum(predictions == "0" & Y_test == "1")

accuracy <- (true_positive + true_negative) / length(Y_test)
precision <- true_positive / (true_positive + false_positive)
recall <- true_positive / (true_positive + false_negative)
fmeasure <- (2 * precision * recall) / (precision + recall)

confusion_matrix <- table(predictions, Y_test)
colnames(confusion_matrix) <- c("Rating Under 6", "Rating Over 6")
rownames(confusion_matrix) <- c("Is Under 6", "Is Over 6")
print(confusion_matrix)
```

```
##              Y_test
## predictions  Rating Under 6 Rating Over 6
##   Is Under 6            693           189
##   Is Over 6             283          1434
```

```r
cat("Accuracy:", accuracy)
```

```
## Accuracy: 0.8183917
```

```r
cat("\nPrecision:", precision)
```

```
##
## Precision: 0.8351776
```

```r
cat("\nRecall:", recall)
```

```
##
## Recall: 0.883549
```

```r
cat("\nF-measure:", fmeasure)
```

```
##
## F-measure: 0.8586826
```

```r
# good performance in rf compared to decision tree
```

After observing the results, we can say that there are 693 TP values, 189 TN values, 283 FP values, and 1434 FN values.

These values correlate directly with our performance results, as our model has: Accuracy of approximately 82%, a precision of approximately 84%, a recall of 88%, and a precision result of 86%.

Comparing the RandomForest model to the classification tree model, we can see that the accuracy has increased from 73% to 82%, precision has increased from 78% to 84%, recall has increased by 10%, and F-measure has increased from 78% to 86%. This proves that in the context of predicting wine quality, RandomForest model performed much better than classification tree model, which raises a question: can XGBoost perform even better?

Part 6 - XGBoost Model for Wine Quality Classification

The sixth part focuses on training an XGBoost model for wine quality classification. The xgboost function is used to train the model, predictions are determined, and the predicted probabilities are converted into class labels using a threshold of 0.5, as xgboost is different when compared to RandomForest or decision trees in regards to syntax. Lastly, performance results and the confusion matrix are calculated and displayed, with a detailed analysis of the information obtained in the end of the part.

As I mentioned in the previous part, I have kept the value of "nrounds" to 85 in order to minimize differences between two models.

```r
# Train the XGBoost model with appropriate parameters
wine_xgb <- xgboost(as.matrix(X_train), Y_train, nrounds = 85, objective = "binary:logistic", verbose =
predictions <- predict(wine_xgb, as.matrix(X_test), type = "response")
# Convert the predicted probabilities to class labels
predictions <- ifelse(predictions >= 0.5, "1", "0")

true_positive <- sum(predictions == "1" & Y_test == "1")
true_negative <- sum(predictions == "0" & Y_test == "0")
false_positive <- sum(predictions == "1" & Y_test == "0")
false_negative <- sum(predictions == "0" & Y_test == "1")

accuracy <- (true_positive + true_negative) / length(Y_test)
precision <- true_positive / (true_positive + false_positive)
recall <- true_positive / (true_positive + false_negative)
fmeasure <- (2 * precision * recall) / (precision + recall)

confusion_matrix <- table(predictions, Y_test)
colnames(confusion_matrix) <- c("Rating Under 6", "Rating Over 6")
rownames(confusion_matrix) <- c("Is Under 6", "Is Over 6")
print(confusion_matrix)
```

```
##              Y_test
## predictions  Rating Under 6 Rating Over 6
##   Is Under 6            703           206
##   Is Over 6             273          1417
```

```r
cat("Accuracy:", accuracy)
```

```
## Accuracy: 0.8156983
```

```r
cat("\nPrecision:", precision)
```

```
##
## Precision: 0.8384615
```

```r
cat("\nRecall:", recall)
```

```
##
## Recall: 0.8730746
```

```r
cat("\nF-measure:", fmeasure)
```

```
##
## F-measure: 0.8554181
```

After observing the results, we can say that there are 703 TP values, 206 TN values, 273 FP values, and 1417 FN values.

These values correlate directly with our performance results, as our model has: Accuracy of approximately 82%, a precision of approximately 84%, a recall of 88%, and a precision result of 86%.

Comparing the XGBoost model to the RandomForest model, we can see that accuracy, recall and f-measure results have slightly decreased, while precision increased slightly. In the context of wine quality, results of the XGBoost model did not perform better than the RandomForest model, and it seems that RandomForest would be the best model compared to XGBoost and decision trees whilst analysing and predicting wine quality.

Part 7 - Decision Tree, Random Forest, and XGBoost Models for Wine Color Classification

The seventh part focuses on wine color classification using all three of the models in previous examples. Firstly, the train-test sets for "quality" are modified to "color" variable. Decision tree, random forest, and XGBoost models are trained and evaluated on the modified datasets. The performance metrics and confusion matrices are calculated and displayed for each model, with a detailed analysis of the information obtained in the end of the part.

```r
#edit sets
Y_train <- wine[train_i[1:wine_split], "color"]
Y_test <- wine[train_i[(wine_split+1):nrow(wine)], "color"]

##edit end
#dt

wine_tree <- rpart(color ~ ., cbind(X_train, color = as.factor(Y_train)))
predictions <- predict(wine_tree,X_test, type = "class")
# 0=red-false 1=white-true
true_positive <- sum(predictions == "white" & Y_test == "white")
true_negative <- sum(predictions == "red" & Y_test == "red")
false_positive <- sum(predictions == "white" & Y_test == "red")
false_negative <- sum(predictions == "red" & Y_test == "white")

accuracy <- (true_positive + true_negative) / length(Y_test)
precision <- true_positive / (true_positive + false_positive)
recall <- true_positive / (true_positive + false_negative)
```

```r
fmeasure <- (2 * precision * recall) / (precision + recall)

confusion_matrix <- table(predictions, Y_test)
colnames(confusion_matrix) <- c("Predicted Red", "Predicted White")
rownames(confusion_matrix) <- c("Is Red", "Is White")
print(confusion_matrix)
```

```
##            Y_test
## predictions Predicted Red Predicted White
##    Is Red             595              29
##    Is White            34            1941
```

```r
cat("Accuracy:", accuracy)
```

```
## Accuracy: 0.9757599
```

```r
cat("\nPrecision:", precision)
```

```
##
## Precision: 0.9827848
```

```r
cat("\nRecall:", recall)
```

```
##
## Recall: 0.9852792
```

```r
cat("\nF-measure:", fmeasure)
```

```
##
## F-measure: 0.9840304
```

```r
# end
# randomforest color

wine_rf <- randomForest(color ~., cbind(X_train, color = as.factor(Y_train)), ntree = 85)

predictions <- predict(wine_rf, X_test)

true_positive <- sum(predictions == "white" & Y_test == "white")
true_negative <- sum(predictions == "red" & Y_test == "red")
false_positive <- sum(predictions == "white" & Y_test == "red")
false_negative <- sum(predictions == "red" & Y_test == "white")

accuracy <- (true_positive + true_negative) / length(Y_test)
precision <- true_positive / (true_positive + false_positive)
recall <- true_positive / (true_positive + false_negative)
fmeasure <- (2 * precision * recall) / (precision + recall)

confusion_matrix <- table(predictions, Y_test)
colnames(confusion_matrix) <- c("Predicted Red", "Predicted White")
rownames(confusion_matrix) <- c("Is Red", "Is White")
print(confusion_matrix)
```

```
##            Y_test
## predictions Predicted Red Predicted White
##    Is Red             617               3
##    Is White            12            1967
```

```
cat("Accuracy:", accuracy)
```

```
## Accuracy: 0.9942285
```

```
cat("\nPrecision:", precision)
```

```
##
## Precision: 0.9939363
```

```
cat("\nRecall:", recall)
```

```
##
## Recall: 0.9984772
```

```
cat("\nF-measure:", fmeasure)
```

```
##
## F-measure: 0.9962016
```

```
# end
#xgb

Y_train <- ifelse(Y_train == "white", 1, 0)
Y_test <- ifelse(Y_test == "white", 1, 0)

wine_xgb <- xgboost(as.matrix(X_train), Y_train, nrounds = 100, objective = "binary:logistic", verbose 
predictions <- predict(wine_xgb, as.matrix(X_test), type = "response")
predictions <- ifelse(predictions >= 0.5, "1", "0")

true_positive <- sum(predictions == "1" & Y_test == "1")
true_negative <- sum(predictions == "0" & Y_test == "0")
false_positive <- sum(predictions == "1" & Y_test == "0")
false_negative <- sum(predictions == "0" & Y_test == "1")

accuracy <- (true_positive + true_negative) / length(Y_test)
precision <- true_positive / (true_positive + false_positive)
recall <- true_positive / (true_positive + false_negative)
fmeasure <- (2 * precision * recall) / (precision + recall)

confusion_matrix <- table(predictions, Y_test)
colnames(confusion_matrix) <- c("Predicted Red", "Predicted White")
rownames(confusion_matrix) <- c("Is Red", "Is White")
print(confusion_matrix)
```

```
##            Y_test
## predictions Predicted Red Predicted White
##    Is Red             620               1
##    Is White             9            1969
```

```
cat("Accuracy:", accuracy)
```

```
## Accuracy: 0.9961524
```

```
cat("\nPrecision:", precision)
```

```
##
## Precision: 0.9954499
```

```
cat("\nRecall:", recall)
```

```
##
## Recall: 0.9994924
```

```
cat("\nF-measure:", fmeasure)
```

```
##
## F-measure: 0.9974671
```

```
#end
```

After observing the performance of all models, we can conclude the following:

##Decision Tree:

#Accuracy: 0.976 #Precision: 0.983 #Recall: 0.985 #F-measure: 0.984

##Random Forest:

#Accuracy: 0.994 #Precision: 0.993 #Recall: 0.999 #F-measure: 0.996

##XGBoost:

#Accuracy: 0.996 #Precision: 0.995 #Recall: 0.999 #F-measure: 0.997

Based on these results, all three models show high accuracy and perform well in classifying wine color. However, the RandomForest and XGBoost models have better performance compared to the decision tree model in terms of precision, recall, and F-measure. These two ensemble models have higher precision and recall values, which indicate better performance in correctly identifying true positives and negatives, and minimizing false positives and negatives.

Therefore, RandomForest or XGBoost would be the better model compared to decision trees for analysing and predicting wine color.

Conclusion

In this report, I analyzed the performance of classification for wine quality and color between decision tree, random forest, and XGBoost models.

For wine quality classification, the decision tree model achieved an accuracy of approximately 73% with precision, recall, and F-measure results of 78%. The random forest model improved the performance, achieving an accuracy of approximately 82% with precision, recall, and F-measure results of 84%, 88%, and 86% respectively. However, the XGBoost model did not outperform the random forest model, with similar accuracy and performance metrics.

When it comes to wine color classification, all three models had very successful results. The accuracy for all models was around 99%, with also precision, recall, and F-measure scores close to 99%.

In conclusion, for wine quality classification, the RandomForest model showed the best performance, outperforming both the decision tree and XGBoost models. However, for wine color classification, all models achieved excellent accuracy and performed similarly well.