

DATA 300 3 Homework 3 Solution

Aadarsha Gopala Reddy

October 18, 2022

Contents

1. Setup	1
2. Test-Train Split	2
3. Train K-Nearest Neighbors	2
4. Test K-Nearest Neighbors	3
5. Train A Decision Tree	4
6. View the Decision Tree	4
7. Test the Decision Tree	5
8. Compare the Algorithms	6

1. Setup

- Install and Import the kidney disease dataset as an object in R.
- Load the following packages: `tidyverse`, `caret`, `rpart`, and `rpart.plot`.

`install.packages()` Installs packages from CRAN.

`library()` Loads any packages.

`read.csv()` Reads the csv file.

```
library(tidyverse)
```

```
## -- Attaching packages ----- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
```

```
## Attaching package: 'caret'
```

```
##
## The following object is masked from 'package:purrr':
##
## lift

library(rpart)
library(rpart.plot)

kidney <- read.csv("kidney_disease.csv")
```

2. Test-Train Split

Split the dataset into a testing and training dataset.

- Selection into the testing and training datasets should be random.
- Each patient should appear in either the testing or training dataset, but no patient should appear in both.
- Your training dataset should have more observations than your testing dataset. Aim to have between 60 and 70% of the observations in the training dataset.

Hint: you might do this using either the `sample()` function or the `rbinom()` function.

`set.seed()` Sets the seed for the random number generator.

`sample()` Samples a vector of numbers.

`setdiff()` Returns the difference between two vectors.

```
set.seed(1)
train <- sample(1:nrow(kidney), size = 0.7 * nrow(kidney))
test <- setdiff(1:nrow(kidney), train)
```

3. Train K-Nearest Neighbors

Train a K-Nearest Neighbors algorithm to predict nephritis.

- When training the model, use the training dataset only.
- You may need to change how the variables in the dataset are coded to train the algorithm.
- Use the `train()` function in the `caret` package.

`as.factor()` Converts a vector to a factor.

`train()` Trains a model.

```
# change the coding of the variables
kidney$nausea <- as.factor(kidney$nausea)
kidney$backpain <- as.factor(kidney$backpain)
kidney$pushing <- as.factor(kidney$pushing)
kidney$pain <- as.factor(kidney$pain)
kidney$itching <- as.factor(kidney$itching)
kidney$nephritis <- as.factor(kidney$nephritis)

# train the model
knn_model <- train(nephritis ~ ., data = kidney[train, ], method = "knn")
```

```
# print the model
knn_model
```

```
## k-Nearest Neighbors
##
## 84 samples
## 6 predictor
## 2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 84, 84, 84, 84, 84, ...
## Resampling results across tuning parameters:
##
##  k  Accuracy  Kappa
##  5  0.9834091  0.9660500
##  7  0.9677500  0.9343441
##  9  0.9466220  0.8918256
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

Interpret the output of the trained algorithm. What value of K works best for predicting nephritis? What is the predictive accuracy for that value of K?

The optimal value of K is 5 which gives an accuracy of 98.34091%.

4. Test K-Nearest Neighbors

Test the algorithm you trained in the previous section using your testing dataset.

- Generate predictions for the patients in the testing dataset using the `predict()` function.
- You may find the `confusionMatrix()` function helpful for comparing the predictions to the actual outcomes.

`predict()` Generates predictions for a model.

`confusionMatrix()` Generates a confusion matrix.

```
# predict the outcomes
knn_pred <- predict(knn_model, kidney[test, ])

# compare the predictions to the actual outcomes
confusionMatrix(knn_pred, kidney$nephritis[test])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction no yes
##      no  23   0
##      yes  0  13
##
##           Accuracy : 1
##           95% CI : (0.9026, 1)
##      No Information Rate : 0.6389
##      P-Value [Acc > NIR] : 9.893e-08
```

```
##
##           Kappa : 1
##
## Mcnemar's Test P-Value : NA
##
##           Sensitivity : 1.0000
##           Specificity : 1.0000
##           Pos Pred Value : 1.0000
##           Neg Pred Value : 1.0000
##           Prevalence : 0.6389
##           Detection Rate : 0.6389
##           Detection Prevalence : 0.6389
##           Balanced Accuracy : 1.0000
##
##           'Positive' Class : no
##
```

What is the predictive accuracy of the algorithm in the testing data?

The predictive accuracy of the k-nearest neighbors algorithm in the testing data is 100%.

5. Train A Decision Tree

Train a decision tree to predict nephritis.

- When training the model, use the training dataset only.
- Use the `rpart()` function in the `rpart` package.

`rpart()` Trains a decision tree.

```
# train the model
tree_model <- rpart(nephritis ~ ., data = kidney[train, ], method = "class")

# print the model
tree_model

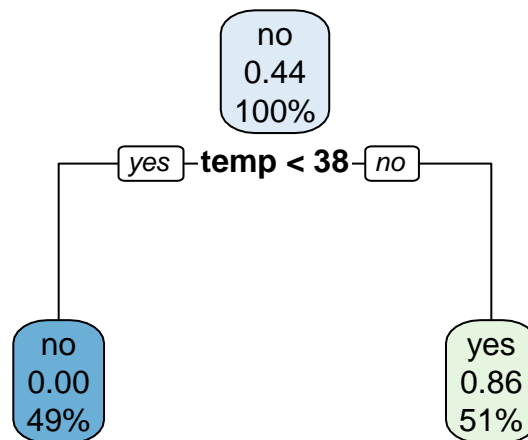
## n= 84
##
## node), split, n, loss, yval, (yprob)
##      * denotes terminal node
##
## 1) root 84 37 no (0.5595238 0.4404762)
##   2) temp< 37.95 41 0 no (1.0000000 0.0000000) *
##   3) temp>=37.95 43 6 yes (0.1395349 0.8604651) *
```

6. View the Decision Tree

Use the `rpart.plot()` function to view the decision tree that you trained in the previous question. Interpret this tree. What symptoms is the algorithm using to predict nephritis?

`rpart.plot()` Plots a decision tree.

```
# plot the tree
rpart.plot(tree_model)
```



7. Test the Decision Tree

Test the decision tree that you trained in question 5 using your testing dataset. Again, you can generate predictions using the `predict()` function.

`predict()` Generates predictions for a model.

```

# predict the outcomes
tree_pred <- predict(tree_model, kidney[test, ], type = "class")

# compare the predictions to the actual outcomes
confusionMatrix(tree_pred, kidney$nephritis[test])

```

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction no yes
##      no  19   0
##      yes   4  13
##
##              Accuracy : 0.8889
##              95% CI : (0.7394, 0.9689)
##      No Information Rate : 0.6389
##      P-Value [Acc > NIR] : 0.0007443
##
##              Kappa : 0.7743

```

```
##
## McNemar's Test P-Value : 0.1336144
##
##          Sensitivity : 0.8261
##          Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.7647
##          Prevalence : 0.6389
##          Detection Rate : 0.5278
##          Detection Prevalence : 0.5278
##          Balanced Accuracy : 0.9130
##
##          'Positive' Class : no
##
```

What is the predictive accuracy of the decision tree in the testing data?

The predictive accuracy of the decision tree in the testing data is 88.89%.

8. Compare the Algorithms

Which of your two algorithms predicts nephritis more accurately?

The k-nearest neighbors algorithm predicts nephritis more accurately than the Decision Tree Algorithm. Although, for most seed values, both algorithms predict with 100% accuracy for this dataset.