DS 8002 Machine Learning Project 1 – Data set classification analysis (October 2016)

Najlis, Bernardo J. (Student Number: 500744793)

Abstract—**In this project, you will apply three algorithms to two data sets. The data and the algorithms are provided by R. Please answer each question in the order they appear. Do not skip to later steps to answer earlier questions that ask you to predict outcomes based on your analysis of the data and understanding of the algorithms. Submit your report in D2L by midnight on the due date.**

I. Introduction

IN this project we perform analysis on two data sets (iris and contact lens) over three different types of classification machine learning algorithms. The analysis was done using R and libraries for each type of algorithm used. As the original assignment was meant to be done using Weka, most of the R code is just a shell interface to Weka classes and libraries.

This report is accompanied by an R markup document with all the code and its output to support all answers and reasoning presented here.

II. DATASETS

The two datasets are different in at least four distinctive ways:

1. **Data format**: Contact Lens is text separated by spaces, Iris is comma separated values
2. **Number of rows**: Contact Lens has only 24 rows, Iris has 150 rows
3. **Attribute data types**: Contact Lens has all numeric integer and discrete values for the attributes, Iris has numeric decimal continuous values for the attributes.
4. **Class attribute**: Contact Lens has a numeric discrete value for class, Iris has a string text for class.

Which algorithm do you expect to perform best on the Contact Lens data data? Why?

I expect the **Multilayer Perceptron** to perform better on the contact lens data, as it tends to perform better over smaller datasets if done with larger structures and high number of epochs.

Which algorithm do you expect to perform best on the Iris data? Why?

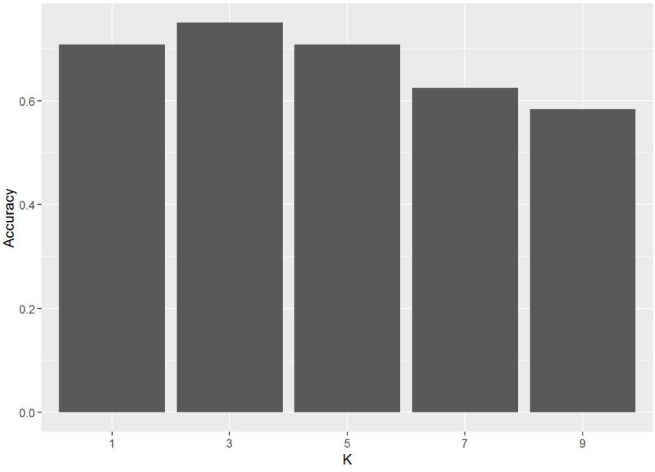
I expect decision trees to perform best on the iris data as it is a larger dataset where the entropy calculations can be applied reasonably, and will not tend to over fit the structure.

III. KNN on the Contact Lens Data

Run KNN on each data set with 1, 3, 5, 7 and 9 neighbors. Report the results for each run in a confusion matrix and comparisons in a table or graph.

Which K gives the best results? Why?

Both K = 1 and K = 3 give the best values, as expected. K = 1 should always give 100% accuracy = 1 and K = 3 also has 100% accuracy as the underlying data has 3 classes.

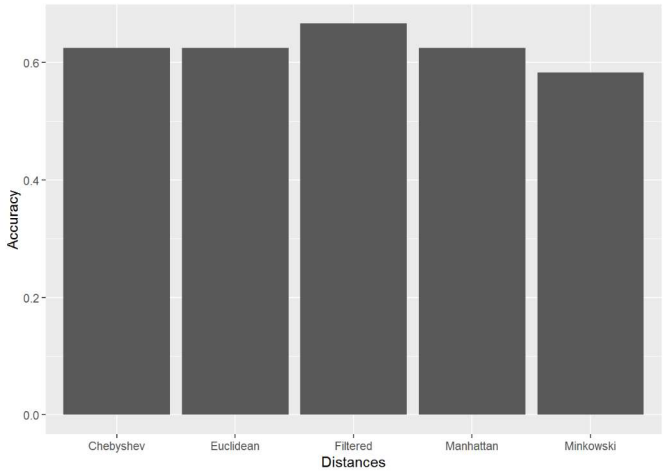


Holding K constant, try different distance functions on each data set.

Which distance function(s) work best for each data set? Why?

Using Weka we set K fixed at 9 (which is the least accurate model using Euclidean distance in the first analysis) and recreated models using Chebyshev, Filtered, Manhattan and Minkowski distance calculations.

Out of all the distance calculations (Chebyshev, Manhattan, Minkowski) the best for the lenses data set is the **Filtered** distance.

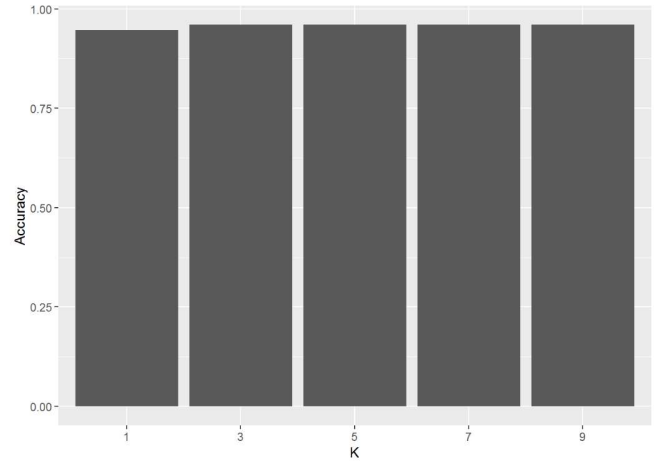


IV. KNN ON THE IRIS DATA

Run KNN on each data set with 1, 3, 5, 7 and 9 neighbors. Report the results for each run in a confusion matrix and comparisons in a table or graph.

Which K gives the best results? Why?

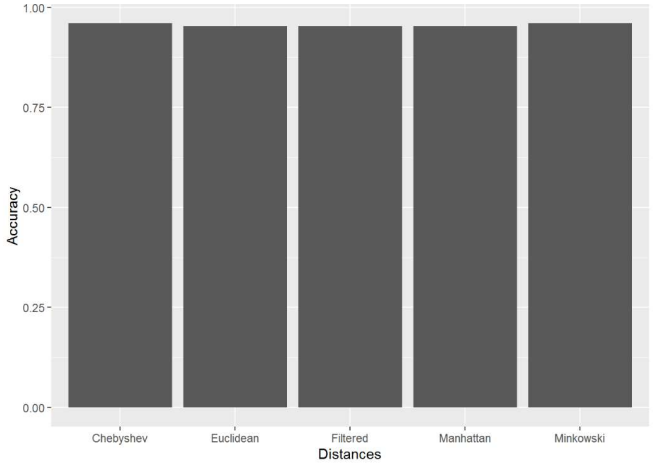
Same as with the contact lens data, the best value is K = 3, as the iris dataset is also arranged in three classes. Accuracy goes down when increasing K over 3. The difference in accuracy is marginal.



Holding K constant, try different distance functions on each data set.

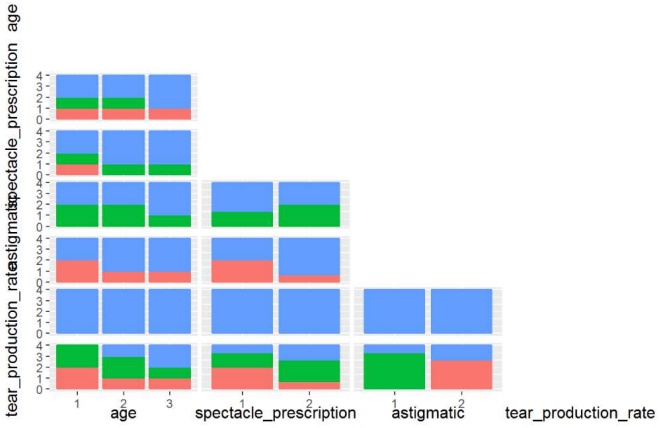
Which distance function(s) work best for each data set? Why?

Same as with the comparison on Euclidean distance with fixed K, the difference in accuracy over different distance calculation methods is marginal.



V. DECISION TREES ON THE CONTACT LENS DATA

Based on R's visualizations, which attribute do you expect to be chosen as the split attribute at the root node?



Based on the R visualization, I expect the age to be the root of the decision tree.

Run each decision tree on the data and report the results for each run in a confusion matrix and comparisons in a table or graph.

How do ID3 and J48 compare in terms of performance?

As the data set is so small, there is no noticeable performance difference between the two methods.

How does pruning affect test performance and generalization performance?

In general, pruning reduces test performance as it removes branches or leaves that do not improve performance over the

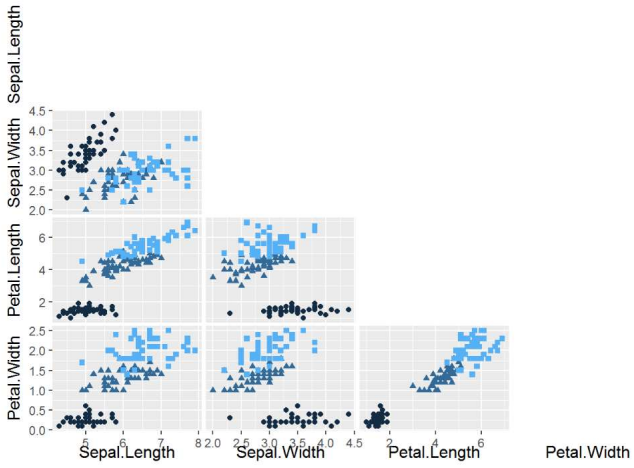
training data.

What does that suggest about overfitting?

This suggests that the tree now is more overfitted in the training data and will not generalize as good as comparable non-pruned tree.

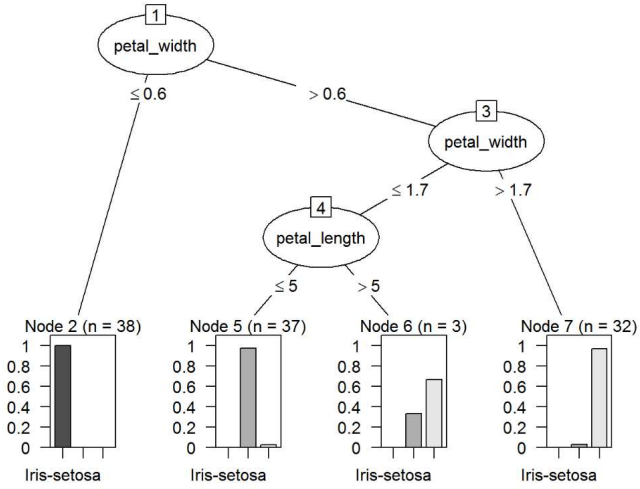
VI. Decision Trees on the Iris Data

Based on R's visualizations, which attribute do you expect to be chosen as the split attribute at the root node?

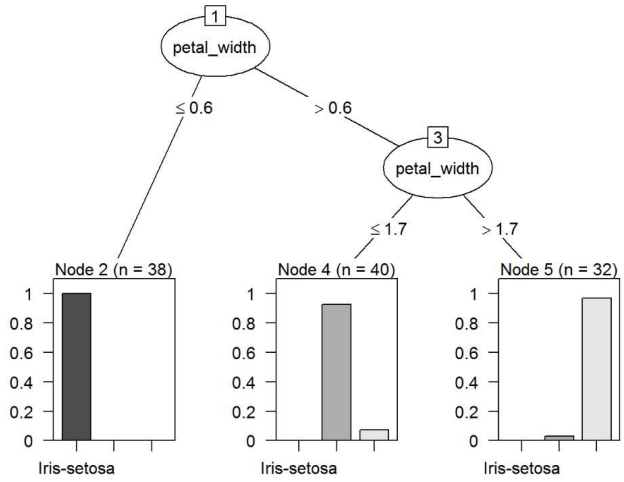


Based on the R visualization, I expect the root of the tree to be Petal.Width.

Run each decision tree on the data and report the results for each run in a confusion matrix and comparisons in a table or graph.



The unpruned version of the tree has 2 levels deep and accuracy of 97.27%.



The pruned version has 1 level deep and accuracy of 96.36%.

The difference in accuracy most likely does not justify the extra computation required if the unpruned version is used.

Why can't you run ID3 on the Iris data?

It doesn't make proper sense to run ID3 on the iris data because the attributes are continuous numeric values and this implementation of the algorithm doesn't account for this types of attributes. In another type of decision trees, these attributes can be handled by taking decision ranges for the continuous variables.

How does pruning affect test performance and generalization performance?

In general, pruning reduces test performance as it removes branches or leaves that do not improve performance over the training data.

What does that suggest about overfitting?

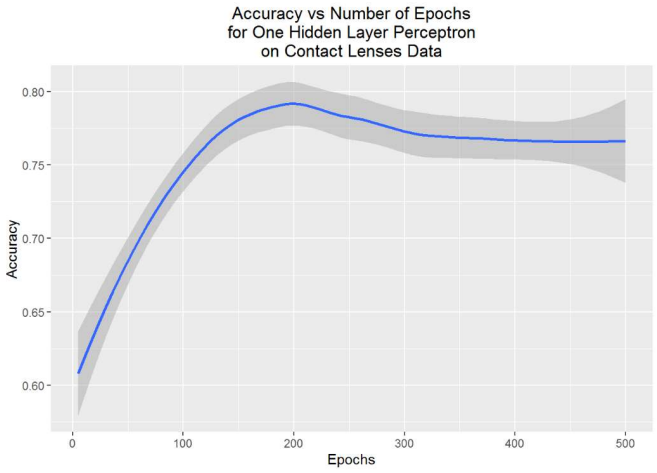
This suggests that the tree now is more overfitted in the training data and will not generalize as good as comparable non-pruned tree.

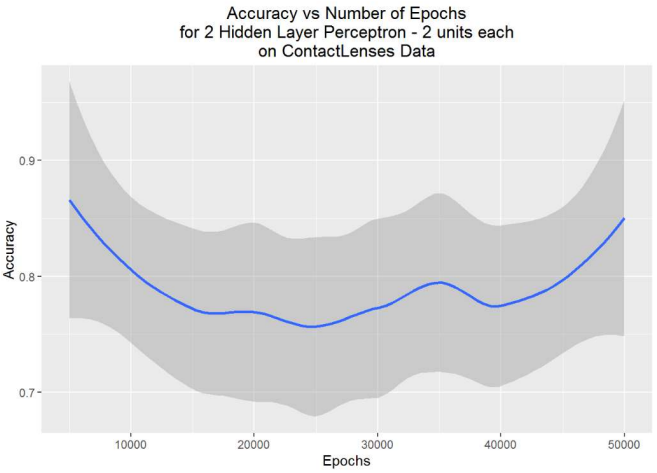
VII. Multilayer Perceptron on the Contact Lens Data

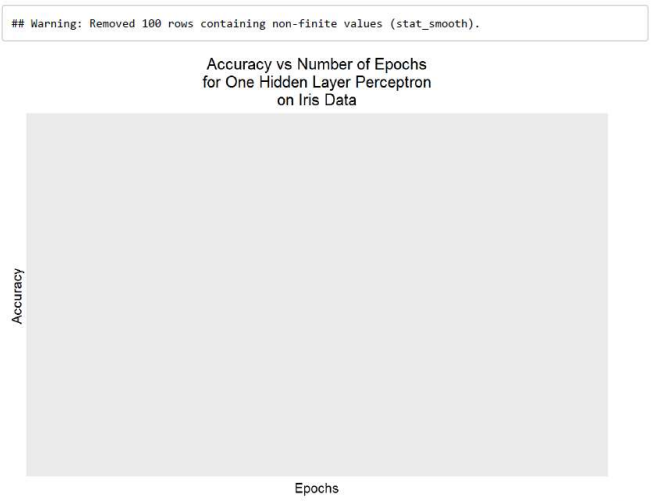
Experiment with different network structures (e.g. extra hidden layers, extra units). Report the results in graphs that show training time (epochs) versus error rate or accuracy.

Which network structures result in the most overfitting?

When comparing One hidden layer vs a two hidden layer – two units multilayer perceptron network, the latter is more overfitting than the former, yielding to the conclusion than the more complex the network the more it will overfit.







VIII. Multilayer Perceptron on the Iris Data

Experiment with different network structures (e.g. extra hidden layers, extra units). Report the results in graphs that show training time (epochs) versus error rate or accuracy.

Which network structures result in the most overfitting?

Note: Due to high number of epochs to run the Iris data set with MLP required to detect any noticeable change in accuracy over epochs, the results throw errors in overflow, and the graphs are shown as empty (see R warning about non-infinite values).

IX. Multilayer Perceptron on the Iris Data

Which algorithm performed best on each data set, for particular definitions of "best?"

For both data sets, the pruned decision tree offers a well-balanced compromise of accuracy and training time required.

For the contact lens data set, as it has very small number of rows, it is very difficult not to occur into overfitting which I tried to avoid by using 10-fold cross validation. The best algorithm for the contact lens data set in terms of accuracy is decision trees.

For the Iris data set as it is fairly sized, in terms of accuracy KNN is the best performing algorithm.

Was the (comparative) performance of the algorithms as you expected? Why?

The performance was fairly much as I expected, the one detail that surprised me was the high number of training time / epochs (and associated training / processing time) required in

order to notice improvements with the multilayer perceptron.

Which data set had the best performance in general across all of the algorithms? Why?

The data set with the best performance is the Iris data set as expected, because it has a much higher number of rows than the contact lens.

DS8002 - Machine Learning Project 1 - Datas set classification analysis (October 2016)

Najlis, Bernardo

October 28, 2016

This is the R code, illustrations and examples that go together with the report for DS8002 - Project 1.

KNN on the Contact Lens Data

1. First install and load libraries used to run KNN.

```
if ( ! any(grepl("RWeka" , installed.packages()))) install.packages("RWeka", dependencies=TRUE)
if ( ! any(grepl("ggplot2" , installed.packages()))) install.packages("ggplot2", dependencies=TR
UE)
library("RWeka")
library("ggplot2")
```

This loads the data file from “lenses.data”. Parameters specify the column names and data types. Also, sample set of the first rows.

```
lenses <- read.table("lenses.data", # name of file reading, this requires setting the working di
rectory to current file and have file in same directory as rmd file
                     header= FALSE, # header is not included in first line
                     col.names =    # to provide names for columns
                       c("id", "age", "spectacle_prescription", "astigmatic", "tear_production_r
ate", "class"), # column names
                     colClasses=    # data types for columns
                       c("NULL",    # as first column is specified as "NULL", read.table will sk
ip this column (row id, which is not to be used)
                          rep("integer", 4), # all other attributes are integer
                           "factor"          # the last column is the class, typified as factor
                          ))
head(lenses)
```

```
##   age spectacle_prescription astigmatic tear_production_rate class
## 1   1                      1          1                    1     3
## 2   1                      1          1                    2     2
## 3   1                      1          2                    1     3
## 4   1                      1          2                    2     1
## 5   1                      2          1                    1     3
## 6   1                      2          1                    2     2
```

Now we set knn to be the desired Weka classifier algorithm (IBk is the class for K nearest neighbors in Weka).

```
weka_knn <- make_Weka_classifier("weka/classifiers/lazy/IBk")
```

These lines generate the KNN models for the different values of K (1, 3, 5, 7 and 9). Each model is saved in a different variable to perform comparisons. The first model for KNN 1 has its parameters commented.

```
lenses_knn1_3fold <-weka_knn(class ~., # take the attribute named 'class' as the class, and all
 others as attributes
                  data=lenses, # training dataset comes from lenses.training
                  control=   # control vector is used to pass parameters to Weka
                    c("-K", 1, # K is number of neighbors in KNN
                      "-W", 0, # windowSize (windowSize -- Gets the maximum number of instances
 allowed in the training pool. The addition of new instances above this value will result in old
 instances being removed. A value of 0 signifies no limit to the number of training instances.)
                      "-A", "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanD
istance -R first-last\"" # nearestNeighborSearchAlgorithm options, includes the distanceFunction
 and the attributeindices (specifies the range of attributes to act on) where "first-last" means
 to use all attributes.
                      ))

lenses_knn3_3fold <- weka_knn(class~., data=lenses, control=c("-K", 3, "-W", 0, "-A", "weka.cor
e.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"" ))

lenses_knn5_3fold <- weka_knn(class~., data=lenses, control=c("-K", 5, "-W", 0, "-A", "weka.cor
e.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"" ))

lenses_knn7_3fold <- weka_knn(class~., data=lenses, control=c("-K", 7, "-W", 0, "-A", "weka.cor
e.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"" ))

lenses_knn9_3fold <- weka_knn(class~., data=lenses, control=c("-K", 9, "-W", 0, "-A", "weka.cor
e.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"" ))
```

This displays the confusion matrix and metrics for KNN.

```
evaluate_Weka_classifier(lenses_knn1_3fold, class=TRUE, numFolds = 3)
```

```
## === 3 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances          20               83.3333 %
## Incorrectly Classified Instances         4               16.6667 %
## Kappa statistic                          0.71
## Mean absolute error                      0.1673
## Root mean squared error                  0.3176
## Relative absolute error                 44.6884 %
## Root relative squared error             74.2756 %
## Total Number of Instances               24
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  0.750    0.100    0.600      0.750    0.667      0.596    0.888     0.681
  1
##                  1.000    0.053    0.833      1.000    0.909      0.889    0.968     0.823
  2
##                  0.800    0.111    0.923      0.800    0.857      0.669    0.844     0.890
  3
## Weighted Avg.    0.833    0.097    0.851      0.833    0.836      0.703    0.877     0.841

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##   3  0  1 |  a = 1
##   0  5  0 |  b = 2
##   2  1 12 |  c = 3
```

```
evaluate_Weka_classifier(lenses_knn3_3fold, class=TRUE, numFolds = 3)
```

```
## === 3 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances          19               79.1667 %
## Incorrectly Classified Instances         5               20.8333 %
## Kappa statistic                          0.6141
## Mean absolute error                      0.2494
## Root mean squared error                  0.3381
## Relative absolute error                 66.6456 %
## Root relative squared error             79.0624 %
## Total Number of Instances               24
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  0.750    0.100    0.600      0.750    0.667      0.596    0.813     0.795
  1
##                  0.600    0.053    0.750      0.600    0.667      0.596    0.932     0.672
  2
##                  0.867    0.222    0.867      0.867    0.867      0.644    0.844     0.896
  3
## Weighted Avg.    0.792    0.167    0.798      0.792    0.792      0.626    0.857     0.832

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##   3  0  1 |  a = 1
##   1  3  1 |  b = 2
##   1  1 13 |  c = 3
```

```
evaluate_Weka_classifier(lenses_knn5_3fold, class=TRUE, numFolds = 3)
```

```
## === 3 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances          17               70.8333 %
## Incorrectly Classified Instances         7               29.1667 %
## Kappa statistic                          0.4783
## Mean absolute error                      0.2707
## Root mean squared error                  0.3378
## Relative absolute error                 72.3355 %
## Root relative squared error             79.0103 %
## Total Number of Instances               24
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  0.750    0.150    0.500      0.750    0.600      0.516    0.888     0.591
  1
##                  0.400    0.105    0.500      0.400    0.444      0.321    0.921     0.639
  2
##                  0.800    0.222    0.857      0.800    0.828      0.567    0.907     0.946
  3
## Weighted Avg.    0.708    0.186    0.723      0.708    0.710      0.508    0.907     0.823

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##   3  1  0 |  a = 1
##   1  2  2 |  b = 2
##   2  1 12 |  c = 3
```

```
evaluate_Weka_classifier(lenses_knn7_3fold, class=TRUE, numFolds = 3)
```

```
## === 3 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances          19               79.1667 %
## Incorrectly Classified Instances         5               20.8333 %
## Kappa statistic                          0.5848
## Mean absolute error                      0.2793
## Root mean squared error                  0.3417
## Relative absolute error                 74.6174 %
## Root relative squared error             79.9178 %
## Total Number of Instances               24
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  0.500    0.050    0.667      0.500    0.571      0.507    0.869     0.685
  1
##                  0.600    0.053    0.750      0.600    0.667      0.596    0.921     0.811
  2
##                  0.933    0.333    0.824      0.933    0.875      0.639    0.937     0.972
  3
## Weighted Avg.    0.792    0.228    0.782      0.792    0.781      0.608    0.922     0.891

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##   2  1  1 |  a = 1
##   0  3  2 |  b = 2
##   1  0 14 |  c = 3
```

```
evaluate_Weka_classifier(lenses_knn9_3fold, class=TRUE, numFolds = 3)
```

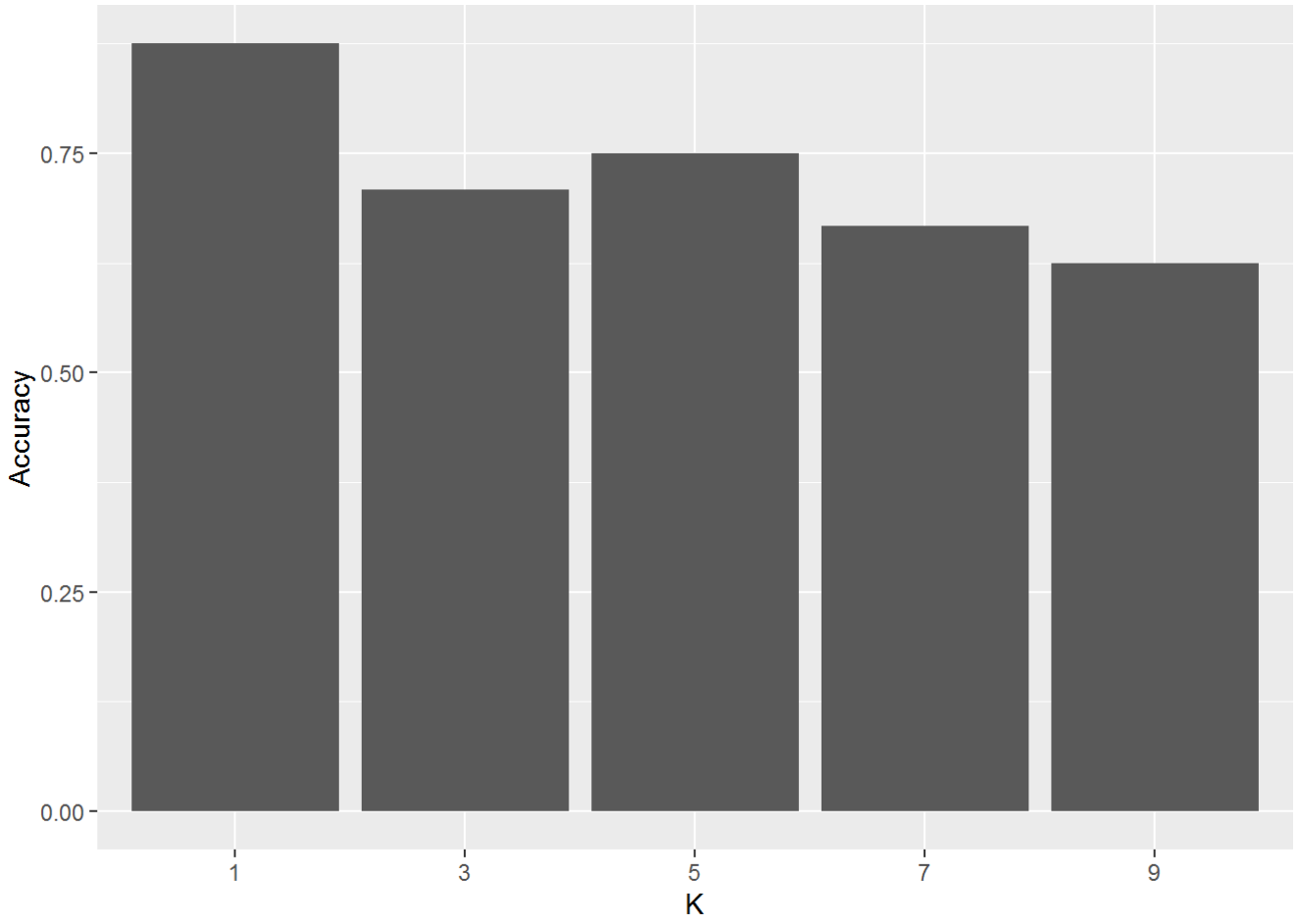
```
## === 3 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances          15               62.5    %
## Incorrectly Classified Instances         9               37.5    %
## Kappa statistic                          0
## Mean absolute error                      0.301
## Root mean squared error                  0.3686
## Relative absolute error                 80.4186 %
## Root relative squared error             86.2142 %
## Total Number of Instances               24
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  0.000    0.000    0.000      0.000    0.000      0.000    0.800     0.381
  1
##                  0.000    0.000    0.000      0.000    0.000      0.000    0.911     0.675
  2
##                  1.000    1.000    0.625      1.000    0.769      0.000    0.893     0.946
  3
## Weighted Avg.    0.625    0.625    0.391      0.625    0.481      0.000    0.881     0.796

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##   0  0  4 |  a = 1
##   0  0  5 |  b = 2
##   0  0 15 |  c = 3
```

Comparison of different K values for Contact Lens Data

```
kvalues = c("1","3","5","7","9")
lenses.accuracies = c(
  evaluate_Weka_classifier(lenses_knn1_3fold, class=TRUE, numFolds = 3)$details['pctCorrect'] / 10
  evaluate_Weka_classifier(lenses_knn3_3fold, class=TRUE, numFolds = 3)$details['pctCorrect'] / 10
  evaluate_Weka_classifier(lenses_knn5_3fold, class=TRUE, numFolds = 3)$details['pctCorrect'] / 10
  evaluate_Weka_classifier(lenses_knn7_3fold, class=TRUE, numFolds = 3)$details['pctCorrect'] / 10
  evaluate_Weka_classifier(lenses_knn9_3fold, class=TRUE, numFolds = 3)$details['pctCorrect'] /
100
)
```

Now we will create a plot, comparing the accuracy for each of the K selected in the different KNN models.

```
lenses.comparison <- data.frame(K=kvalues,Accuracy=lenses.accuracies)
ggplot(lenses.comparison, aes(x=K, y=Accuracy)) + geom_bar(stat="identity")
```



Comparison with fixed K and different distance functions

We will now set K to a fixed value (in this case, K=9 as it was the least accurate model with euclidean distance calculation) and re-evaluate accuracy using different distance functions (Chebyschev, Filtered, Manhattan and Minkowski).

```
# Models creation

lenses.knn9.euclidean <- weka_knn(class~., data=lenses, control=c("-K", 9, "-W", 0, "-A", "weka.
core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"" ))

lenses.knn9.chebyshev <- weka_knn(class~., data=lenses, control=c("-K", 9, "-W", 0, "-A", "weka.
core.neighboursearch.LinearNNSearch -A \"weka.core.ChebyshevDistance -R first-last\"" ))

lenses.knn9.filtered <- weka_knn(class~., data=lenses, control=c("-K", 9, "-W", 0, "-A", "weka.c
ore.neighboursearch.LinearNNSearch -A \"weka.core.FilteredDistance -R first-last\"" ))

lenses.knn9.manhattan <- weka_knn(class~., data=lenses, control=c("-K", 9 , "-W", 0, "-A", "wek
a.core.neighboursearch.LinearNNSearch -A \"weka.core.ManhattanDistance -R first-last\"" ))

lenses.knn9.minkowski <- weka_knn(class~., data=lenses, control=c("-K", 9, "-W", 0, "-A", "weka.
core.neighboursearch.LinearNNSearch -A \"weka.core.MinkowskiDistance -R first-last\"" ))

# Models evaluation and confusion matrix display

evaluate_Weka_classifier(lenses.knn9.euclidean, numFolds = 3, class=TRUE)
```

```
## === 3 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances          15               62.5    %
## Incorrectly Classified Instances         9               37.5    %
## Kappa statistic                          0.0847
## Mean absolute error                      0.3113
## Root mean squared error                  0.3756
## Relative absolute error                 83.1775 %
## Root relative squared error             87.8491 %
## Total Number of Instances               24
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  0.000    0.000    0.000      0.000    0.000      0.000    0.719     0.300
  1
##                  0.200    0.053    0.500      0.200    0.286      0.217    0.905     0.625
  2
##                  0.933    0.889    0.636      0.933    0.757      0.078    0.889     0.953
  3
## Weighted Avg.    0.625    0.567    0.502      0.625    0.532      0.094    0.864     0.776

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##   0  0  4 |  a = 1
##   0  1  4 |  b = 2
##   0  1 14 |  c = 3
```

```
evaluate_Weka_classifier(lenses.knn9.chebyshev, numFolds = 3, class=TRUE)
```

```
## === 3 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances          15               62.5    %
## Incorrectly Classified Instances         9               37.5    %
## Kappa statistic                          0
## Mean absolute error                      0.3621
## Root mean squared error                  0.4269
## Relative absolute error                 96.7423 %
## Root relative squared error             99.8445 %
## Total Number of Instances               24
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  0.000    0.000    0.000      0.000    0.000      0.000    0.400     0.146
  1
##                  0.000    0.000    0.000      0.000    0.000      0.000    0.416     0.192
  2
##                  1.000    1.000    0.625      1.000    0.769      0.000    0.500     0.625
  3
## Weighted Avg.    0.625    0.625    0.391      0.625    0.481      0.000    0.466     0.455

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##   0  0  4 |  a = 1
##   0  0  5 |  b = 2
##   0  0 15 |  c = 3
```

```
evaluate_Weka_classifier(lenses.knn9.filtered, numFolds = 3, class=TRUE)
```

```
## === 3 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances          15               62.5    %
## Incorrectly Classified Instances         9               37.5    %
## Kappa statistic                          0.0442
## Mean absolute error                      0.3141
## Root mean squared error                  0.3779
## Relative absolute error                 83.9294 %
## Root relative squared error             88.3851 %
## Total Number of Instances               24
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  0.000    0.000    0.000      0.000    0.000      0.000    0.656     0.235
  1
##                  0.000    0.053    0.000      0.000    0.000      -0.107   0.784     0.484
  2
##                  1.000    0.889    0.652      1.000    0.789      0.269    0.956     0.967
  3
## Weighted Avg.    0.625    0.567    0.408      0.625    0.493      0.146    0.870     0.744

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##   0  1  3 |  a = 1
##   0  0  5 |  b = 2
##   0  0 15 |  c = 3
```

```
evaluate_Weka_classifier(lenses.knn9.manhattan, numFolds = 3, class=TRUE)
```

```
## === 3 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances          16               66.6667 %
## Incorrectly Classified Instances         8               33.3333 %
## Kappa statistic                          0.1504
## Mean absolute error                      0.3059
## Root mean squared error                  0.3758
## Relative absolute error                 81.7232 %
## Root relative squared error             87.8884 %
## Total Number of Instances               24
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  0.000    0.000    0.000      0.000    0.000      0.000    0.856     0.406
  1
##                  0.200    0.000    1.000      0.200    0.333      0.406    0.832     0.552
  2
##                  1.000    0.889    0.652      1.000    0.789      0.269    0.844     0.930
  3
## Weighted Avg.    0.667    0.556    0.616      0.667    0.563      0.253    0.844     0.764

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##   0  0  4 |  a = 1
##   0  1  4 |  b = 2
##   0  0 15 |  c = 3
```

```
evaluate_Weka_classifier(lenses.knn9.minkowski, numFolds = 3, class=TRUE)
```

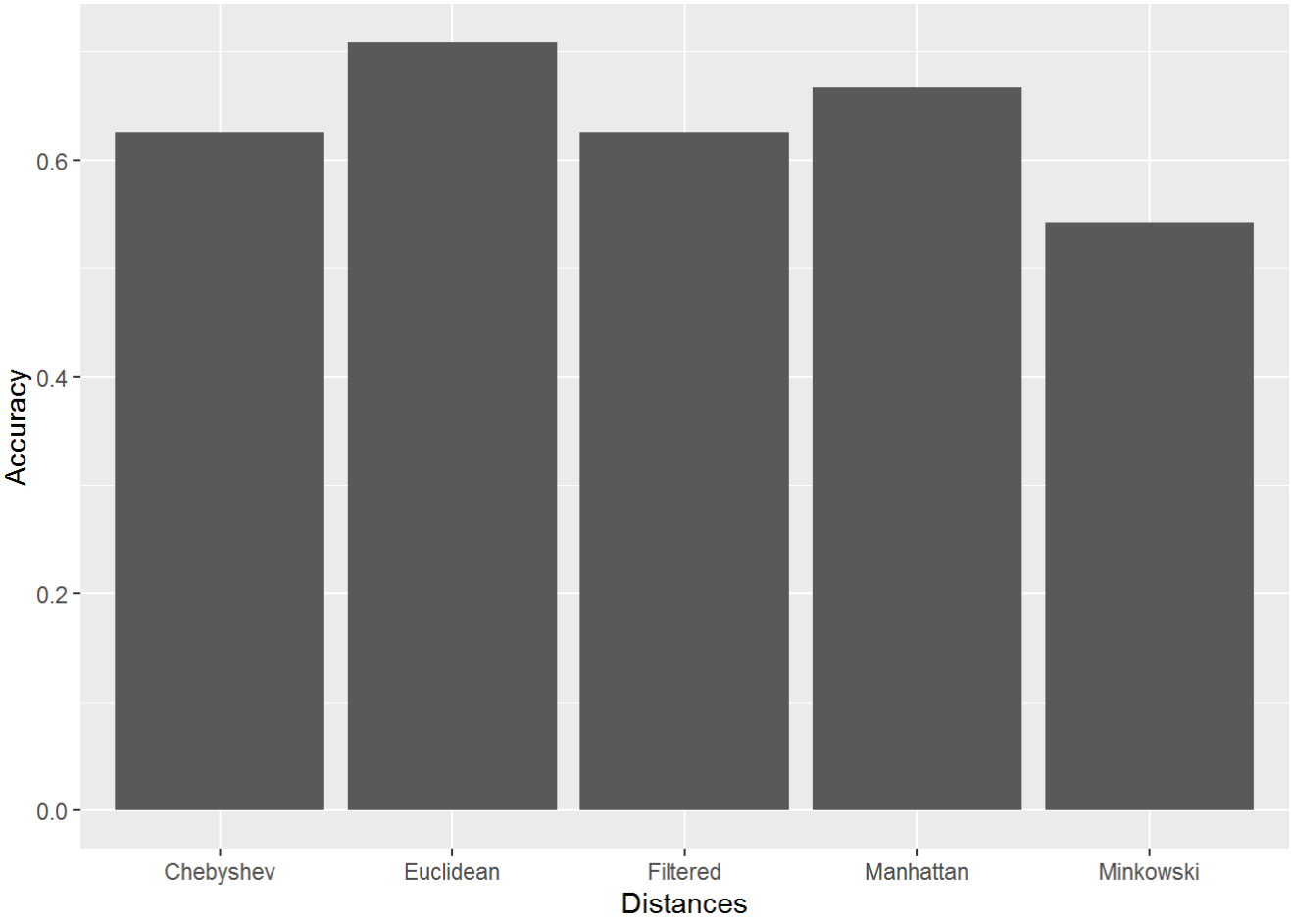
```
## === 3 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances          16               66.6667 %
## Incorrectly Classified Instances         8               33.3333 %
## Kappa statistic                          0.1504
## Mean absolute error                      0.3084
## Root mean squared error                  0.3731
## Relative absolute error                 82.3998 %
## Root relative squared error             87.2528 %
## Total Number of Instances               24
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  0.000    0.000    0.000      0.000    0.000      0.000    0.781     0.379
  1
##                  0.200    0.000    1.000      0.200    0.333      0.406    0.989     0.943
  2
##                  1.000    0.889    0.652      1.000    0.789      0.269    0.863     0.929
  3
## Weighted Avg.    0.667    0.556    0.616      0.667    0.563      0.253    0.876     0.840

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##   0  0  4 |  a = 1
##   0  1  4 |  b = 2
##   0  0 15 |  c = 3
```

Comparison for distance models with Contact Lens Data

```
distancevalues = c("Euclidean", "Chebyshev", "Filtered", "Manhattan", "Minkowski")
lenses.accuracies.dist = c(
  evaluate_Weka_classifier(lenses.knn9.euclidean, numFolds = 3,
class=TRUE)$details['pctCorrect'] / 100,
  evaluate_Weka_classifier(lenses.knn9.chebyshev, numFolds = 3,
class=TRUE)$details['pctCorrect'] / 100,
  evaluate_Weka_classifier(lenses.knn9.filtered, numFolds = 3, class=TRUE)$details['pctCorrect']
 / 100,
  evaluate_Weka_classifier(lenses.knn9.manhattan, numFolds = 3,
class=TRUE)$details['pctCorrect'] / 100,
  evaluate_Weka_classifier(lenses.knn9.minkowski, numFolds = 3,
class=TRUE)$details['pctCorrect'] / 100
)
```

Now we will create a plot, comparing the accuracy for each of the different distance calculations selected in the KNN models.

```
lenses.comparison.dist <- data.frame(Distances=distancevalues,Accuracy=lenses.accuracies.dist)
ggplot(lenses.comparison.dist, aes(x=Distances, y=Accuracy)) + geom_bar(stat="identity")
```



KNN on the Iris data set

This loads the data file from “iris.data”. Parameters specify the column names and data types. Also, sample set of the first rows.

```
iris <- read.csv("iris.data", # name of file reading, this requires setting the working director
y to current file and have file in same directory as rmd file
                 header=FALSE, # header is not included in first line
                   col.names= # to provide names for columns
                   c("sepal_length", "sepal_width", "petal_length", "petal_width", "class"), # c
olumn names
                 colClasses= # data types for columns
                   c(rep("double", 4), #all attributes are double
                     "factor") # with exception of label/class which is factor
                 )

head(iris)
```

```
##   sepal_length sepal_width petal_length petal_width       class
## 1          5.1         3.5          1.4         0.2 Iris-setosa
## 2          4.9         3.0          1.4         0.2 Iris-setosa
## 3          4.7         3.2          1.3         0.2 Iris-setosa
## 4          4.6         3.1          1.5         0.2 Iris-setosa
## 5          5.0         3.6          1.4         0.2 Iris-setosa
## 6          5.4         3.9          1.7         0.4 Iris-setosa
```

Now we set knn to be the desired Weka classifier algorithm (IBk is the class for K nearest neighbors in Weka).

```
weka_knn <- make_Weka_classifier("weka/classifiers/lazy/IBk")
```

These generate the KNN models for the different values of K (1, 3, 5, 7 and 9). Each model is saved in a different variable to perform comparisons. The first model for KNN 1 has its parameters commented.

```
iris_knn1_10fold <-weka_knn(class ~., # take the attribute named 'class' as the class, and all o
thers as attributes
                  data=iris, # training dataset comes from iris.training
                  control=   # control vector is used to pass parameters to Weka
                    c("-K", 1, # K is number of neighbors in KNN
                      "-W", 0, # windowSize (windowSize -- Gets the maximum number of instances
 allowed in the training pool. The addition of new instances above this value will result in old
 instances being removed. A value of 0 signifies no limit to the number of training instances.)
                       "-A", "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanD
istance -R first-last\"" # nearestNeighborSearchAlgorithm options, includes the distanceFunction
 and the attributeindices (specifies the range of attributes to act on) where "first-last" means
 to use all attributes.
                       ))

iris_knn3_10fold <- weka_knn(class~., data=iris, control=c("-K", 3, "-W", 0, "-A", "weka.core.ne
ighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"" ))

iris_knn5_10fold <- weka_knn(class~., data=iris, control=c("-K", 5, "-W", 0, "-A", "weka.core.ne
ighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"" ))

iris_knn7_10fold <- weka_knn(class~., data=iris, control=c("-K", 7, "-W", 0, "-A", "weka.core.ne
ighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"" ))

iris_knn9_10fold <- weka_knn(class~., data=iris, control=c("-K", 9, "-W", 0, "-A", "weka.core.ne
ighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"" ))
```

Evaluation of KNN on Iris Data

This displays the confusion matrix and metrics for KNN.

```
evaluate_Weka_classifier(iris_knn1_10fold, class=TRUE, numFolds = 10)
```

```
## === 10 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances         143               95.3333 %
## Incorrectly Classified Instances         7                4.6667 %
## Kappa statistic                          0.93
## Mean absolute error                      0.0399
## Root mean squared error                  0.1747
## Relative absolute error                  8.9763 %
## Root relative squared error             37.0695 %
## Total Number of Instances              150
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000
  Iris-setosa
##                  0.940    0.040    0.922      0.940    0.931      0.896    0.952     0.887
  Iris-versicolor
##                  0.920    0.030    0.939      0.920    0.929      0.895    0.947     0.894
  Iris-virginica
## Weighted Avg.    0.953    0.023    0.953      0.953    0.953      0.930    0.966     0.927

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##  50  0  0 |  a = Iris-setosa
##   0 47  3 |  b = Iris-versicolor
##   0  4 46 |  c = Iris-virginica
```

```
evaluate_Weka_classifier(iris_knn3_10fold, class=TRUE, numFolds = 10)
```

```
## === 10 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances         143               95.3333 %
## Incorrectly Classified Instances         7                4.6667 %
## Kappa statistic                          0.93
## Mean absolute error                      0.0385
## Root mean squared error                  0.1629
## Relative absolute error                  8.6696 %
## Root relative squared error             34.5634 %
## Total Number of Instances              150
##
## === Detailed Accuracy By Class ===
##
##                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000
  Iris-setosa
##                 0.940    0.040    0.922      0.940    0.931      0.896    0.968     0.927
  Iris-versicolor
##                 0.920    0.030    0.939      0.920    0.929      0.895    0.968     0.928
  Iris-virginica
## Weighted Avg.    0.953    0.023    0.953      0.953    0.953      0.930    0.979     0.952

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##  50  0  0 |  a = Iris-setosa
##   0 47  3 |  b = Iris-versicolor
##   0  4 46 |  c = Iris-virginica
```

```
evaluate_Weka_classifier(iris_knn5_10fold, class=TRUE, numFolds = 10)
```

```
## === 10 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances         143               95.3333 %
## Incorrectly Classified Instances         7                4.6667 %
## Kappa statistic                          0.93
## Mean absolute error                      0.0399
## Root mean squared error                  0.144
## Relative absolute error                  8.9695 %
## Root relative squared error             30.5457 %
## Total Number of Instances              150
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000
  Iris-setosa
##                  0.940    0.040    0.922      0.940    0.931      0.896    0.994     0.984
  Iris-versicolor
##                  0.920    0.030    0.939      0.920    0.929      0.895    0.994     0.984
  Iris-virginica
## Weighted Avg.    0.953    0.023    0.953      0.953    0.953      0.930    0.996     0.989

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##  50  0  0 |  a = Iris-setosa
##   0 47  3 |  b = Iris-versicolor
##   0  4 46 |  c = Iris-virginica
```

```
evaluate_Weka_classifier(iris_knn7_10fold, class=TRUE, numFolds = 10)
```

```
## === 10 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances         145               96.6667 %
## Incorrectly Classified Instances         5                3.3333 %
## Kappa statistic                          0.95
## Mean absolute error                      0.0391
## Root mean squared error                  0.1315
## Relative absolute error                  8.788  %
## Root relative squared error             27.9008 %
## Total Number of Instances              150
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000
  Iris-setosa
##                  0.980    0.040    0.925      0.980    0.951      0.927    0.995     0.988
  Iris-versicolor
##                  0.920    0.010    0.979      0.920    0.948      0.925    0.995     0.987
  Iris-virginica
## Weighted Avg.    0.967    0.017    0.968      0.967    0.967      0.951    0.997     0.992

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##  50  0  0 |  a = Iris-setosa
##   0 49  1 |  b = Iris-versicolor
##   0  4 46 |  c = Iris-virginica
```

```
evaluate_Weka_classifier(iris_knn9_10fold, class=TRUE, numFolds = 10)
```

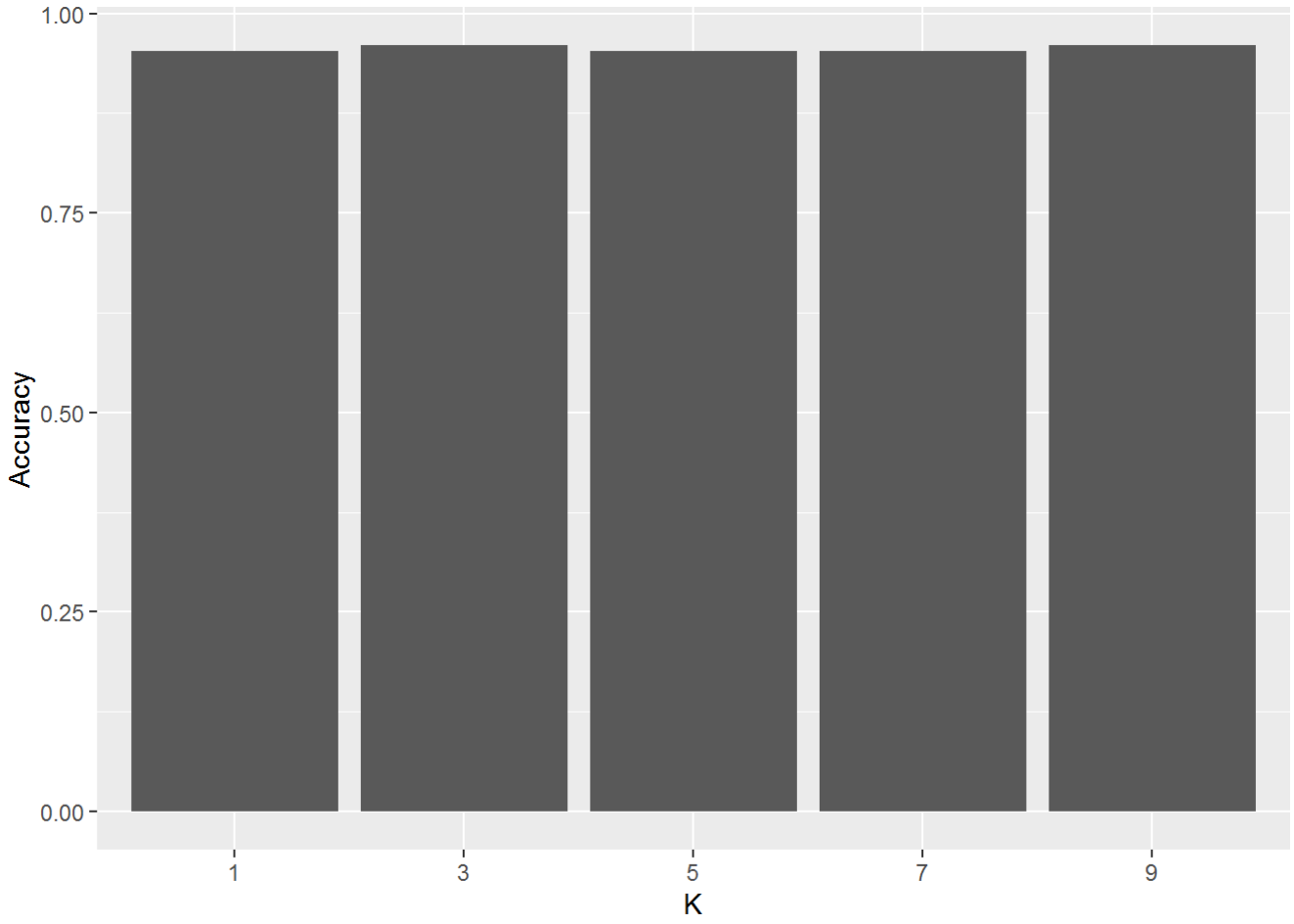
```
## === 10 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances         143               95.3333 %
## Incorrectly Classified Instances         7                4.6667 %
## Kappa statistic                          0.93
## Mean absolute error                      0.0392
## Root mean squared error                  0.129
## Relative absolute error                  8.824  %
## Root relative squared error             27.3706 %
## Total Number of Instances              150
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000
  Iris-setosa
##                  0.940    0.040    0.922      0.940    0.931      0.896    0.996     0.991
  Iris-versicolor
##                  0.920    0.030    0.939      0.920    0.929      0.895    0.996     0.991
  Iris-virginica
## Weighted Avg.    0.953    0.023    0.953      0.953    0.953      0.930    0.997     0.994

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##  50  0  0 |  a = Iris-setosa
##   0 47  3 |  b = Iris-versicolor
##   0  4 46 |  c = Iris-virginica
```

Comparison of different K values for Iris Data

```
kvalues = c("1","3","5","7","9")
iris.accuracies = c(
  evaluate_Weka_classifier(iris_knn1_10fold, class=TRUE, numFolds = 10)$details['pctCorrect'] / 10
  evaluate_Weka_classifier(iris_knn3_10fold, class=TRUE, numFolds = 10)$details['pctCorrect'] / 10
  evaluate_Weka_classifier(iris_knn5_10fold, class=TRUE, numFolds = 10)$details['pctCorrect'] / 10
  evaluate_Weka_classifier(iris_knn7_10fold, class=TRUE, numFolds = 10)$details['pctCorrect'] / 10
  evaluate_Weka_classifier(iris_knn9_10fold, class=TRUE, numFolds = 10)$details['pctCorrect'] /
100
)
```

Now we will create a plot, comparing the accuracy for each of the K selected in the different KNN models.

```
iris.comparison <- data.frame(K=kvalues,Accuracy=iris.accuracies)
ggplot(iris.comparison, aes(x=K, y=Accuracy)) + geom_bar(stat="identity")
```



Comparison with fixed K and different distance functions

We will now set K to a fixed value (in this case, K=9 as it was the least accurate model with euclidean distance calculation) and re-evaluate accuracy using different distance functions (Chebyshev, Filtered, Manhattan and Minkowski).

```
# Models creation

iris.knn9.euclidean <- weka_knn(class~., data=iris, control=c("-K", 9, "-W", 0, "-A", "weka.cor
e.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"" ))

iris.knn9.chebyshev <- weka_knn(class~., data=iris, control=c("-K", 9, "-W", 0, "-A", "weka.cor
e.neighboursearch.LinearNNSearch -A \"weka.core.ChebyshevDistance -R first-last\"" ))

iris.knn9.filtered <- weka_knn(class~., data=iris, control=c("-K", 9, "-W", 0, "-A", "weka.core.
neighboursearch.LinearNNSearch -A \"weka.core.FilteredDistance -R first-last\"" ))

iris.knn9.manhattan <- weka_knn(class~., data=iris, control=c("-K", 9 , "-W", 0, "-A", "weka.cor
e.neighboursearch.LinearNNSearch -A \"weka.core.ManhattanDistance -R first-last\"" ))

iris.knn9.minkowski <- weka_knn(class~., data=iris, control=c("-K", 9, "-W", 0, "-A", "weka.cor
e.neighboursearch.LinearNNSearch -A \"weka.core.MinkowskiDistance -R first-last\"" ))

# Models evaluation and confusion matrix display

evaluate_Weka_classifier(iris.knn9.euclidean, class=TRUE, numFolds = 10)
```

```
## === 10 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances         145               96.6667 %
## Incorrectly Classified Instances         5                3.3333 %
## Kappa statistic                          0.95
## Mean absolute error                      0.0415
## Root mean squared error                  0.1321
## Relative absolute error                  9.3342 %
## Root relative squared error             28.0332 %
## Total Number of Instances              150
##
## === Detailed Accuracy By Class ===
##
##                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000
  Iris-setosa
##                 0.960    0.030    0.941      0.960    0.950      0.925    0.996     0.991
  Iris-versicolor
##                 0.940    0.020    0.959      0.940    0.949      0.925    0.996     0.991
  Iris-virginica
## Weighted Avg.   0.967    0.017    0.967      0.967    0.967      0.950    0.997     0.994

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##  50  0  0 |  a = Iris-setosa
##   0 48  2 |  b = Iris-versicolor
##   0  3 47 |  c = Iris-virginica
```

```
evaluate_Weka_classifier(iris.knn9.chebyshev, class=TRUE, numFolds = 10)
```

```
## === 10 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances         145               96.6667 %
## Incorrectly Classified Instances         5                3.3333 %
## Kappa statistic                          0.95
## Mean absolute error                      0.0491
## Root mean squared error                  0.1427
## Relative absolute error                 11.0487 %
## Root relative squared error             30.2676 %
## Total Number of Instances              150
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000
  Iris-setosa
##                  0.980    0.040    0.925      0.980    0.951      0.927    0.993     0.982
  Iris-versicolor
##                  0.920    0.010    0.979      0.920    0.948      0.925    0.993     0.987
  Iris-virginica
## Weighted Avg.    0.967    0.017    0.968      0.967    0.967      0.951    0.995     0.990

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##  50  0  0 |  a = Iris-setosa
##   0 49  1 |  b = Iris-versicolor
##   0  4 46 |  c = Iris-virginica
```

```
evaluate_Weka_classifier(iris.knn9.filtered, class=TRUE, numFolds = 10)
```

```
## === 10 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances         143               95.3333 %
## Incorrectly Classified Instances         7                4.6667 %
## Kappa statistic                          0.93
## Mean absolute error                      0.0456
## Root mean squared error                  0.1433
## Relative absolute error                 10.266  %
## Root relative squared error             30.4024 %
## Total Number of Instances              150
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000
  Iris-setosa
##                  0.940    0.040    0.922      0.940    0.931      0.896    0.995     0.987
  Iris-versicolor
##                  0.920    0.030    0.939      0.920    0.929      0.895    0.995     0.988
  Iris-virginica
## Weighted Avg.    0.953    0.023    0.953      0.953    0.953      0.930    0.996     0.992

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##  50  0  0 |  a = Iris-setosa
##   0 47  3 |  b = Iris-versicolor
##   0  4 46 |  c = Iris-virginica
```

```
evaluate_Weka_classifier(iris.knn9.manhattan, class=TRUE, numFolds = 10)
```

```
## === 10 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances         142               94.6667 %
## Incorrectly Classified Instances         8                5.3333 %
## Kappa statistic                          0.92
## Mean absolute error                      0.0449
## Root mean squared error                  0.1404
## Relative absolute error                 10.1099 %
## Root relative squared error             29.7822 %
## Total Number of Instances              150
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000
  Iris-setosa
##                  0.940    0.050    0.904      0.940    0.922      0.882    0.995     0.988
  Iris-versicolor
##                  0.900    0.030    0.938      0.900    0.918      0.879    0.995     0.988
  Iris-virginica
## Weighted Avg.    0.947    0.027    0.947      0.947    0.947      0.920    0.997     0.992

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##  50  0  0 |  a = Iris-setosa
##   0 47  3 |  b = Iris-versicolor
##   0  5 45 |  c = Iris-virginica
```

```
evaluate_Weka_classifier(iris.knn9.minkowski, class=TRUE, numFolds = 10)
```

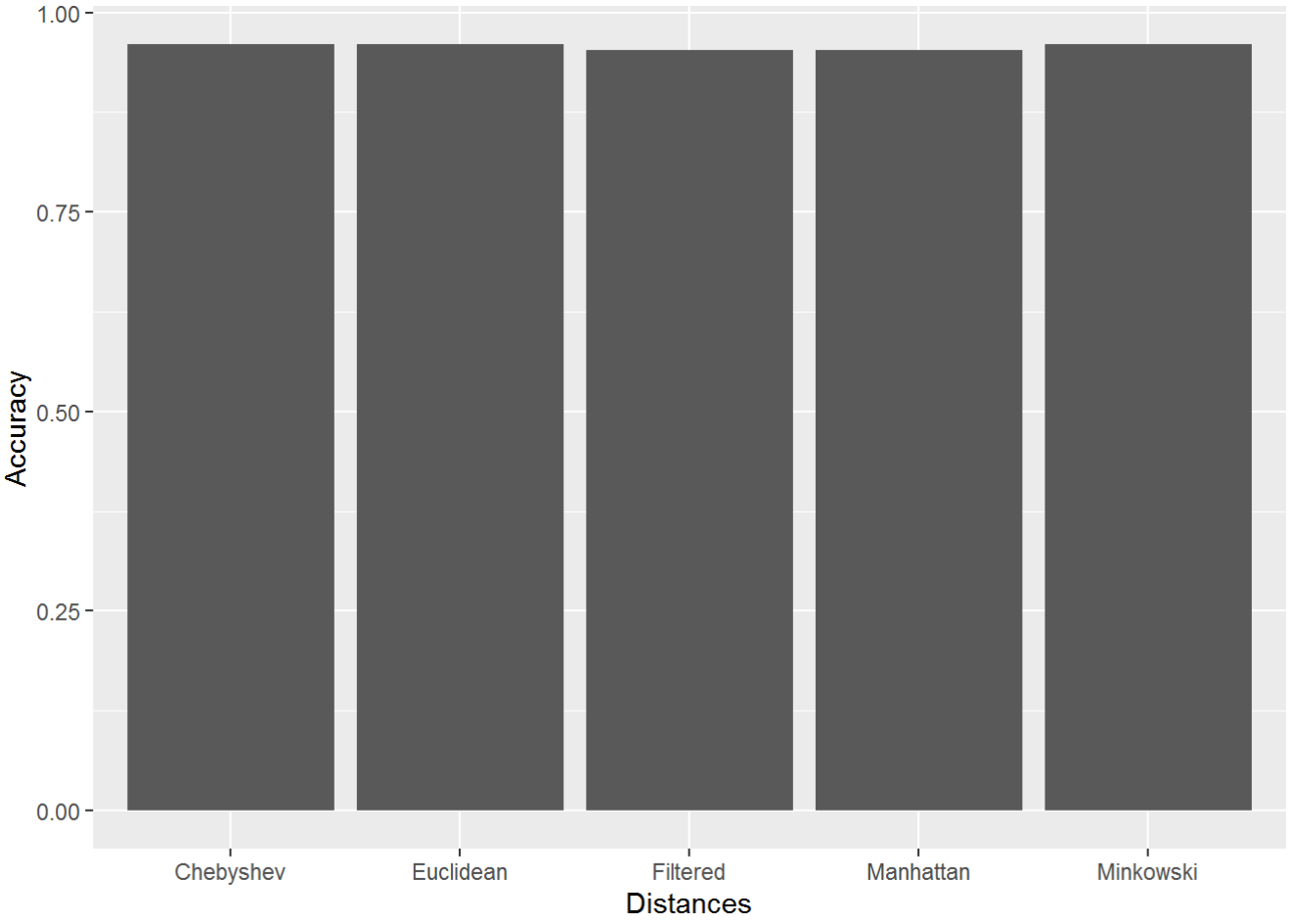
```
## === 10 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances         143               95.3333 %
## Incorrectly Classified Instances         7                4.6667 %
## Kappa statistic                          0.93
## Mean absolute error                      0.0402
## Root mean squared error                  0.1278
## Relative absolute error                  9.0454 %
## Root relative squared error             27.102  %
## Total Number of Instances              150
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000
  Iris-setosa
##                  0.940    0.040    0.922      0.940    0.931      0.896    0.996     0.992
  Iris-versicolor
##                  0.920    0.030    0.939      0.920    0.929      0.895    0.996     0.992
  Iris-virginica
## Weighted Avg.    0.953    0.023    0.953      0.953    0.953      0.930    0.997     0.994

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##  50  0  0 |  a = Iris-setosa
##   0 47  3 |  b = Iris-versicolor
##   0  4 46 |  c = Iris-virginica
```

Comparison for distance models with Iris Data

```
distancevalues = c("Euclidean", "Chebyshev", "Filtered", "Manhattan", "Minkowski")
iris.accuracies.dist = c(
  evaluate_Weka_classifier(iris.knn9.euclidean, class=TRUE, numFolds = 10)$details['pctCorrect']
 / 100,
  evaluate_Weka_classifier(iris.knn9.chebyshev, class=TRUE, numFolds = 10)$details['pctCorrect']
 / 100,
  evaluate_Weka_classifier(iris.knn9.filtered, class=TRUE, numFolds = 10)$details['pctCorrect']
/ 100,
  evaluate_Weka_classifier(iris.knn9.manhattan, class=TRUE, numFolds = 10)$details['pctCorrect']
 / 100,
  evaluate_Weka_classifier(iris.knn9.minkowski, class=TRUE, numFolds = 10)$details['pctCorrect']
 / 100
)
```

Now we will create a plot, comparing the accuracy for each of the different distance calculations selected in the KNN models.

```
iris.comparison.dist <- data.frame(Distances=distancevalues,Accuracy=iris.accuracies.dist)
ggplot(iris.comparison.dist, aes(x=Distances, y=Accuracy)) + geom_bar(stat="identity")
```



Decision trees on the Contact Lens data

RWeka doesn't come with ID3 installed by default, so we need to add the additional package. Also, ID3 doesn't take numeric attributes, so we have to reload the data from the data set file and have all attributes as factors.

```
#if ( any(grepl("simpleEducationalLearningSchemes" , WPM("list-packages", "installed"))))
#  WPM("install-package", "simpleEducationalLearningSchemes")

WPM("load-package", "simpleEducationalLearningSchemes")
if ( ! any(grepl("ggvis", installed.packages()))) install.packages("ggvis")
if ( ! any(grepl("GGally", installed.packages()))) install.packages("GGally")
library(ggvis)
```

```
##
## Attaching package: 'ggvis'
```

```
## The following object is masked from 'package:ggplot2':
##
##     resolution
```

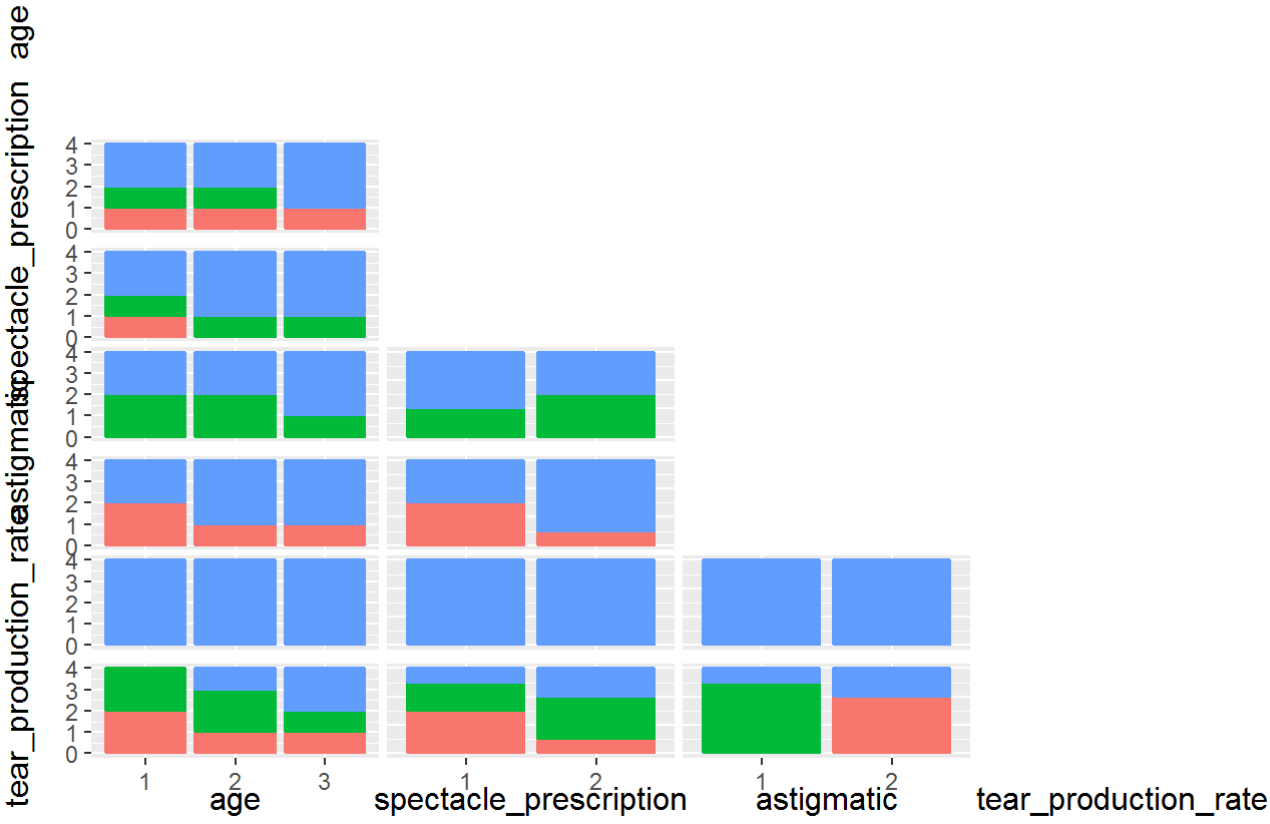
```
library(ggplot2)
library(GGally)
```

We load the lenses data in a separate data frame and do some data transformation to adapt all types as factors for the tree algorithms.

```
lenses2 <- lenses

lenses2$age = as.factor(lenses2$age)
lenses2$spectacle_prescription = as.factor(lenses2$spectacle_prescription)
lenses2$astigmatic = as.factor(lenses2$astigmatic)
lenses2$tear_production_rate = as.factor(lenses2$tear_production_rate)
```

We will identify the tree root using visualization libraries from R.

```
ggpairs(lenses2, mapping = aes(color = class, shape = age), columns = 1:4, diag = "blank", upper
 = "blank", legends = F)
```



For this analysis we will run different types of decision tree algorithms: ID3 and J48.

```
# ID3
weka_id3 <- make_Weka_classifier("weka/classifiers/trees/Id3")

lenses_id3 <- weka_id3(class~., data=lenses2, control=c("-C", 0.25, "-M", 2))
evaluate_Weka_classifier(lenses_id3, numFolds = 3, class=TRUE)
```

```
## === 3 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances          16               66.6667 %
## Incorrectly Classified Instances         8               33.3333 %
## Kappa statistic                          0.3402
## Mean absolute error                      0.2222
## Root mean squared error                  0.4714
## Relative absolute error                 59.375  %
## Root relative squared error            110.2492 %
## Total Number of Instances               24
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  0.250    0.200    0.200      0.250    0.222      0.046    0.525     0.175
  1
##                  0.400    0.000    1.000      0.400    0.571      0.588    0.700     0.525
  2
##                  0.867    0.444    0.765      0.867    0.813      0.450    0.711     0.746
  3
## Weighted Avg.    0.667    0.311    0.720      0.667    0.664      0.411    0.678     0.605

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##   1  0  3 |  a = 1
##   2  2  1 |  b = 2
##   2  0 13 |  c = 3
```

```
# J48
weka_j48 <- make_Weka_classifier("weka/classifiers/trees/J48")

# non-prunned version of J48 tree
lenses_j48_nonprunned <- weka_j48(class~., data=lenses2, control=Weka_control(U=TRUE))
evaluate_Weka_classifier(lenses_j48_nonprunned, numFolds = 3, class=TRUE)
```

```
## === 3 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances          20               83.3333 %
## Incorrectly Classified Instances         4               16.6667 %
## Kappa statistic                          0.71
## Mean absolute error                      0.1398
## Root mean squared error                  0.3099
## Relative absolute error                 37.3568 %
## Root relative squared error             72.4701 %
## Total Number of Instances               24
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  0.750    0.100    0.600      0.750    0.667      0.596    0.819     0.467
  1
##                  1.000    0.053    0.833      1.000    0.909      0.889    0.958     0.750
  2
##                  0.800    0.111    0.923      0.800    0.857      0.669    0.848     0.881
  3
## Weighted Avg.    0.833    0.097    0.851      0.833    0.836      0.703    0.866     0.785

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##   3  0  1 |  a = 1
##   0  5  0 |  b = 2
##   2  1 12 |  c = 3
```

```
#prunned version of J48 tree
lenses_j48_prunned <- weka_j48(class~., data=lenses2, control=Weka_control(R=TRUE))
evaluate_Weka_classifier(lenses_j48_prunned, numFolds = 3, class=TRUE)
```

```
## === 3 Fold Cross Validation ===
##
## === Summary ===
##
## Correctly Classified Instances          17               70.8333 %
## Incorrectly Classified Instances         7               29.1667 %
## Kappa statistic                          0.4783
## Mean absolute error                      0.1852
## Root mean squared error                  0.3469
## Relative absolute error                 49.4792 %
## Root relative squared error             81.1412 %
## Total Number of Instances               24
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  0.500    0.200    0.333      0.500    0.400      0.258    0.838     0.408
  1
##                  0.600    0.053    0.750      0.600    0.667      0.596    0.942     0.700
  2
##                  0.800    0.222    0.857      0.800    0.828      0.567    0.870     0.916
  3
## Weighted Avg.    0.708    0.183    0.748      0.708    0.723      0.522    0.880     0.786

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##   2  0  2 |  a = 1
##   2  3  0 |  b = 2
##   2  1 12 |  c = 3
```

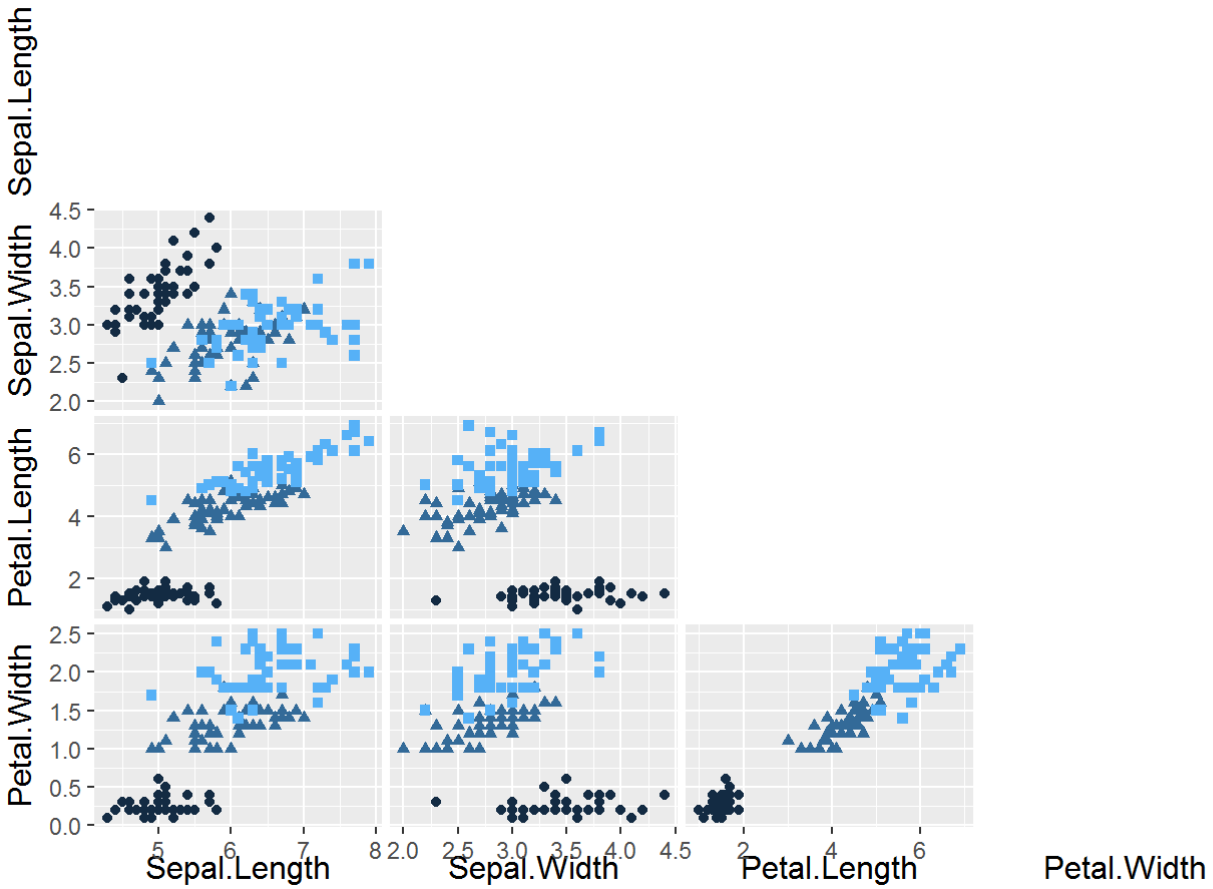
Decision trees on the Iris data

Same as with the contact lens data before, we need to convert all attributes from numeric to factors.

```
iris2 <- iris
# subset train-test data
set.seed(1234) # for reproducibility of the sample. change to get different random number to cre
ate sample from
ind1 <- sample(2, nrow(iris2), replace=TRUE, prob=c(0.67, 0.33))

iris2.training <- iris2[ind1==1, 1:4]
iris2.test <- iris2[ind1==2, 1:4]
iris2.trainLabels <- iris2[ind1==1, 5]
iris2.testLabels <- iris2[ind1==2, 5]
```

First we reload the iris data set, do some data type conversion and identify the node root with R visualization.s

```
data(iris)
iris$class= as.integer(as.factor(iris$Species))
ggpairs(iris, mapping = aes(color = class, shape = Species), columns = 1:4, diag = "blank", uppe
r = "blank", legends = F)
```



Now we create prunned and unprunned versions of the same J48 tree.

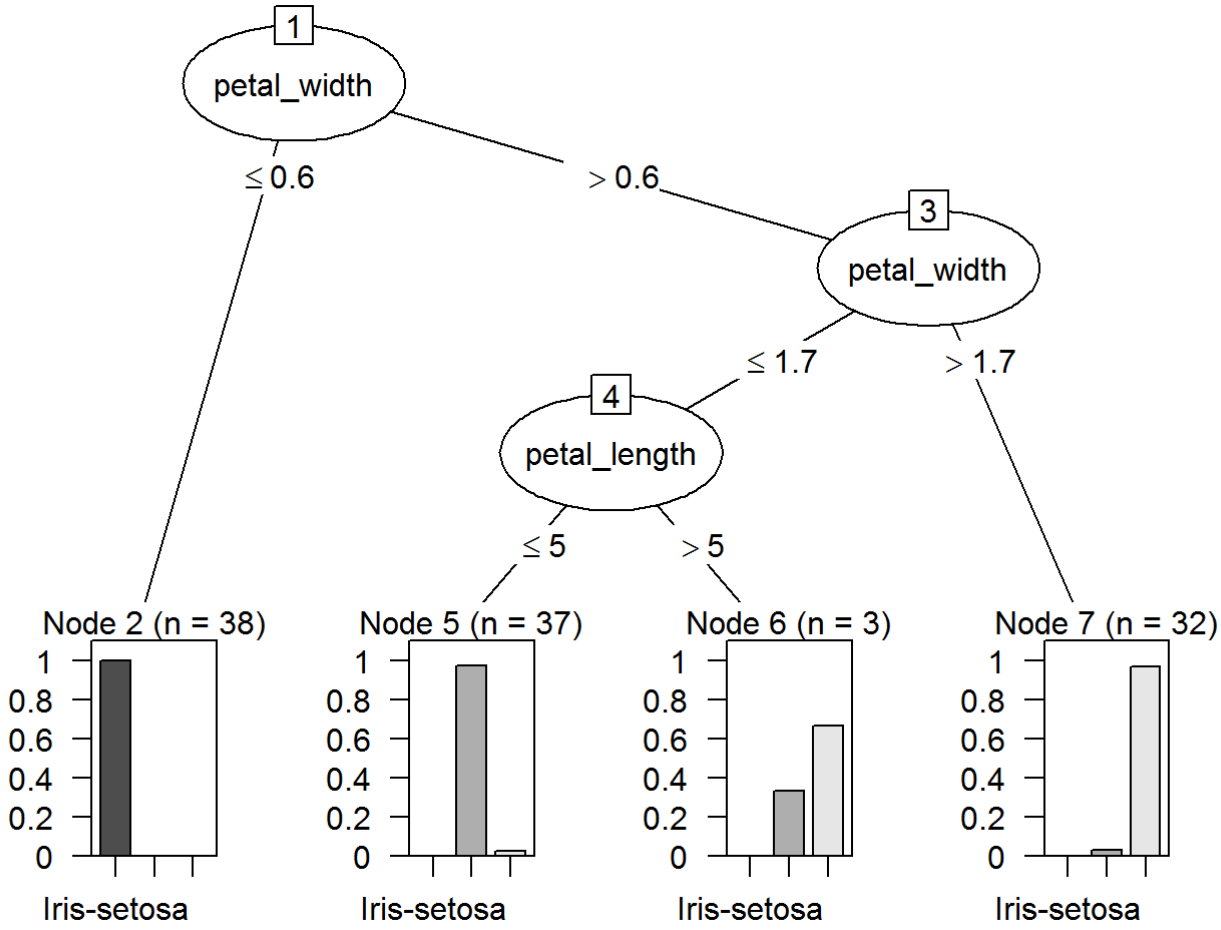
```
# fit models pruned and without prunning
fit_noprunned <- J48(iris2.trainLabels ~ ., data = iris2.training, control = Weka_control(U=TRUE))

fit_prunned <- J48(iris2.trainLabels ~ ., data = iris2.training, control = Weka_control(R=TRUE))

#make predictions
predictions_noprunned <- predict(fit_noprunned, iris2.test)
predictions_prunned <- predict(fit_prunned, iris2.test)

# summarize accuracy
evaluate_Weka_classifier(fit_noprunned, class=TRUE)
```

```
##
## === Summary ===
##
## Correctly Classified Instances         107               97.2727 %
## Incorrectly Classified Instances         3                2.7273 %
## Kappa statistic                          0.9591
## Mean absolute error                      0.0316
## Root mean squared error                  0.1257
## Relative absolute error                  7.123  %
## Root relative squared error             26.6893 %
## Total Number of Instances              110
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000
  Iris-setosa
##                  0.947    0.014    0.973      0.947    0.960      0.940    0.986     0.960
  Iris-versicolor
##                  0.971    0.026    0.943      0.971    0.957      0.937    0.985     0.953
  Iris-virginica
## Weighted Avg.    0.973    0.013    0.973      0.973    0.973      0.960    0.991     0.972

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##  38  0  0 |  a = Iris-setosa
##   0 36  2 |  b = Iris-versicolor
##   0  1 33 |  c = Iris-virginica
```

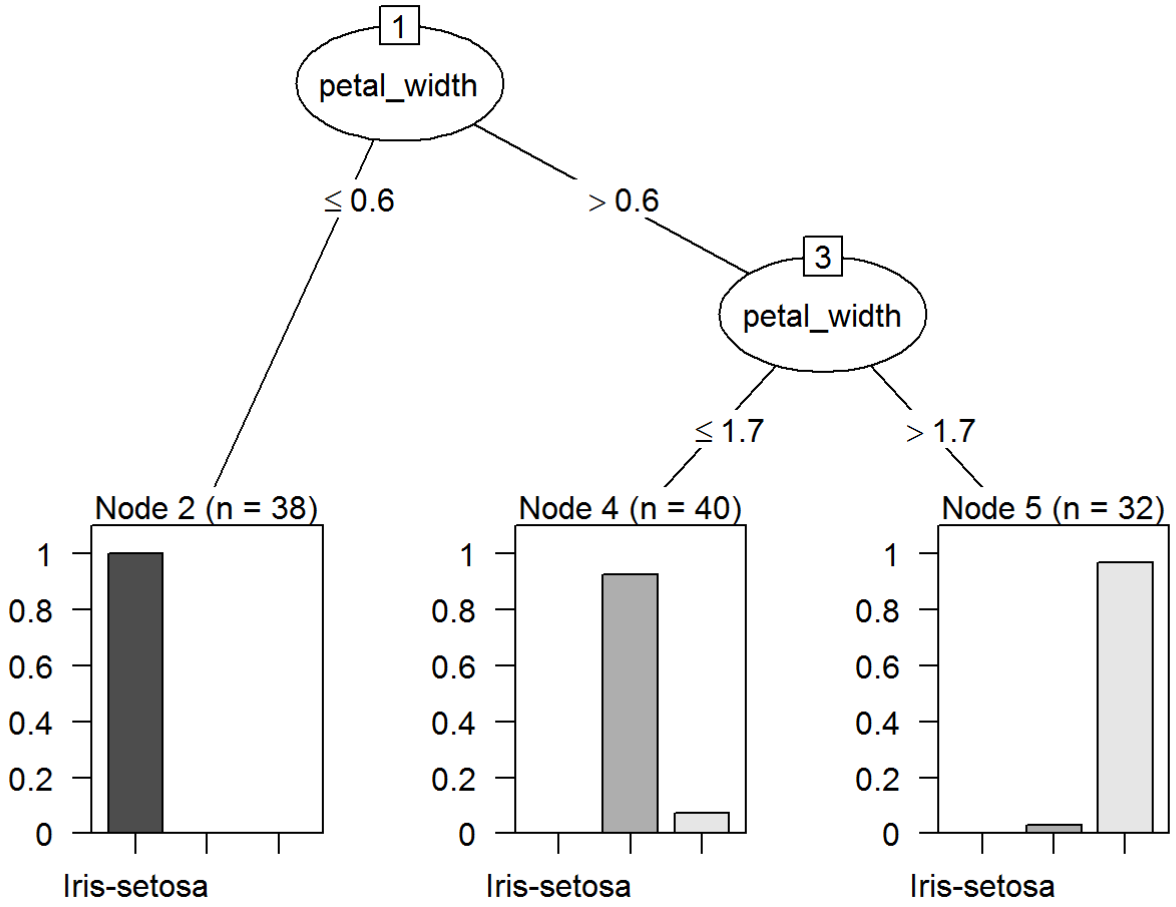
```
evaluate_Weka_classifier(fit_prunned, class=TRUE)
```

```
##
## === Summary ===
##
## Correctly Classified Instances         106               96.3636 %
## Incorrectly Classified Instances         4                3.6364 %
## Kappa statistic                          0.9453
## Mean absolute error                      0.0463
## Root mean squared error                  0.1521
## Relative absolute error                 10.4355 %
## Root relative squared error             32.283  %
## Total Number of Instances              110
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
  Class
##                  1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000
  Iris-setosa
##                  0.974    0.042    0.925      0.974    0.949      0.921    0.966     0.910
  Iris-versicolor
##                  0.912    0.013    0.969      0.912    0.939      0.914    0.971     0.925
  Iris-virginica
## Weighted Avg.    0.964    0.018    0.964      0.964    0.964      0.946    0.979     0.946

##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##  38  0  0 |  a = Iris-setosa
##   0 37  1 |  b = Iris-versicolor
##   0  3 31 |  c = Iris-virginica
```

```
#ploting the decision trees
plot(fit_noprunned)
```




```
plot(fit_prunned)
```



Multilayer Perceptron on Contact Lens data

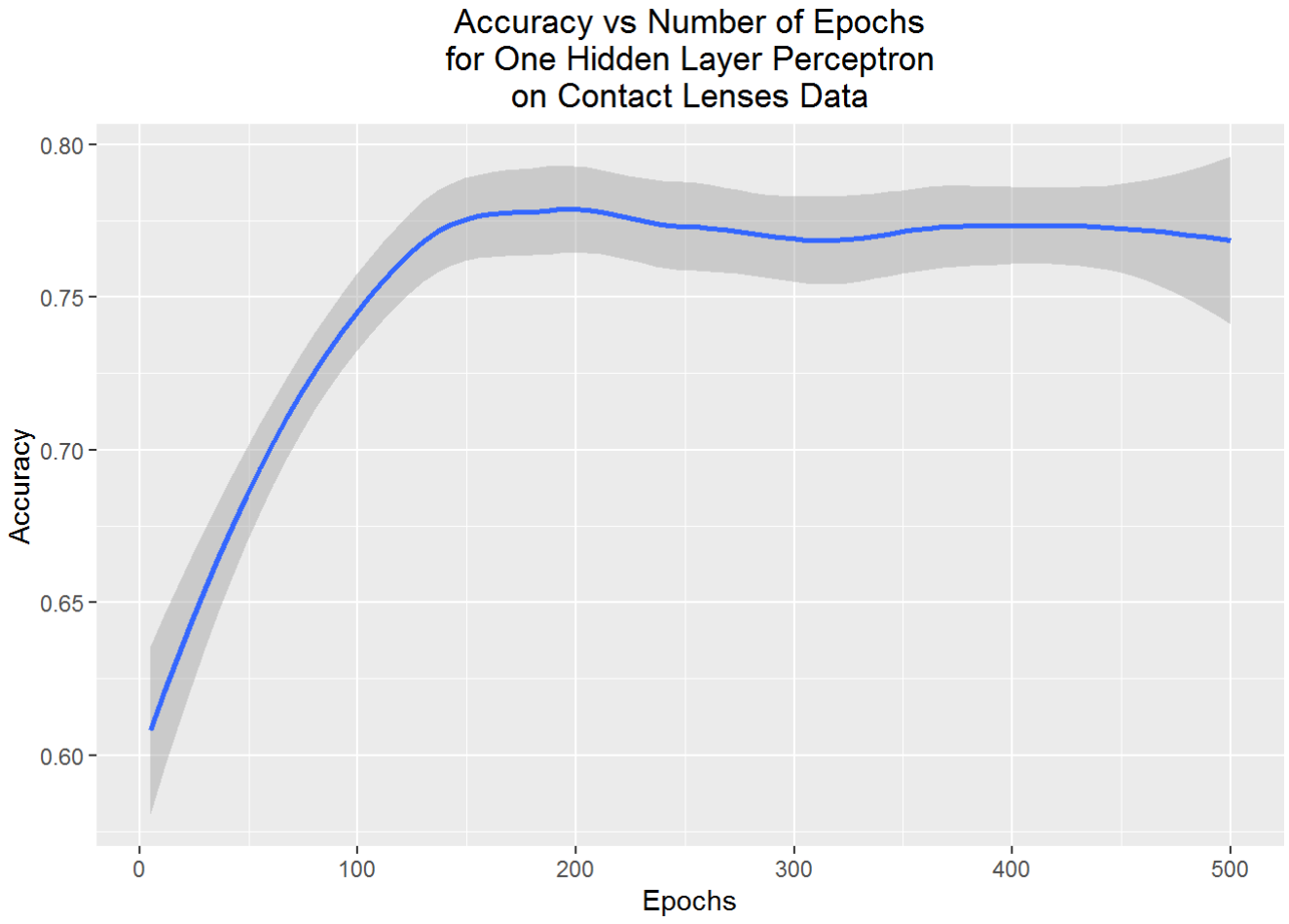
We will now create a multilayer perceptron network for the contact lens data.

First create the Multilayer percetron weka classifier and two empty vectors to hold the values for epochs and accuracies through different runs.

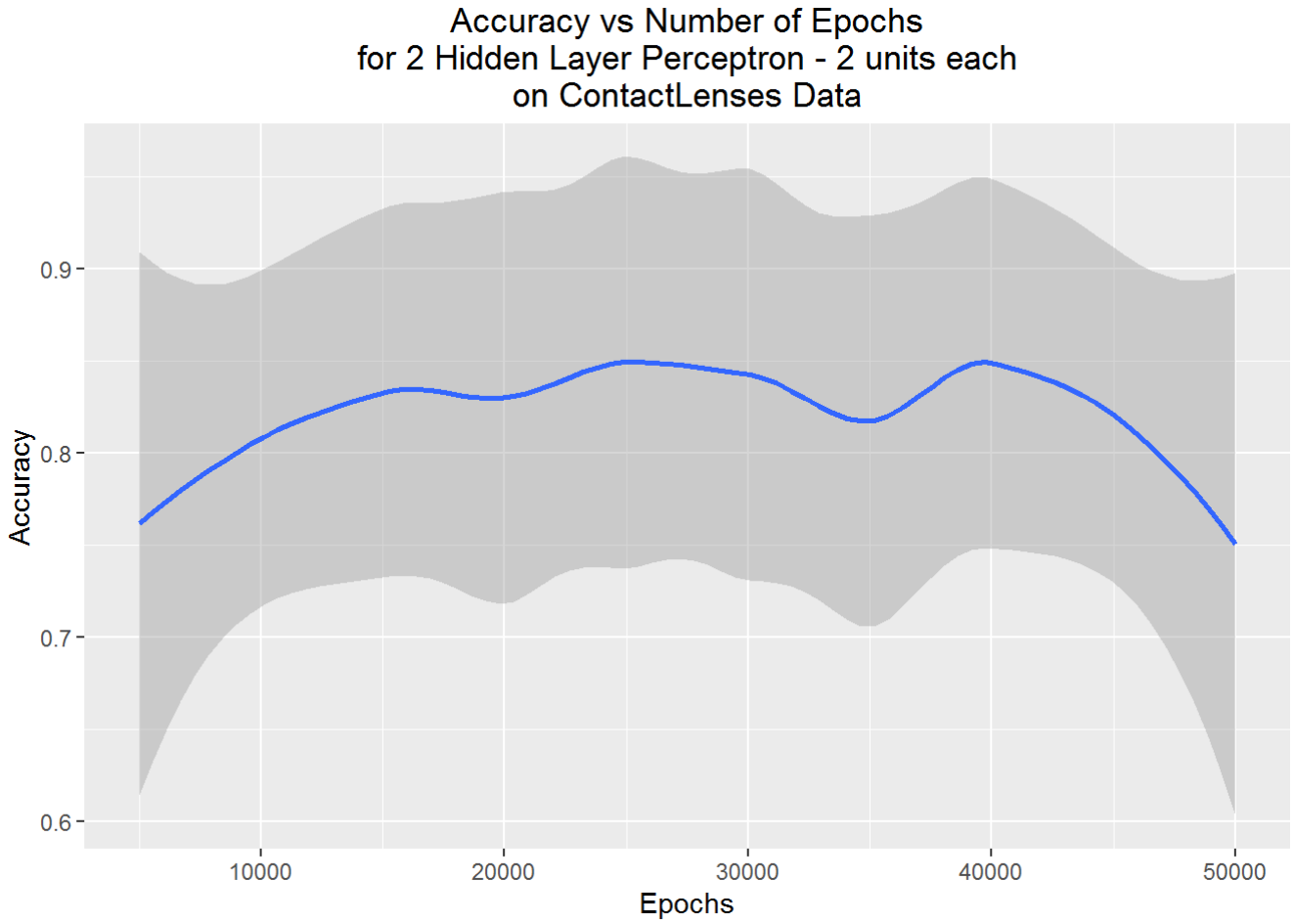
```
weka_mlp <- make_Weka_classifier("weka/classifiers/functions/MultilayerPerceptron")

epochs <- c()
accuracies <- c()
```

Now iterate through different a fixed set of epochs (5-500 in steps of 5) to get accuracy values. We collect the values in the empty vectors created above and then plot them.

```
# multilayer perceptron classifier
for(i in 1:100) # do 100 iterations
{
  epochs <- append(epochs, 5  *i) # accumulate the epoch number over a vector for plotting
  lenses_mlp_a <- weka_mlp(class~., # class is the network output
                           data=lenses, # data comes from the lenses data frame
                           control=c("-L", 0.3, # learning rate
                                     "-M", 0.2, # momentum
                                     "-N", 5 * i, # number of epochs, changes through iterations
                                     "-V", 0, # validation set size
                                     "-S", 0, # seed number
                                     "-E", 20, # validation threshold
                                     "-H", "a" # hidden layers
                                     ))
  accuracies <- append (accuracies, evaluate_Weka_classifier(lenses_mlp_a, numFolds = 10,
class=TRUE)$details['pctCorrect'] / 100) # accumulate the accuracy over a vector for plotting
}
nn.epochs.comparison <- data.frame(Epochs=epochs,Accuracy=accuracies) # create data frame of epo
chs vs accuracies
ggplot(nn.epochs.comparison, aes(Epochs, Accuracy)) + # plot epochs vs accuracies
  geom_smooth() + # smooth the values
  ggtitle("Accuracy vs Number of Epochs\nfor One Hidden Layer Perceptron\non Contact Lenses Dat
a") #add title
```



```
#two hidden layers, two units each
rm(epochs); rm(accuracies); epochs <- c(); accuracies <- c();
for(i in 1:10) # do 100 iterations
{
  epochs <- append(epochs, 5000 * i)
  lenses_mlp_a <- weka_mlp(class~., data=lenses, control=c("-L", 0.3, "-M", 0.2, "-N", 5000 * i,
 "-V", 0, "-S", 0, "-E", 20, "-H", "a,2,2" ))
  accuracies <- append (accuracies, evaluate_Weka_classifier(lenses_mlp_a, numFolds = 10,
class=TRUE)$details['pctCorrect'] / 100)
}
nn.epochs.comparison <- data.frame(Epochs=epochs,Accuracy=accuracies)
ggplot(nn.epochs.comparison, aes(Epochs, Accuracy)) + geom_smooth() + ggtitle("Accuracy vs Numbe
r of Epochs\nfor 2 Hidden Layer Perceptron - 2 units each\non ContactLenses Data")
```

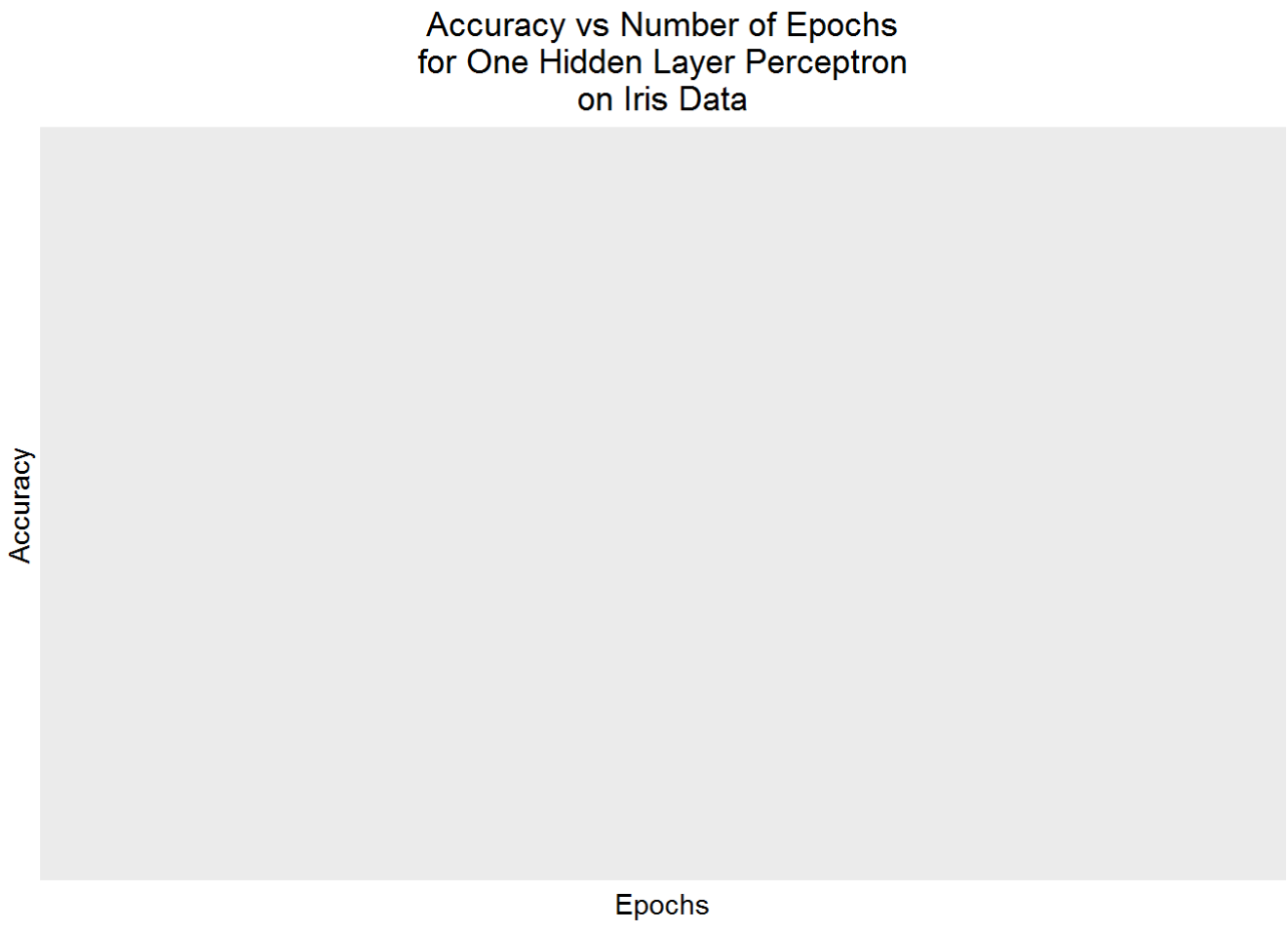


Multilayer Perceptron on Iris data

We will now create a multilayer perceptron network for the Iris data.

```
#multilayer perceptron classifier
rm(epochs); rm(accuracies); epochs <- c(); accuracies <- c(); rm(nn.epochs.comparison);
weka_mlp <- make_Weka_classifier("weka/classifiers/functions/MultilayerPerceptron")

# one hidden layer
for(i in 1:100)
{
  epochs <- append(epochs, 5  *i)
  iris_mlp_a <- weka_mlp(class~., data=iris, control=c("-L", 0.3, "-M", 0.2, "-N", 5 * i, "-V", 0,
 "-S", 0, "-E", 20, "-H", "a" ))
  accuracies <- append (accuracies, evaluate_Weka_classifier(iris_mlp_a, numFolds = 10, class=TR
UE)$details['pctCorrect'] / 100)
}
nn.epochs.comparison <- data.frame(Epochs=epochs,Accuracy=accuracies)
ggplot(nn.epochs.comparison, aes(Epochs, Accuracy)) + geom_smooth() + ggtitle("Accuracy vs Numbe
r of Epochs\nfor One Hidden Layer Perceptron\non Iris Data")
```

```
## Warning: Removed 100 rows containing non-finite values (stat_smooth).
```



```
# two hidden layers, two units each
rm(epochs); rm(accuracies); epochs <- c(); accuracies <- c();
for(i in 1:10)
{
  epochs <- append(epochs, 5000 * i)
  iris_mlp_a <- weka_mlp(class~., data=iris, control=c("-L", 0.3, "-M", 0.2, "-N", 5000 * i, "-
V", 0, "-S", 0, "-E", 20, "-H", "a,2,2" ))
  accuracies <- append (accuracies, evaluate_Weka_classifier(iris_mlp_a, numFolds = 10, class=TR
UE)$details['pctCorrect'] / 100)
}
nn.epochs.comparison <- data.frame(Epochs=epochs,Accuracy=accuracies)
ggplot(nn.epochs.comparison, aes(Epochs, Accuracy)) + geom_smooth() + ggtitle("Accuracy vs Numbe
r of Epochs\nfor 2 Hidden Layer Perceptron - 2 units each\non Iris Data")
```

```
## Warning: Removed 10 rows containing non-finite values (stat_smooth).
```