Data Mining In R

Connor Segneri

9/15/2017

Contents

The following are popular data mining techniques. They are all described in terms of their strengths and weaknesses, and an example is provided for each.

Classification

"In classification learning, a classifier is presented with a set of examples that are already classified and, from these examples, the classifier learns to assign unseen examples." (Data Science and Big Data Analytics, 192). Classification algorithms simply assign labels to new observations of data. The set of labels used in classification must be predetermined. Most classification methods are supervised, which means that they discover how to label new observations by first analyzing a training set.

Some Classification Methods:

- Logistic Regression
- Decision Trees
- Support Vector Machines

For this example decision trees are the specific classification method used. "A decision tree (also called a prediction tree) uses a tree structure to specify sequences of decisions and consequences." (Data Science and Big Data Analytics, 192).

The `MASS` library has multiple data sets that will be used for the various examples in this report.

```
library(MASS)
set.seed(1)
```

The `Boston` data set is used in this example. It contains housing and neighborhood information for the suburbs of Boston with the following columns:

- crim: per capita crime rate by town
- zn: proportion of residential land zoned for lots over 25,000 sq. ft
- indus: proportion of non-retail business acres per town
- chas: Charles River dummy variable (=1 if tract bounds river; 0 otherwise)
- nox: nitrogen oxides concentration
- rm: average number of rooms per dwelling
- age: proportion of owner occupied units built prior to 1940
- dis: weighted mean of distances to five Boston employment centres
- rad: index of accessibility to radial highways
- tax: full-value property-tax rate per $10,000
- ptratio: pupil-teacher ratio by town
- black: 1,000(Bk-0.63)^2 where Bk is the proportion of African Americans by town
- lstat: lower status of the population (percent)

- medv: median value of owner-occupied homes in $1,000s

The main purpose of this example is to use a decision tree to predict whether a home is located along Charles River or not given the rest of the information listed above.

First, the data is randomly split in half, one half being the training set and the other being the test set.

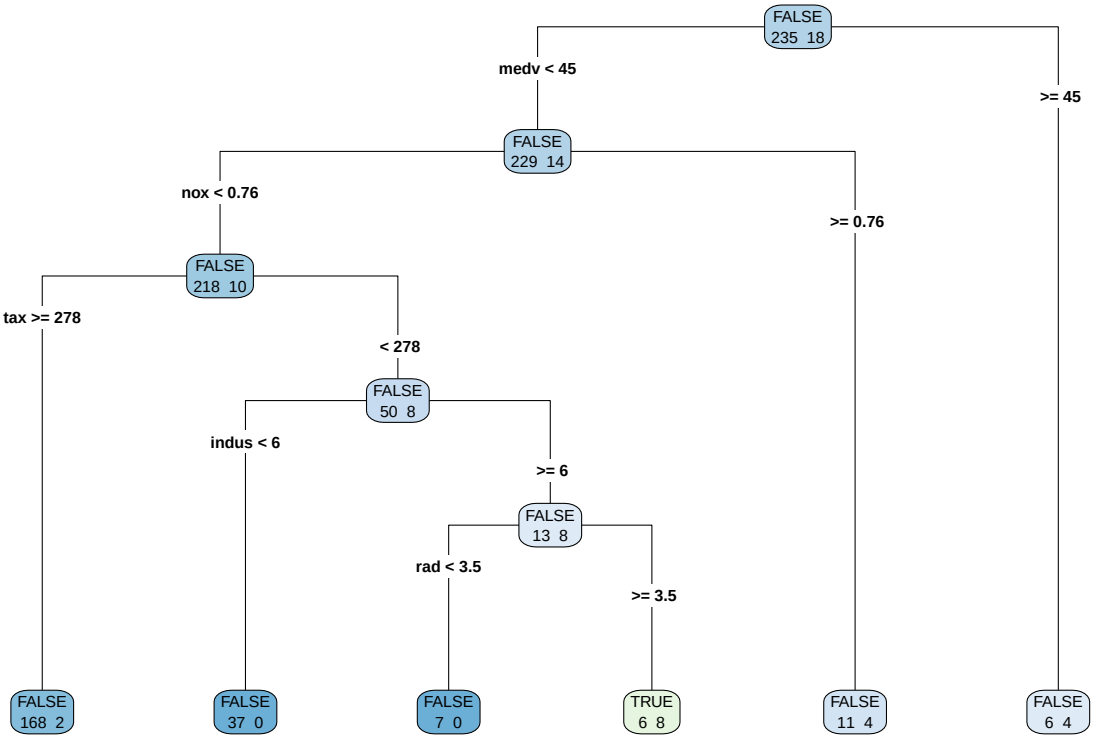
```
Boston$chas = as.logical(Boston$chas)
train = sample(dim(Boston)[1], dim(Boston)[1]/2)
BostonTrain = Boston[train,]
BostonTest = Boston[-train,]
```

Next, a classification decision tree is made from the training data using the `rpart()` method.

```
library(rpart)
library(rpart.plot)
riverFit = rpart(chas~., method="class", data=BostonTrain)
```

The classification tree can be seen in the plot below.

```
rpart.plot(riverFit, type=4, extra=1)
```



The classification tree is used to predict whether homes are located along Charles River using the information in the testing data set. The accuracy is found by comparing the `chas` column in the testing data set with predictions that are made from the rest of the information in the same testing set. The percentage of accurately predicted classifications can be seen below.

```
riverPred = predict(riverFit, newdata=BostonTest, type="class")
accuracy = riverPred == BostonTest$chas
table(accuracy)
```

```
## accuracy
## FALSE  TRUE
##    20   233
```

```
percentCorrect = sum(accuracy==1) / length(accuracy)
percentCorrect
```

```
## [1] 0.9209486
```

Prediction

Prediction means to declare or indicate something in advance. In data mining, prediction is when a specific outcome can be declared before it actually happens. Prediction is never full proof however, and even if a trend seems to be leading towards a concrete outcome, there is never an absolute guarantee that the outcome is going to happen. Predictive modelling is an attempt to predict outcomes using statistics.

Most data mining models can be used to predict outcomes (in fact decision trees were used to predict outcomes in the last section). Here are some more examples:

- Naive Bayes
- Support Vector Machines
- Decision Trees
- Linear Regression
- Logistic Regression
- Neural Networks

In this example, linear regression is going to be used to predict the value of a dependent variable, given the value of an independent variable. "Linear regression is an analytical technique used to model the relationship between several input variables and a continuous outcome variable." (Data Science and Big Data Analytics, 162). An important assumption is made before any linear regression, and this is that the relationship between the input variable and the output variable is linear.

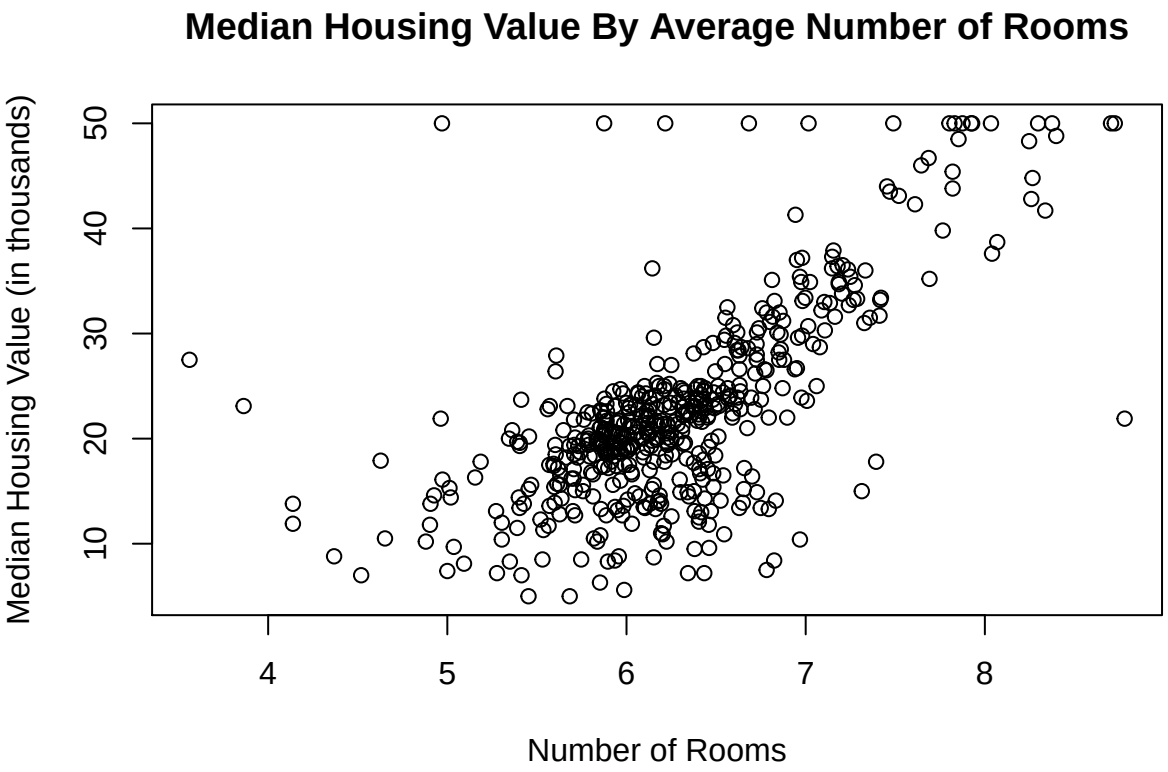
For the linear regression example, the `Boston` data set is going to be utilized again. After looking at the relationships within this data set (using the `pairs()` function, which displays scatter plots of all the variables in the `Boston` data frame), it appears that there may be a linear relationship between the average number of rooms in a household and the median household value. This suspicion is explored below.

The data is read in the same way as before.

```
library(MASS)
set.seed(1)
```

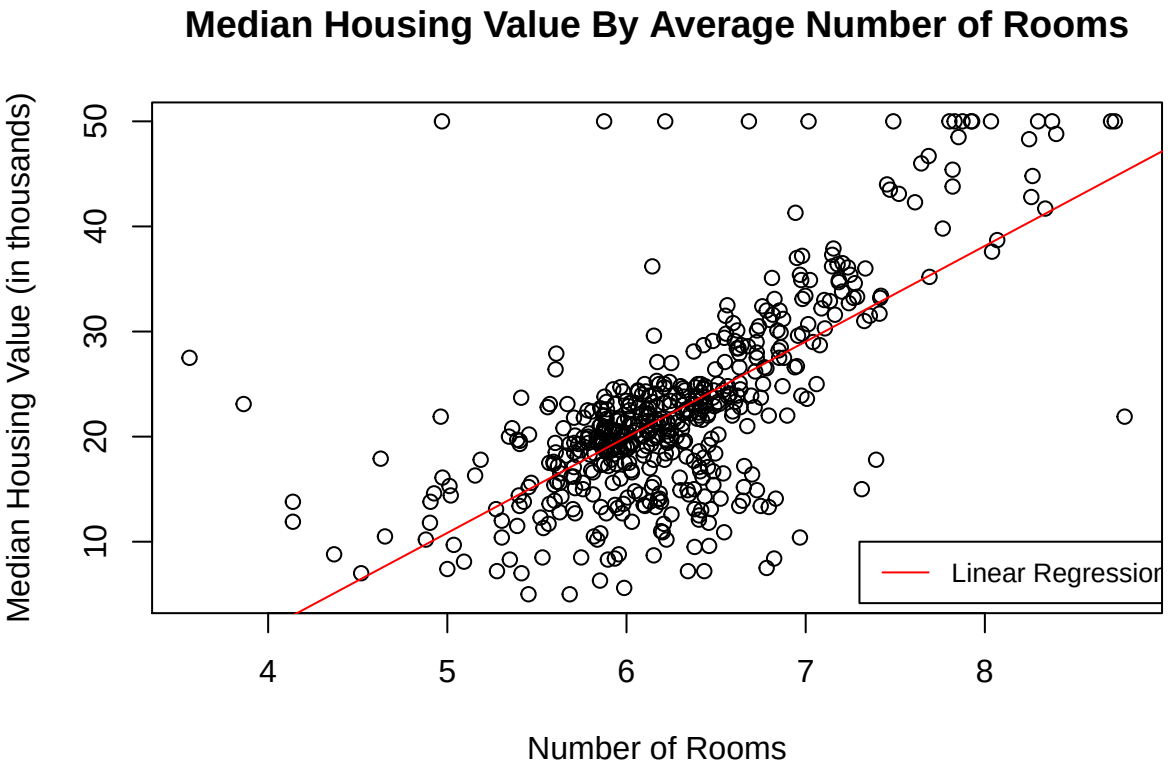
First, it is always a good idea to view the data, so the characteristics in question are plotted against each other.

```
plot(Boston$rm, Boston$medv, xlab="Number of Rooms", ylab="Median Housing Value (in thousands)",
     main="Median Housing Value By Average Number of Rooms")
```



It appears that there may be a linear correlation, so a linear regression is created from the `rm` and `medv` columns of the `Boston` data frame. This regression line is then plotted.

```
plot(Boston$rm, Boston$medv, xlab="Number of Rooms", ylab="Median Housing Value (in thousands)",
     main="Median Housing Value By Average Number of Rooms")
linFit = lm(medv ~ rm, data=Boston)
abline(linFit, col="red")
legend(7.3, 10, legend=c("Linear Regression"), col=c("red"), lty=1, cex=0.8)
```



When looking at a summary of the linear fit, it becomes possible to see how closely these two variables are correlated. In the Coefficients section of the summary, the associated hypothesis tests' p-value (the Pr(>|t|) column) for the `rm` parameter is <2e-16. Since the p-value is so small, this suggests that there is a strong correlation between `rm` and `medv`. Another summary statistic to look at is the residual standard error, which "can be used to examine the variance of the assumed normally distributed error terms." (Data Science and Big Data Analytics, 170). The adjusted r-squared goes from 0 to 1, and the closer the value is to 1, the better the fit. The adjusted R-squared value for this fit is 0.4686. This means that the linear regression does **not** provide a very good model for explaining the data. Adding more variables or performing another type of regression may improve the current model, but this fit is still fine for the purposes of this example.

```
summary(linFit)
```

```
##
## Call:
## lm(formula = medv ~ rm, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.346  -2.547   0.090   2.986  39.433
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -34.671      2.650  -13.08   <2e-16 ***
## rm             9.102      0.419   21.72   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.616 on 504 degrees of freedom
## Multiple R-squared:  0.4835, Adjusted R-squared:  0.4825
## F-statistic: 471.8 on 1 and 504 DF,  p-value: < 2.2e-16
```

As seen in the plots above, the average number of rooms per household never exceeds 9. The linear model can be used to predict the median housing value when the average number of rooms per household is 9, 10, or 11.

```
newData = data.frame(rm = c(9,10,11))
linPred = predict(linFit, newData)
linPred
```

```
##        1        2        3
## 47.24836 56.35047 65.45258
```

Clustering

"In general, clustering is the use of unsupervised techniques for grouping similar objects." (Data Science and Big Data Analytics, 118). Unsupervised learning is when the data is unlabeled and the purpose is to find hidden structures within the data. Clustering is used to find similarities between objects and object attributes, and it is not used in any type of predictive analytics.

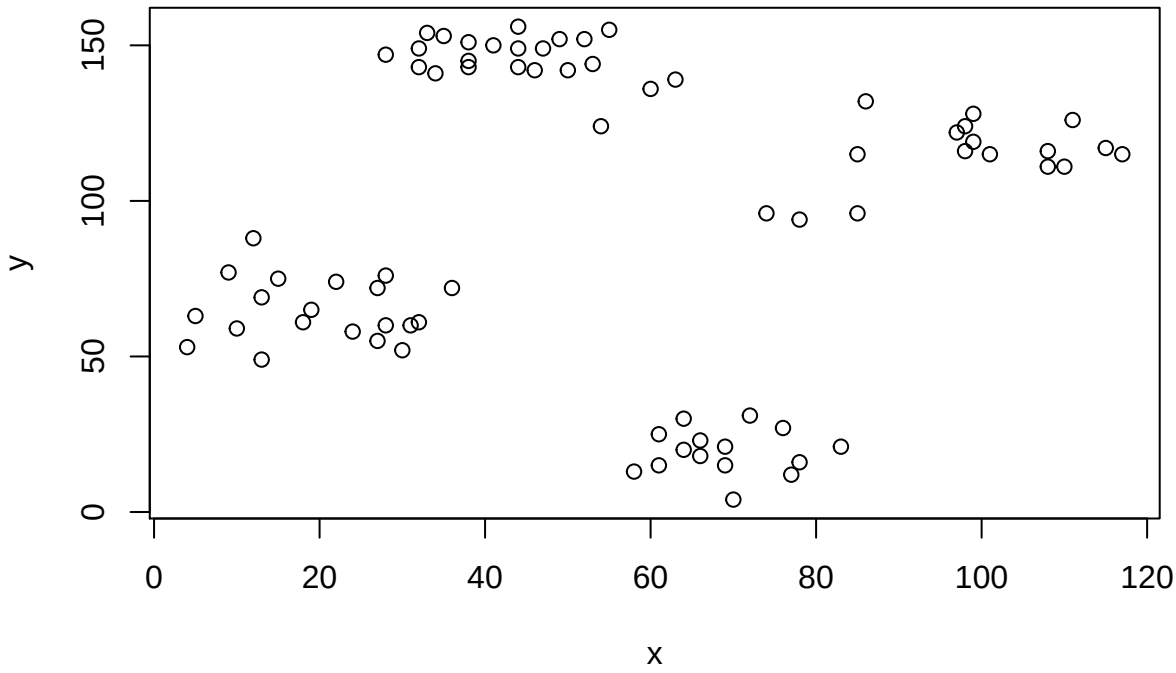
Some examples of clustering techniques:

- K-means
- Gaussian
- Hierarchical

In this example, k-means is going to be used to form clusters from the `Ruspini` data set. This data set does not display any specific information, as it was made for the purpose of displaying clusters of data. "Given a collection of objects with n measurable attributes, k-means is an analytical technique that, for a chosen value of k, identifies k clusters of objects based on the objects' proximity to the center of the k groups." (Data Science and Big Data Analytics, 118).

First, the data should be read in and plotted so that a number of clusters, k, can be discerned.

```
library(ggplot2)
set.seed(1)
ruspini = read.csv("ruspini.csv")
ruspini = ruspini[,2:3]
plot(ruspini)
```



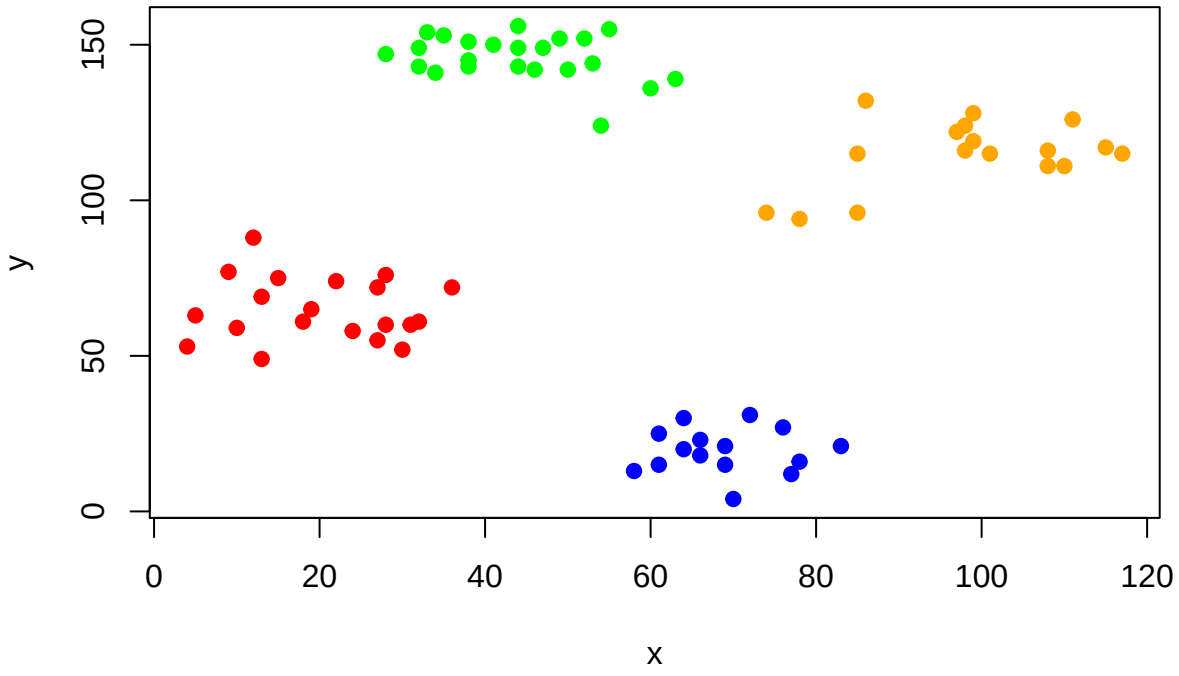
From the above plot, it seems that there should be 4 clusters, so when deploying k-means, the k variable is set to 4. Once the k-means is computed, the cluster means, along with other information, are displayed. These are the 4 points that represent the mean of their respective clusters.

```
ruspiniKM = kmeans(ruspini, 4)
ruspiniKM
```

```
## K-means clustering with 4 clusters of sizes 20, 23, 17, 15
##
## Cluster means:
##          x        y
## 1 20.15000  64.9500
## 2 43.91304 146.0435
## 3 98.17647 114.8824
## 4 68.93333  19.4000
##
## Clustering vector:
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4
## [71] 4 4 4 4 4
##
## Within cluster sum of squares by cluster:
## [1] 3689.500 3176.783 4558.235 1456.533
##  (between_SS / total_SS =  94.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

The 4 clusters are colored and added to the plot.

```
colors = c("red", "green", "orange", "blue")
plot(ruspini[,1:2], pch = 19, col = colors[ruspiniKM$cluster])
```



Association Rules

Association rules is an unsupervised learning method. Like clustering, this is a descriptive analytical method, **not** a predictive one. This technique is used to "discover interesting relationships hidden in a large data set." (Data Science and Big Data Analytics, 138).

Examples of association rules techniques:

- Apriori
- Eclat
- FP-growth

In this example, the apriori algorithm is used. "The Apriori algorithm takes a bottom-up iterative approach to uncovering the frequent itemsets by first determining all the possible items and then identifying which among them are frequent." (Data Science and Big Data Analytics, 140). The algorithm uses a minimum support criteria to determine which itemsets remain and which are pruned. After the pruning, the algorithm pairs all of the 1-itemsets into 2-itemsets and determines which among these are frequent by looking at the data set. Then the minimum support criteria is used to prune again, and the process is repeated until no more itemsets can be made.

First, the data is read in. The `animals` data set is going to be used in this example. This data set is located in the `cluster` library.

```
library(arules)
library(arulesViz)
library(cluster)

animals = na.omit(animals)
animals[animals==1] = 0
```

```
animals[animals==2] = 1
animals = sapply(animals, function(x) as.logical(x))
```

The `animals` data set contains 20 observations of different animals, and identifies 6 binary characteristics for each animal:

- war: warm-blooded
- fly: can fly
- ver: vertebrate
- end: endangered
- gro: live in groups
- hai: have hair

The first observation is for an ant. The FALSE in the `war` column indicates that ants are not warm-blooded. The TRUE in the `gro` column indicates that ants live in groups.

```
animals[1,]
```

```
##   war   fly   ver   end   gro   hai
## FALSE FALSE FALSE FALSE  TRUE FALSE
```

The apriori algorithm is going to be utilized with this data to determine if there are any associations between the various traits.

This can be done using the `apriori()` function. The `support` variable in this function is set to the desired minimum support criteria. Since it is set to 0.20, this means that the itemsets need to appear in at least 20% of the rows in order to be considered a frequent itemset.

```
itemsets = apriori(animals, parameter = list(minlen=1, maxlen=6,
          support=0.20, target="frequent itemsets"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##          NA    0.1    1 none FALSE            TRUE       5     0.2      1
##  maxlen            target   ext
##       6 frequent itemsets FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 3
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[6 item(s), 15 transaction(s)] done [0.00s].
## sorting and recoding items ... [6 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [24 set(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

The element length distribution section of the summary shows that, at a 20% minimum support distribution, 6 1-itemsets, 9 2-itemsets, 7 3-itemsets, and 2 4-itemsets were found.

```
summary(itemsets)
```

```
## set of 24 itemsets
##
## most frequent items:
##     war     ver     gro     end     hai (Other)
##      12      12      12       8       8       1
##
## element (itemset/transaction) length distribution:sizes
## 1 2 3 4
## 6 9 7 2
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.750   2.000   2.208   3.000   4.000
##
## summary of quality measures:
##     support           count
##  Min.   :0.2667   Min.   : 4.000
##  1st Qu.:0.2667   1st Qu.: 4.000
##  Median :0.3333   Median : 5.000
##  Mean   :0.3889   Mean   : 5.833
##  3rd Qu.:0.4667   3rd Qu.: 7.000
##  Max.   :0.7333   Max.   :11.000
##
## includes transaction ID lists: FALSE
##
## mining info:
##     data ntransactions support confidence
##  animals            15     0.2          1
```

The following code displays just the 4-itemsets. The support column indicates the percentage of the data set that this itemset was found in. So the 2 sets of 4 characteristics displayed below were all present together about 27% of the time.

```
inspect(tail(sort(itemsets, by='support'),2))
```

```
##     items              support   count
## [1] {war,ver,end,gro} 0.2666667 4
## [2] {war,ver,gro,hai} 0.2666667 4
```

This information may be useful to biologists who are trying to analyze which animal characteristics tend to exist together in various animals. Knowing that animals who are warm-blooded also tend to have vertebrates could assist them in organizing the animal kingdom.

Correlation Analysis

Correlation is "the measure of influence that a set of variables has on the outcome of another variable of interest." (Data Science and Big Data Analytics, 162). This concept is very similar to that of linear regression, but the two are not the same. Linear regression is fitting a line through data points, while correlation is computing a coefficient that describes the change in one variable as another changes. In order to conduct a correlation analysis, the two variables being analyzed must be quantitative.

In this example, the `Boston` data set will be utilized again.

```
library(MASS)
```

The code below performs a correlation analysis on the two variables that were used in the linear regression example, `rm` and `medv`. As before, `rm` is the average number of rooms per dwelling, and `medv` is the median

value of owner-occupied homes in $1,000s. The output displays the correlation coefficient, which can range from -1 to 1. If the value is zero, then there is no correlation. As the value decreases to -1, the correlation is more significant, and `medv` would decrease as `rm` increases. As the value increases to 1, the correlation is more significant, and `medv` would increase as `rm` increases. Because the coefficient is 0.6953599, this means that there is a strong correlation between `medv` and `rm`, and as `rm` increases, `medv` increases.

```
cor.test(Boston$rm, Boston$medv)
```

```
##
##  Pearson's product-moment correlation
##
## data:  Boston$rm and Boston$medv
## t = 21.722, df = 504, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6474346 0.7378075
## sample estimates:
##       cor
## 0.6953599
```

There is no predictive power in correlation analysis. This is useful for determining the strength of the linear relationship, but in order to predict `medv` given some value of `rm`, a linear regression would need to be performed, like the one computed in the **Prediction** section of this report.

References

(n.d.). Retrieved September 17, 2017, from http://rmarkdown.rstudio.com/authoring_quick_tour.html

Association rule learning. (2017, September 14). Retrieved September 17, 2017, from https://en.wikipedia.org/wiki/Association_rule_learning

U. (2014, February 24). Correlation in RStudio. Retrieved September 17, 2017, from https://www.youtube.com/watch?v=xsL4yLBNyDg

Dietrich, D., Heller, B., & Yang, B. (2015). Data science & big data analytics: discovering, analyzing, visualizing and presenting data. Indianapolis, IN: Wiley.

Ondemand. (n.d.). Retrieved September 17, 2017, from https://www.coursera.org/learn/data-patterns/lecture/6qIY8/2-2-the-apriori-algorithm

Predictive modelling. (2017, September 08). Retrieved September 17, 2017, from https://en.wikipedia.org/wiki/Predictive_modelling