MSDS680 - Week 5 - ANN and SVM

Benjamin Siebold

Regis University

MSDS680 - Week 5 - ANN and SVM

Introduction

In this project, both Support Vector Machines and Neural Networks will be explored using the Mushroom dataset from UCI to predict whether or not a mushroom is edible or poisonous. The mushroom dataset contains 8124 rows and 23 variables. This includes 22 features and one label (edible, poisionous). The muhsroom dataset does not include full descriptions of each feature, rather an abbreviation or indicator for every variable. These features are attributes of the mushroom, such as color, texture, smell, along with where it grows. To access the dataset and find the values for each column please refer to http://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/.

Methodology

Set up

As is the first step in any analysis, the data needs to be loaded in, the necessary libraries and packages for exploration, cleaning, and model application need to be installed and loaded int the instance, the seed is set for repeateable results, and because there are not column names in the reading of the data in this instance column names will be applied to the dataset.

```r
library(data.table) #used to read in data
library(DataExplorer)#used to inspect the data and get initial understanding
library(e1071) #used for SVM model
library(caret) #used for model analysis
library(class) #used for converting data types
library(dbplyr) #used for removing/adding columns
library(neuralnet) #used for neural net model
```

```r
set.seed(476)

mushroom_data <- read.table(
  'http://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.da
  stringsAsFactors = T, header = F, sep = ',')

col_names <- c('class', 'cap.shape', 'cap.surface', 'cap.clr', 'bruises', 'odor',
               'gill.attach', 'gill.spacing', 'gill.sz', 'gill.clr',
               'stalk.shape', 'stalk.root', 'stalk.sar', 'stalk.sbr',
               'stalk.clr.abv.rng', 'stalk.clr.bel.rng', 'veil.type', 'veil.clr',
               'ring.num', 'ring.type', 'spore.prnt.clr', 'pop', 'habitat')

names(mushroom_data) <- col_names
```

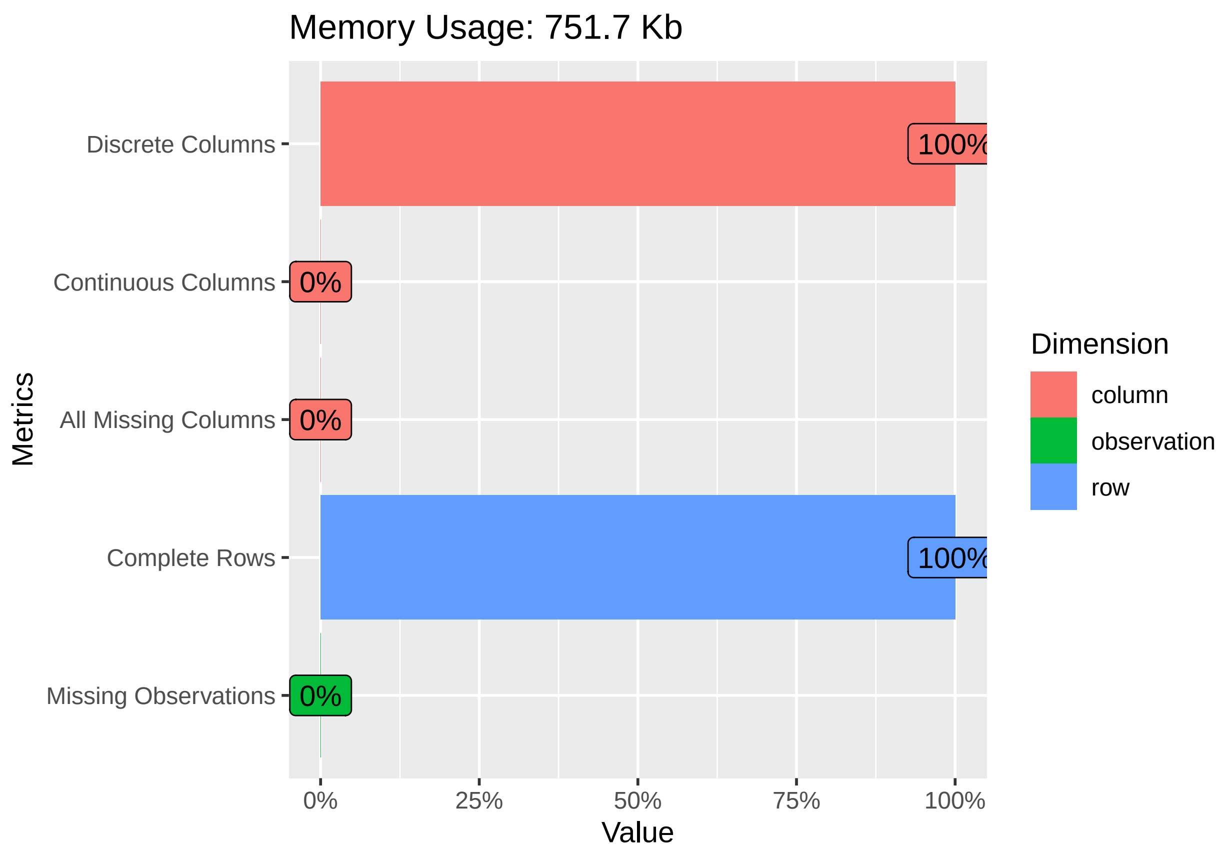
Data Exploration

With the data loaded, exploration of what exists in the data can begin.

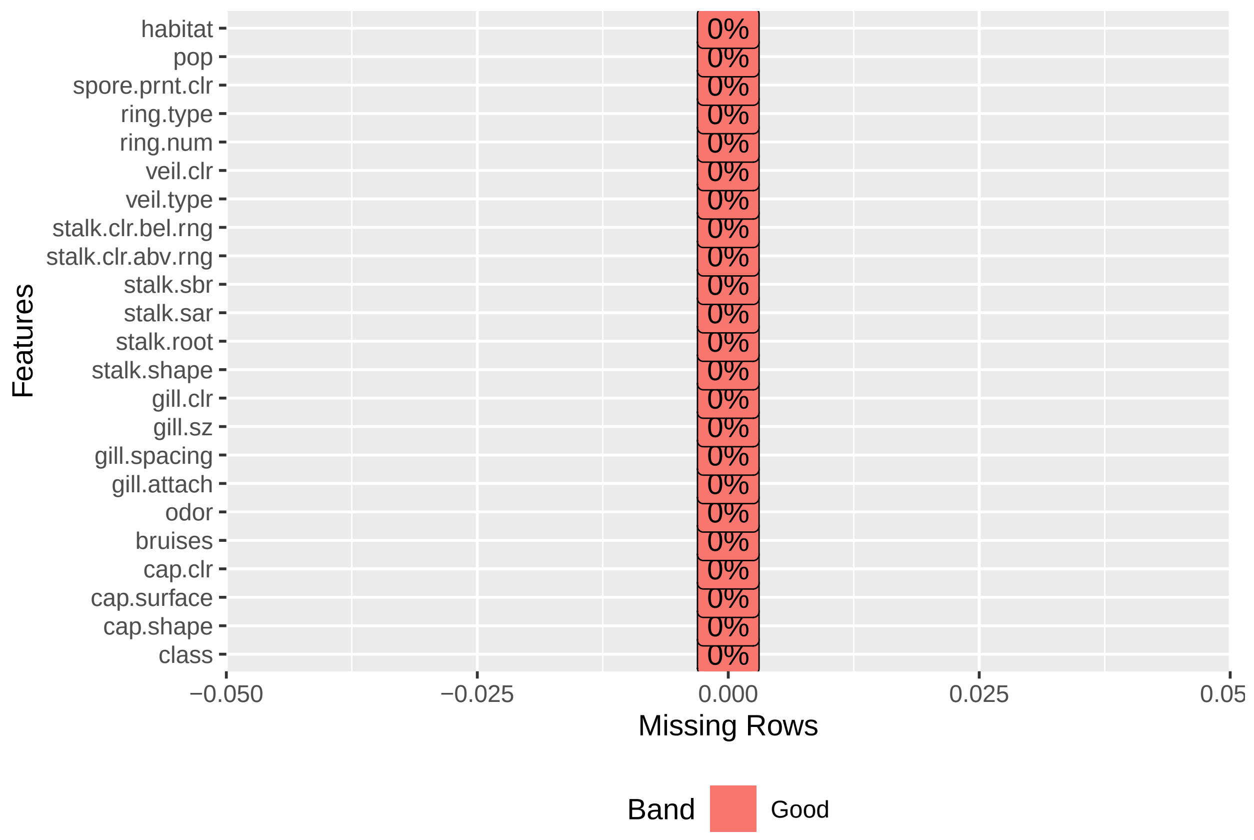
```r
# initial data inspectio
summary(mushroom_data) # provides initial idea of what data looks like
```

```
##  class    cap.shape cap.surface    cap.clr      bruises       odor
##  e:4208   b: 452    f:2320      n      :2284   f:4748   n      :3528
##  p:3916   c:   4    g:   4      g      :1840   t:3376   f      :2160
##           f:3152    s:2556      e      :1500            s      : 576
##           k: 828    y:3244      y      :1072            y      : 576
##           s:  32                w      :1040            a      : 400
##           x:3656                b      : 168            l      : 400
##                                 (Other): 220            (Other): 484
##  gill.attach gill.spacing gill.sz     gill.clr    stalk.shape stalk.root
##  a: 210      c:6812       b:5612  b      :1728   e:3516      ?:2480
##  f:7914      w:1312       n:2512  p      :1492   t:4608      b:3776
##                                    w      :1202               c: 556
##                                    n      :1048               e:1120
##                                    g      : 752               r: 192
##                                    h      : 732
##                                    (Other):1170
##  stalk.sar stalk.sbr stalk.clr.abv.rng stalk.clr.bel.rng veil.type veil.clr
##  f: 552    f: 600    w      :4464      w      :4384      p:8124    n:  96
##  k:2372    k:2304    p      :1872      p      :1872                o:  96
##  s:5176    s:4936    g      : 576      g      : 576                w:7924
##  y:  24    y: 284    n      : 448      n      : 512                y:   8
##                      b      : 432      b      : 432
##                      o      : 192      o      : 192
##                      (Other): 140      (Other): 156
##  ring.num ring.type spore.prnt.clr pop      habitat
##  n:  36   e:2776    w      :2388   a: 384   d:3148
##  o:7488   f:  48    n      :1968   c: 340   g:2148
##  t: 600   l:1296    k      :1872   n: 400   l: 832
##           n:  36    h      :1632   s:1248   m: 292
##           p:3968    r      :  72   v:4040   p:1144
##                     b      :  48   y:1712   u: 368
##                     (Other): 144            w: 192
```

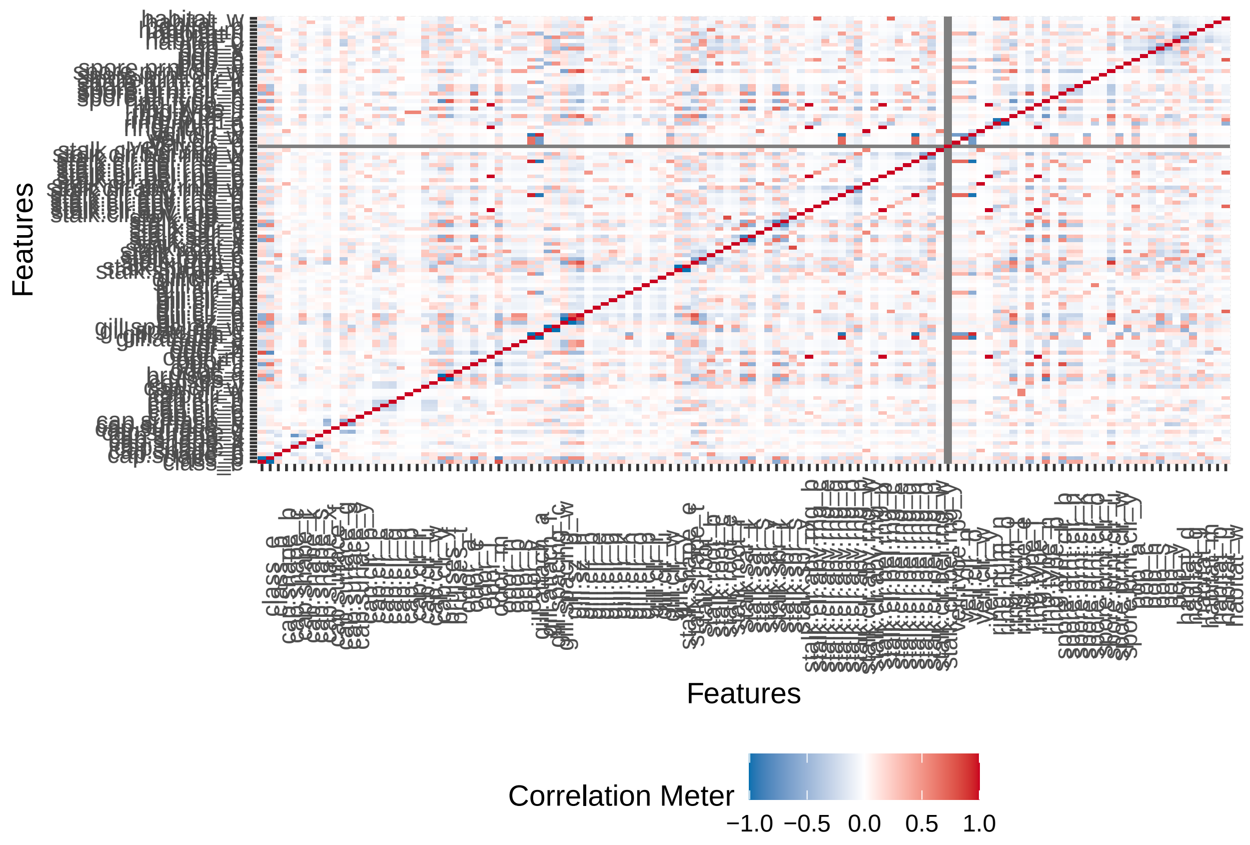
```r
plot_intro(mushroom_data)
```



```r
plot_missing(mushroom_data)
```



```r
plot_correlation(mushroom_data)
```



```r
str(mushroom_data)
```

```
## 'data.frame':    8124 obs. of  23 variables:
##  $ class           : Factor w/ 2 levels "e","p": 2 1 1 2 1 1 1 1 2 1 ...
##  $ cap.shape       : Factor w/ 6 levels "b","c","f","k",..: 6 6 1 6 6 6 1 1 6 1 ...
##  $ cap.surface     : Factor w/ 4 levels "f","g","s","y": 3 3 3 4 3 4 3 4 4 3 ...
##  $ cap.clr         : Factor w/ 10 levels "b","c","e","g",..: 5 10 9 9 4 10 9 9 9 10
##  $ bruises         : Factor w/ 2 levels "f","t": 2 2 2 2 1 2 2 2 2 2 ...
##  $ odor            : Factor w/ 9 levels "a","c","f","l",..: 7 1 4 7 6 1 1 4 7 1 ...
##  $ gill.attach     : Factor w/ 2 levels "a","f": 2 2 2 2 2 2 2 2 2 2 ...
##  $ gill.spacing    : Factor w/ 2 levels "c","w": 1 1 1 1 2 1 1 1 1 1 ...
##  $ gill.sz         : Factor w/ 2 levels "b","n": 2 1 1 2 1 1 1 1 2 1 ...
##  $ gill.clr        : Factor w/ 12 levels "b","e","g","h",..: 5 5 6 6 5 6 3 6 8 3 ...
##  $ stalk.shape      : Factor w/ 2 levels "e","t": 1 1 1 1 2 1 1 1 1 1 ...
##  $ stalk.root       : Factor w/ 5 levels "?","b","c","e",..: 4 3 3 4 4 3 3 3 4 3 ...
##  $ stalk.sar        : Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.sbr        : Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.clr.abv.rng: Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ stalk.clr.bel.rng: Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ veil.type        : Factor w/ 1 level "p": 1 1 1 1 1 1 1 1 1 1 ...
##  $ veil.clr         : Factor w/ 4 levels "n","o","w","y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ ring.num         : Factor w/ 3 levels "n","o","t": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ring.type        : Factor w/ 5 levels "e","f","l","n",..: 5 5 5 5 1 5 5 5 5 5 ...
##  $ spore.prnt.clr   : Factor w/ 9 levels "b","h","k","n",..: 3 4 4 3 4 3 3 4 3 3 ...
##  $ pop              : Factor w/ 6 levels "a","c","n","s",..: 4 3 3 4 1 3 3 4 5 4 ...
##  $ habitat          : Factor w/ 7 levels "d","g","l","m",..: 6 2 4 6 2 2 4 4 2 4 ...
```

Above a few visuals show inspecting the data is quite difficult in its current state. To allow for easier inspection of the data, it has been converted to factors. A few aspects of the data are interesting. For example vei.type only has one value and thus can be removed. Additionally stalk.root has almost one third of the data being "?". To fall in line with the rest of the column, the question marks will be converted to "u" for unknown.

```r
shroom_dat <- subset(mushroom_data, select = -c(`veil.type`))
levels(shroom_dat$`stalk.root`)[levels(shroom_dat$`stalk.root`)=="?"] <- "u" #converts 


shroom_dat$class <- as.factor(shroom_dat$class)
str(shroom_dat)
```

```
## 'data.frame':    8124 obs. of  22 variables:
##  $ class            : Factor w/ 2 levels "e","p": 2 1 1 2 1 1 1 1 2 1 ...
##  $ cap.shape        : Factor w/ 6 levels "b","c","f","k",..: 6 6 1 6 6 6 1 1 6 1 ...
##  $ cap.surface      : Factor w/ 4 levels "f","g","s","y": 3 3 3 4 3 4 3 4 4 3 ...
##  $ cap.clr          : Factor w/ 10 levels "b","c","e","g",..: 5 10 9 9 4 10 9 9 9 10
##  $ bruises          : Factor w/ 2 levels "f","t": 2 2 2 2 1 2 2 2 2 2 ...
##  $ odor             : Factor w/ 9 levels "a","c","f","l",..: 7 1 4 7 6 1 1 4 7 1 ...
##  $ gill.attach      : Factor w/ 2 levels "a","f": 2 2 2 2 2 2 2 2 2 2 ...
##  $ gill.spacing     : Factor w/ 2 levels "c","w": 1 1 1 1 2 1 1 1 1 1 ...
##  $ gill.sz          : Factor w/ 2 levels "b","n": 2 1 1 2 1 1 1 1 2 1 ...
##  $ gill.clr         : Factor w/ 12 levels "b","e","g","h",..: 5 5 6 6 5 6 3 6 8 3 ..
##  $ stalk.shape      : Factor w/ 2 levels "e","t": 1 1 1 1 2 1 1 1 1 1 ...
##  $ stalk.root       : Factor w/ 5 levels "u","b","c","e",..: 4 3 3 4 4 3 3 3 4 3 ...
##  $ stalk.sar        : Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.sbr        : Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.clr.abv.rng: Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ stalk.clr.bel.rng: Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ veil.clr         : Factor w/ 4 levels "n","o","w","y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ ring.num         : Factor w/ 3 levels "n","o","t": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ring.type        : Factor w/ 5 levels "e","f","l","n",..: 5 5 5 5 1 5 5 5 5 5 ...
##  $ spore.prnt.clr   : Factor w/ 9 levels "b","h","k","n",..: 3 4 4 3 4 3 3 4 3 3 ...
##  $ pop              : Factor w/ 6 levels "a","c","n","s",..: 4 3 3 4 1 3 3 4 5 4 ...
##  $ habitat          : Factor w/ 7 levels "d","g","l","m",..: 6 2 4 6 2 2 4 4 2 4 ...
```

SVM Model

The data is ready to be split and models are ready to be applied. The first model that will be used is the Support Vector Machine, which creates planes based off features to classify the data to either “edible” or “poisonous”. For curiosity sake, all four types of the SVM models will be applied to compare how they impact results. SVM models traditionally require numeric data in order for them to function; however, the library e1071 interprets the data as numeric based off classifying each label or category as a number.

```
idx <- createDataPartition(shroom_dat$class, p=0.7, list=FALSE) #creates index to split

shroom_train <- shroom_dat[idx,]
shroom_test <- shroom_dat[-idx,]

svm_linear <- svm(class~., data=shroom_train, type='C-classification', kernel='linear')
shroom_lin_pred <- predict(svm_linear, shroom_test)
svm_radial <- svm(class~., data=shroom_train, type='C-classification', kernel='radial')
shroom_rad_pred <- predict(svm_radial, shroom_test)
svm_poly <- svm(class~., data=shroom_train, type='C-classification', kernel='polynomial'
shroom_poly_pred <- predict(svm_poly, shroom_test)
svm_sigmoid <- svm(class~., data=shroom_train, type='C-classification', kernel='sigmoid'
shroom_sig_pred <- predict(svm_sigmoid, shroom_test)

confusionMatrix(shroom_lin_pred, shroom_test$class)$table
```

```
##           Reference
## Prediction    e    p
##          e 1262    0
##          p    0 1174
```

```
confusionMatrix(shroom_rad_pred, shroom_test$class)$table
```

```
##           Reference
## Prediction    e    p
##          e 1262    3
##          p    0 1171
```

```
confusionMatrix(shroom_poly_pred, shroom_test$class)$table
```

```
##           Reference
## Prediction    e    p
##          e 1248  148
##          p   14 1026
```

```
confusionMatrix(shroom_sig_pred, shroom_test$class)$table
```

```
##           Reference
## Prediction    e    p
##          e 1262   30
##          p    0 1144
```

From applying the SVM models above, it can be seen the linear model and the radial model both perform similarly, while the polynomial and sigmoid models struggle. Due to the complexity of the data, even though the radial model is slightly less accurate it is recommended to apply this model to future, unknown data.

Neural Network

Another model that could work well for the analysis of the mushroom dataset are neural networks. These models operate by mimicking neural networks in the brain of humans, but at a much less complicated level. One caveat of neural networks is there is an absolute requirement that the data is numerical. Because of this, two different approaches will be used: One that replaces the data with dummy variables, and Two one that converts the categorical data to numeric.

Dummy Variable test. The first neural network to be built converts the data into many columns of booleans if a label is met. For example if there is a column with three categories, it will be split into three columns. Once this is done, the data is split into training and test data. Additionally, the training columns need to be assigned for the network formula to be built.

```
shroom_features <- subset(shroom_dat, select = -class) #removing class to dummy variabl
shroom_Dummy <- dummyVars(~., data=shroom_features) #applies levels to variables for sp
shroom_Dummy <- data.frame(predict(shroom_Dummy, shroom_features)) #transforms data int
shroom_Dummy$class <- shroom_dat$class
ncol(shroom_Dummy)
```

```
## [1] 117
```

```
idx <- createDataPartition(shroom_Dummy$class, p=0.7, list=FALSE) #creates index to spl
ann_shroom_train <- shroom_Dummy[idx,]
ann_shroom_test <- shroom_Dummy[-idx,]
training_cols <- names(subset(shroom_Dummy, select = -c(class)))
```

With the training columns set, the formula for how the neural net can be built. This is necessary because neural nets don't allow for the (type ~ .) syntax to be used, so rather than type out each feature the formula can be pasted in. Once the formula is built, a network can be built, and a plot of the network is provided.

```
nn.formula <- as.formula(paste("class ~ ", paste(training_cols, collapse = " + "))) #co
network <- neuralnet(nn.formula, shroom_Dummy) #creates neural network on dummy variabl
plot(network)
```

Above the general shape of the network can be seen as having 116 input nodes, one hidden node, and two output nodes. Lastly we apply the network to the test data to see how accurate it is. To do this, the network is computed, and the the net results are compared to the test data.

```
net_predict <- compute(network, subset(ann_shroom_test, select = -c(`class`)))$net.resul
net_predict <- as.factor(c('e','p')[apply(net_predict, 1, which.max)]) #converts result

confusionMatrix(net_predict, ann_shroom_test$class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    e    p
##          e 1262    0
##          p    0 1174
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.5181
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   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 : 0.5181
##           Detection Rate : 0.5181
##     Detection Prevalence : 0.5181
##        Balanced Accuracy : 1.0000
##
##         'Positive' Class : e
##
```

Now the initial NN model with dummy variables is completed, and it can be seen the model predicted with 100% accuracy. The amount of features in this model makes for difficult interpretations of the visuals; thus next a second NN will be built converting the category data to numeric.

Numeric Variable Test. Similarly to the dummy variable NN model, there is some initial setup necessary for the model to work. The feature columns all need to be converted to numeric from factor, and then the data can be split into training and test columns. Lastly the column names need to be assigned for the formula to be built.

```r
shroom_numeric <- shroom_dat #replicates data for lewss features
shroom_numeric[, 2:22] <- sapply(shroom_numeric[,2:22],as.numeric) # converts all colum

num_idx <- createDataPartition(shroom_Dummy$class, p=0.7, list=FALSE) #creates index to
num_train <- shroom_numeric[num_idx,]
num_test <- shroom_numeric[-num_idx,]
num_train_names <- names(subset(shroom_numeric, select = -c(class)))
```

Now that the data is ready to be used, the formula and model can be built. In the MSDS680 Week-5 SVM and ANN.R file many iterations of the model can be found; however, only the most accurate/efficient one will be included in this write up. Through the iterations, it was discovered that building a layer with 3 nodes produced accurate model without sacrificing too much speed. Below the formula and network are built, and a plot is provided to show what the network looks like.

```
numeric_formula <- as.formula(paste("class ~ ", paste(num_train_names, collapse = " + ")
num_network <- neuralnet(numeric_formula, shroom_numeric, hidden= c(3)) #performs neura
plot(num_network)
```

The above network shows the advantage of building a model with less features from the perspective of visualizing. You can see the node interaction and the assigned values for separating. With this network, the test data will be predicted to compare the numeric to the dummy features.

```
num_predict <- compute(num_network, subset(num_test, select = -c(class)))$net.result
num_predict <- as.factor(c('e','p')[apply(num_predict, 1, which.max)])


confusionMatrix(num_predict, num_test$class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    e    p
##          e 1247  122
##          p   15 1052
##
##                Accuracy : 0.9438
##                  95% CI : (0.9339, 0.9526)
##     No Information Rate : 0.5181
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.887
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9881
##             Specificity : 0.8961
##          Pos Pred Value : 0.9109
##          Neg Pred Value : 0.9859
##              Prevalence : 0.5181
##          Detection Rate : 0.5119
##    Detection Prevalence : 0.5620
##       Balanced Accuracy : 0.9421
##
##        'Positive' Class : e
##
```

The numeric and dummy variable have their pros and cons. For visualizing data, the numeric network with multiple layers is much easier to interpret. From am accuracy perspective they are identical. From a efficiency perspective the dummy variable model ended up being significantly faster. Because of this, it is recommended a dummy variable NN would be more practical to use.

Conclusion

This analysis shows the strengths and weaknesses of both SVM, along with how incredibly accurate they both have the potential to be. The next steps to decrease possibility of overfitting could be to reduce the features based off of correlation in order to see if there are some features unnecessary, or reduce the amount of training data provided to the model to provide a better split of unknown data. This concludes the analysis of SVM and ANN models in R.

References

Günther, F., and Fritsch, S. (2010). neuralnet: Training of neural networks. The R journal, 2(1), 30-38.

Yu-Wei, C. (2015). Machine Learning with R Cookbook. Packt Publishing.

Lantz, B. (2015). Machine Learning with R: Expert Techniques for Predictive Modeling to Solve All Your Data Analysis Problems: Vol. Second edition. Packt Publishing.