Support Vector Machines and Artificial Neural Networks

Objective: to build a classifier for edible and poisonous mushrooms.

Data set: Mushrooms dataset http://archive.ics.uci.edu/ml/datasets/Mushroom

First, I prepared the environment, set the working directory and loaded the libraries using the following:

```
> ### SVMs and ANNs
>
> rm(list=ls()) #Clear the environment
> setwd("YOUR_PATH") #Set working directory for the assignment
> getwd() #Check working directory
[1] "YOUR_PATH"
>
> #Use SVS and ANNs to build an edible/poisonois mushrooms classifier
> #Mushroom Dataset from http://archive.ics.uci.edu/ml/datasets/Mushroom
>
> #Load packages
> library(caret)
Loading required package: lattice
Loading required package: ggplot2
> library(e1071)
> library(nnet)
```

Next, I loaded the mushroom dataset from the previously downloaded data file (http://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/) using the following code:

```
> #Load data
> shroom <- read.table("agaricus-lepiota.data", header = FALSE, sep = ",")
```

To make sure that the data was loaded correctly, I previewed the data and looked at the internal structure of the data frame:

```
> #Check
> View(shroom) #Preview data
> str(shroom) #Check internal structure of the data frame
'data.frame':      8124 obs. of  23 variables:
 $ V1 : Factor w/ 2 levels "e","p": 2 1 1 2 1 1 1 1 2 1 ...
```

```
 $ V2 : Factor w/ 6 levels "b","c","f","k",..: 6 6 1 6 6 6 1 1 6 1 ...
 $ V3 : Factor w/ 4 levels "f","g","s","y": 3 3 3 4 3 4 3 4 4 3 ...
 $ V4 : Factor w/ 10 levels "b","c","e","g",..: 5 10 9 9 4 10 9 9 9 10 ...
 $ V5 : Factor w/ 2 levels "f","t": 2 2 2 2 1 2 2 2 2 2 ...
 $ V6 : Factor w/ 9 levels "a","c","f","l",..: 7 1 4 7 6 1 1 4 7 1 ...
 $ V7 : Factor w/ 2 levels "a","f": 2 2 2 2 2 2 2 2 2 2 ...
 $ V8 : Factor w/ 2 levels "c","w": 1 1 1 1 2 1 1 1 1 1 ...
 $ V9 : Factor w/ 2 levels "b","n": 2 1 1 2 1 1 1 1 2 1 ...
 $ V10: Factor w/ 12 levels "b","e","g","h",..: 5 5 6 6 5 6 3 6 8 3 ...
 $ V11: Factor w/ 2 levels "e","t": 1 1 1 1 2 1 1 1 1 1 ...
 $ V12: Factor w/ 5 levels "?","b","c","e",..: 4 3 3 4 4 3 3 3 4 3 ...
 $ V13: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
 $ V14: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
 $ V15: Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...
 $ V16: Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...
 $ V17: Factor w/ 1 level "p": 1 1 1 1 1 1 1 1 1 1 ...
 $ V18: Factor w/ 4 levels "n","o","w","y": 3 3 3 3 3 3 3 3 3 3 ...
 $ V19: Factor w/ 3 levels "n","o","t": 2 2 2 2 2 2 2 2 2 2 ...
 $ V20: Factor w/ 5 levels "e","f","l","n",..: 5 5 5 5 1 5 5 5 5 5 ...
 $ V21: Factor w/ 9 levels "b","h","k","n",..: 3 4 4 3 4 3 3 4 3 3 ...
 $ V22: Factor w/ 6 levels "a","c","n","s",..: 4 3 3 4 1 3 3 4 5 4 ...
 $ V23: Factor w/ 7 levels "d","g","l","m",..: 6 2 4 6 2 2 4 4 2 4 ...
```

The data was loaded correctly.

The data set contains 8124 observations of 23 categorical variables that were loaded as factors. First, for convenience I added column names based on the meta data contained in the description of the dataset:

```
> #Change column names
> names(shroom) <- c("class", "cap_shape", "cap_surface", "cap_color", "bruises", "odor", "gill_a
ttachment", "gill_spacing", "gill_size", "gill_color", "stalk_shape", "stalk_root", "stalk_surfac
e_above_ring", "stalk_surface_below_ring", "stalk_color_above_ring", "stalk_color_below_ring", "v
eil_type", "veil_color", "ring_number", "ring_type", "spore_print_color", "population", "habitat"
)
> str(shroom) #Check results
'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_color                : 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_attachment          : 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_size                : Factor w/ 2 levels "b","n": 2 1 1 2 1 1 1 1 2 1 ...
 $ gill_color               : 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_surface_above_ring: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
 $ stalk_surface_below_ring: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
 $ stalk_color_above_ring  : Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 ...
```

```
$ stalk_color_below_ring  : Factor w/ 9 levels "b","c","e","g",..: 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 ...
$ veil_color              : Factor w/ 4 levels "n","o","w","y": 3 3 3 3 3 3 3 3 3 ...
$ ring_number             : Factor w/ 3 levels "n","o","t": 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 ...
$ spore_print_color       : Factor w/ 9 levels "b","h","k","n",..: 3 4 4 3 4 3 3 4 3 3 ...
$ population              : 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 ...
```

As the above output shows, the column names were changed successfully. Next, I looked at the summary statistics for all variables in the dataset:

```
> #Summary statistics for all variables in the dataset
> summary(shroom)
 class     cap_shape cap_surface   cap_color    bruises       odor      gill_attachment gill_spacing gill_size
 e:4208   b: 452    f:2320      n      :2284   f:4748   n      :3528   a: 210          c:6812       b:5612
 p:3916   c:   4    g:   4      g      :1840   t:3376   f      :2160   f:7914          w:1312       n:2512
          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_color   stalk_shape stalk_root stalk_surface_above_ring stalk_surface_below_ring stalk_color_above_ring
 b      :1728   e:3516      ?:2480     f: 552                   f: 600                   w      :4464
 p      :1492   t:4608      b:3776     k:2372                   k:2304                   p      :1872
 w      :1202               c: 556     s:5176                   s:4936                   g      : 576
 n      :1048               e:1120     y:  24                   y: 284                   n      : 448
 g      : 752               r: 192                                                       b      : 432
 h      : 732                                                                           o      : 192
 (Other):1170                                                                           (Other): 140
 stalk_color_below_ring veil_type veil_color ring_number ring_type spore_print_color population habitat
 w      :4384           p:8124    n:  96     n:  36      e:2776    w      :2388       a: 384     d:3148
 p      :1872                     o:  96     o:7488      f:  48    n      :1968       c: 340     g:2148
 g      : 576                     w:7924     t: 600      l:1296    k      :1872       n: 400     l: 832
 n      : 512                     y:   8                 n:  36    h      :1632       s:1248     m: 292
 b      : 432                                            p:3968    r      :  72       v:4040     p:1144
 o      : 192                                                      b      :  48       y:1712     u: 368
 (Other): 156                                                      (Other): 144                  w: 192
>
```

I noticed that veil_type contains only one level – p, and it will not help with classification task. So, I decided to delete this column:

```
> #Delete veil-type column
> shroom <- shroom[,-17]
```

I also noticed, that stalk_root column contains 2480 observation with level “?”, that according to the dataset description, was used to denote missing values. None of the other columns had missing values. I decided to add a new factor level – m (for missing) in the stalk_rook column and substitute level”?” with “m”:

```
> #Replace ? with level m (missing) in stalk_root variable
> #add m to the factor levels
> levels(shroom$stalk_root) <- c(levels(shroom$stalk_root), 'm')
> #Change level "?" to "m"
> shroom$stalk_root[shroom$stalk_root == "?"] <- "m"
> summary(shroom$stalk_root) #check results
   ?    b    c    e    r    m
   0 3776  556 1120  192 2480
```

Since all variables in the dataset are categorical, I used one-hot full encoding to prepare data for modeling. I used dummyVar function on all independent variables in the dataset:

```
> ####One-hot full rank encoding
>
> shroom_var <- shroom[, -1] #df with independent variables only
> shroom_fr <- dummyVars(~ ., data = shroom_var, fullRank = TRUE)
> shroom_varfr <- predict(shroom_fr, shroom)
```

Sample output:

```
> #Check results
> str(shroom_varfr)
 num [1:8124, 1:96] 0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:8124] "1" "2" "3" "4" ...
  ..$ : chr [1:96] "cap_shape.c" "cap_shape.f" "cap_shape.k" "cap_shape.s" ...
> summary(shroom_varfr)
  cap_shape.c        cap_shape.f      cap_shape.k      cap_shape.s       cap_shape.x    cap_surface.g
 Min.   :0.0000000   Min.   :0.000   Min.   :0.0000   Min.   :0.000000   Min.   :0.00   Min.   :0.0000000
 1st Qu.:0.0000000   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.000000   1st Qu.:0.00   1st Qu.:0.0000000
 Median :0.0000000   Median :0.000   Median :0.0000   Median :0.000000   Median :0.00   Median :0.0000000
 Mean   :0.0004924   Mean   :0.388   Mean   :0.1019   Mean   :0.003939   Mean   :0.45   Mean   :0.0004924
 3rd Qu.:0.0000000   3rd Qu.:1.000   3rd Qu.:0.0000   3rd Qu.:0.000000   3rd Qu.:1.00   3rd Qu.:0.0000000
 Max.   :1.0000000   Max.   :1.000   Max.   :1.0000   Max.   :1.000000   Max.   :1.00   Max.   :1.0000000
 cap_surface.s     cap_surface.y     cap_color.c        cap_color.e      cap_color.g       cap_color.n
 Min.   :0.0000   Min.   :0.0000   Min.   :0.000000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :0.0000   Median :0.0000   Median :0.000000   Median :0.0000   Median :0.0000   Median :0.0000
 Mean   :0.3146   Mean   :0.3993   Mean   :0.005416   Mean   :0.1846   Mean   :0.2265   Mean   :0.2811
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.000000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:1.0000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.000000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
  cap_color.p       cap_color.r        cap_color.u        cap_color.w      cap_color.y       bruises.t
 Min.   :0.00000   Min.   :0.000000   Min.   :0.000000   Min.   :0.000   Min.   :0.000   Min.   :0.0000
 1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.0000
 Median :0.00000   Median :0.000000   Median :0.000000   Median :0.000   Median :0.000   Median :0.0000
 Mean   :0.01773   Mean   :0.001969   Mean   :0.001969   Mean   :0.128   Mean   :0.132   Mean   :0.4156
 3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:1.0000
```

As the sample output shows, the resulting large matric contained 8124 records of 96 variables (779904 elements).

Then, to reconstruct the full data, I added the class variable back to the data frame, added the column name and made sure that class variable is a factor datatype:

```
> #Add class variable back to the data frame
>
```

```
> shroom_fullfr <- as.data.frame(cbind(shroom$class, shroom_varfr)) #add the first column
> colnames(shroom_fullfr)[1] <- "class" #add column name
> shroom_fullfr$class <- as.factor(shroom_fullfr$class)
```

I used summary() and str() commands to check the results. Sample output is below:

```
> summary(shroom_fullfr) #Check results
 class      cap_shape.c         cap_shape.f      cap_shape.k      cap_shape.s         cap_shape.x
 1:4208   Min.   :0.0000000   Min.   :0.000   Min.   :0.0000   Min.   :0.000000   Min.   :0.00
 2:3916   1st Qu.:0.0000000   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.000000   1st Qu.:0.00
          Median :0.0000000   Median :0.000   Median :0.0000   Median :0.000000   Median :0.00
          Mean   :0.0004924   Mean   :0.388   Mean   :0.1019   Mean   :0.003939   Mean   :0.45
          3rd Qu.:0.0000000   3rd Qu.:1.000   3rd Qu.:0.0000   3rd Qu.:0.000000   3rd Qu.:1.00
          Max.   :1.0000000   Max.   :1.000   Max.   :1.0000   Max.   :1.000000   Max.   :1.00
 cap_surface.g       cap_surface.s    cap_surface.y     cap_color.c         cap_color.e      cap_color.g
 Min.   :0.0000000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :0.0000000   Median :0.0000   Median :0.0000   Median :0.000000   Median :0.0000   Median :0.0000
 Mean   :0.0004924   Mean   :0.3146   Mean   :0.3993   Mean   :0.005416   Mean   :0.1846   Mean   :0.2265
 3rd Qu.:0.0000000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.000000   3rd Qu.:0.0000   3rd Qu.:0.0000
 Max.   :1.0000000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000000   Max.   :1.0000   Max.   :1.0000
  cap_color.n       cap_color.p        cap_color.r        cap_color.u         cap_color.w      cap_color.y
 Min.   :0.0000   Min.   :0.00000   Min.   :0.000000   Min.   :0.000000   Min.   :0.000   Min.   :0.000
 1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000   1st Qu.:0.000
 Median :0.0000   Median :0.00000   Median :0.000000   Median :0.000000   Median :0.000   Median :0.000
```

```
> str(shroom_fullfr) #Look at the resulting df
'data.frame':   8124 obs. of  97 variables:
 $ class                    : Factor w/ 2 levels "1","2": 2 1 1 2 1 1 1 1 2 1 ...
 $ cap_shape.c              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ cap_shape.f              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ cap_shape.k              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ cap_shape.s              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ cap_shape.x              : num  1 1 0 1 1 1 0 0 1 0 ...
 $ cap_surface.g            : num  0 0 0 0 0 0 0 0 0 0 ...
 $ cap_surface.s            : num  1 1 1 0 1 0 1 0 0 1 ...
 $ cap_surface.y            : num  0 0 0 1 0 1 0 1 1 0 ...
 $ cap_color.c              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ cap_color.e              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ cap_color.g              : num  0 0 0 0 1 0 0 0 0 0 ...
 $ cap_color.n              : num  1 0 0 0 0 0 0 0 0 0 ...
 $ cap_color.p              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ cap_color.r              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ cap_color.u              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ cap_color.w              : num  0 0 1 1 0 0 1 1 1 0 ...
 $ cap_color.y              : num  0 1 0 0 0 1 0 0 0 1 ...
 $ bruises.t                : num  1 1 1 1 0 1 1 1 1 1 ...
 $ odor.c                   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ odor.f                   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ odor.l                   : num  0 0 1 0 0 0 0 1 0 0 ...
 $ odor.m                   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ odor.n                   : num  0 0 0 0 1 0 0 0 0 0 ...
 $ odor.p                   : num  1 0 0 1 0 0 0 0 1 0 ...
 $ odor.s                   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ odor.y                   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ gill_attachment.f        : num  1 1 1 1 1 1 1 1 1 1 ...
 $ gill_spacing.w           : num  0 0 0 0 1 0 0 0 0 0 ...
 $ gill_size.n              : num  1 0 0 1 0 0 0 0 1 0 ...
 $ gill_color.e             : num  0 0 0 0 0 0 0 0 0 0
```

After the above step I had preprocessed full initial data set ready to be split into training and testing data:

```
> #Split the dataset into training and testing data
> #(80% training, 20% testing)
>
> set.seed(15)
> ind <- sample.int(n=nrow(shroom_fullfr), size=floor(0.8*nrow(shroom_fullfr)), replace = FALSE)
> shroom_train <-  shroom_fullfr[ind, ] #training set
> shroom_test <- shroom_fullfr[ -ind, ] #testing set
```

As a result, the training dataset contains 6499 observation of 97 variables and the testing set contains 1625 observations of 97 variables:

```
> #Check resulting datasets
> dim(shroom_train)
[1] 6499   97
> dim(shroom_test)
[1] 1625   97
```

I also checked the proportions between the edible and poisonous mushrooms in both subsets, which turned out to be approximately equal:

```
> #Compare proportions between edible and poisonous mashrooms in testing and training data
> prop.table(table(shroom_train$class))

        1         2
0.5196184 0.4803816
> prop.table(table(shroom_test$class))

        1         2
0.5113846 0.4886154
```

In the following steps I used the same training set to construct an SVM and ANN classifiers and tested them on the same testing data set.

First, I used e1071 package to create a Support Vector Machine (SVM) using the following command:

```
> ####SVM
>
> svm_model1 <- svm(class ~ ., data = shroom_train, kernel = "radial", cost = 1, gamma = 1/ncol(s
hroom_train))
> summary(svm_model1)

Call:
svm(formula = class ~ ., data = shroom_train, kernel = "radial", cost = 1, gamma = 1/ncol(shroom_
train))
```

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.01030928

Number of Support Vectors:  1200

 ( 471 729 )


Number of Classes:  2

Levels:
 1 2
```

As the summary above shows, an SVM with a default radial kernel used 1200 support vector to construct the model. The model applied to the testing set yielded the following results:

```
> #Using the model for predictions
> svm_prediction1 <- predict(svm_model1, shroom_test[ , !names(shroom_test) %in% c("class")])
> #generate classification table
> svm_table1 <- table(svm_prediction1, shroom_test$class)
> svm_table1

svm_prediction1   1   2
              1 831   0
              2   0 794
```

It means that the model was able to classify the mushrooms in the dataset with the 100% accuracy:

```
> #Evaluate results
> classAgreement(svm_table1)
$`diag`
[1] 1

$kappa
[1] 1

$rand
[1] 1

$crand
[1] 1

> confusionMatrix(svm_table1)
Confusion Matrix and Statistics
```

```
svm_prediction1   1   2
              1 831   0
              2   0 794

               Accuracy : 1
                 95% CI : (0.9977, 1)
    No Information Rate : 0.5114
    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 : 0.5114
         Detection Rate : 0.5114
   Detection Prevalence : 0.5114
      Balanced Accuracy : 1.0000

       'Positive' Class : 1
```

Frankly, I was very suspicious of the 100% accuracy and decided to try to manipulate the SVM parameters (chose different kernel types) to try to "break" the model.

I created a model with the linear kernel:

```
> #Try out other kernel types
> #linear
> svm_model2 <- svm(class ~ ., data = shroom_train, kernel = "linear", cost = 1, gamma = 1/ncol(s
hroom_train))
> summary(svm_model2)

Call:
svm(formula = class ~ ., data = shroom_train, kernel = "linear", cost = 1, gamma = 1/ncol(shroom_
train))


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.01030928

Number of Support Vectors:  225

 ( 110 115 )
```

```
Number of Classes:  2

Levels:
 1 2
```

It required considerably fewer support vectors – 225. I used the linear kernel model to generate predictions:

```
> #Using the model for predictions
> shroom_testnoclass <- shroom_test[-1] #Test dataset without the class column
> svm_prediction2 <- predict(svm_model1, shroom_testnoclass)
>
> #generate classification table
> svm_table2 <- table(svm_prediction2, shroom_test$class)
> svm_table2

svm_prediction2   1   2
              1 831   0
              2   0 794
```

Again, the model was able to classify the test observations with 100% accuracy:

```
> #Evaluate results
> classAgreement(svm_table2)
$`diag`
[1] 1

$kappa
[1] 1

$rand
[1] 1

$crand
[1] 1

> confusionMatrix(svm_table2)
Confusion Matrix and Statistics


svm_prediction2   1   2
              1 831   0
              2   0 794

               Accuracy : 1
                 95% CI : (0.9977, 1)
    No Information Rate : 0.5114
    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 : 0.5114
         Detection Rate : 0.5114
   Detection Prevalence : 0.5114
      Balanced Accuracy : 1.0000

       'Positive' Class : 1
```

I repeated the same procedure for the polynomial kernel SVM

```
#polynomial
> svm_model3 <- svm(class ~ ., data = shroom_train, kernel = "polynomial", cost = 1, gamma = 1/nc
ol(shroom_train))
> summary(svm_model3)

Call:
svm(formula = class ~ ., data = shroom_train, kernel = "polynomial", cost = 1, gamma = 1/ncol(shr
oom_train))


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  1
     degree:  3
      gamma:  0.01030928
     coef.0:  0

Number of Support Vectors:  1199

 ( 487 712 )


Number of Classes:  2

Levels:
 1 2



> #Using the model for predictions
> svm_prediction3 <- predict(svm_model3, shroom_testnoclass)
>
> #generate classification table
> svm_table3 <- table(svm_prediction3, shroom_test$class)
> svm_table3

svm_prediction3   1   2
              1 831   0
              2   0 794
```

And again received 100% classification accuracy:

```
> #Evaluate results
> classAgreement(svm_table3)
$`diag`
[1] 1

$kappa
[1] 1

$rand
[1] 1

$crand
[1] 1

> confusionMatrix(svm_table3)
Confusion Matrix and Statistics


svm_prediction3   1   2
              1 831   0
              2   0 794

               Accuracy : 1
                 95% CI : (0.9977, 1)
    No Information Rate : 0.5114
    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 : 0.5114
         Detection Rate : 0.5114
   Detection Prevalence : 0.5114
      Balanced Accuracy : 1.0000

       'Positive' Class : 1
```

Finally, I constructed a model with the sigmoid kernel:

```
> #sigmoid
> svm_model4 <- svm(class ~ ., data = shroom_train, kernel = "sigmoid", cost = 1, gamma = 1/ncol(
shroom_train))
> summary(svm_model4)

Call:
svm(formula = class ~ ., data = shroom_train, kernel = "sigmoid", cost = 1, gamma = 1/ncol(shroom
_train))
```

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  sigmoid
       cost:  1
      gamma:  0.01030928
     coef.0:  0

Number of Support Vectors:  228

 ( 113 115 )


Number of Classes:  2

Levels:
 1 2
```

And used it to generate predictions:

```
> #Using the model for predictions
> svm_prediction4 <- predict(svm_model4, shroom_testnoclass)
>
> #generate classification table
> svm_table4 <- table(svm_prediction4, shroom_test$class)
> svm_table4

svm_prediction4   1   2
              1 827  10
              2   4 784
```

This time 14 observations were misclassified, which still resulted in 99.14% accuracy:

```
> #Evaluate results
> classAgreement(svm_table4)
$`diag`
[1] 0.9913846

$kappa
[1] 0.9827574

$rand
[1] 0.9829072

$crand
[1] 0.9658143

> confusionMatrix(svm_table4)
Confusion Matrix and Statistics


svm_prediction4   1   2
              1 827  10
              2   4 784
```

```
               Accuracy : 0.9914
                 95% CI : (0.9856, 0.9953)
    No Information Rate : 0.5114
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9828
 Mcnemar's Test P-Value : 0.1814

            Sensitivity : 0.9952
            Specificity : 0.9874
         Pos Pred Value : 0.9881
         Neg Pred Value : 0.9949
             Prevalence : 0.5114
         Detection Rate : 0.5089
   Detection Prevalence : 0.5151
      Balanced Accuracy : 0.9913

       'Positive' Class : 1
```

Next, I used the nnet package to construct a neural network.

I used the following code to fit the model with :

```
> #train the neural network with nnet
> nn_model1 <- nnet(class ~  ., data = shroom_train, size = 2, rang = 0.1, decay = 5e-4, maxit =
500 )
```

Sample output:

```
iter 460 value 0.377965
iter 470 value 0.377964
iter 480 value 0.377963
final  value 0.377963
converged
> summary(nn_model1)
a 96-2-1 network with 197 weights
options were - entropy fitting  decay=5e-04
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1  i9->h1 i10->h1 i11->h1 i12->h1 i13->h1
  -0.36    0.22   -0.18   -0.16   -0.41   -0.26    0.46    0.21    0.21   -0.73    0.03   -0.11   -0.40    0.14
i14->h1 i15->h1 i16->h1 i17->h1 i18->h1 i19->h1 i20->h1 i21->h1 i22->h1 i23->h1 i24->h1 i25->h1 i26->h1 i27->h1
  -0.37   -0.31    0.02   -0.35   -0.08    2.79    2.01   -0.83    0.04   -2.46    2.11    1.01    1.03    0.17
i28->h1 i29->h1 i30->h1 i31->h1 i32->h1 i33->h1 i34->h1 i35->h1 i36->h1 i37->h1 i38->h1 i39->h1 i40->h1 i41->h1
```

As the sample output above shows, it took less than 500 iterations for the model to converge and I resulted in 96-2-1 network with 197 weights. I used the model to generate predictions:

```
> nn_prediction1 <- predict (nn_model1, shroom_testnoclass, type="class")
> nn_table1 <- table(shroom_test$class, nn_prediction1)
> nn_table1
```

```
   nn_prediction1
      1   2
  1 831   0
  2   0 794
```

The prediction results on the test set were 100% accurate:

```
> confusionMatrix(nn_table1)
Confusion Matrix and Statistics

   nn_prediction1
      1   2
  1 831   0
  2   0 794

               Accuracy : 1
                 95% CI : (0.9977, 1)
    No Information Rate : 0.5114
    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 : 0.5114
         Detection Rate : 0.5114
   Detection Prevalence : 0.5114
      Balanced Accuracy : 1.0000

       'Positive' Class : 1
```

Again, highly desirable, but very suspicious accuracy prompted me to try to change the model parameters in way decreasing it accuracy.

First, I tried to severely restrict the number of iterations allowed with maxit parameter set to 50:

```
> ###ANN model adjustments
> #model2 - restrict number of iterations
> nn_model2 <- nnet(class ~  ., data = shroom_train, size = 2, rang = 0.1, de
cay = 5e-4, maxit = 50 )
# weights:  197
initial  value 4511.688326
iter  10 value 602.729446
iter  20 value 241.843884
iter  30 value 218.339481
iter  40 value 215.896634
```

```
iter  50 value 213.828157
final  value 213.828157
stopped after 50 iterations
> summary(nn_model2)
a 96-2-1 network with 197 weights
options were - entropy fitting  decay=5e-04
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->
```

(remainder of output omitted)

I used the model to generate predictions:

```
> nn_prediction2 <- predict (nn_model2, shroom_testnoclass, type="class")
> nn_table2 <- table(shroom_test$class, nn_prediction2)
> nn_table2
   nn_prediction2
      1   2
  1 809  22
  2   0 794
```

This time 22 mushroom samples were misclassified, which resulted in 98.65% accuracy of the model:

```
> confusionMatrix(nn_table2)
Confusion Matrix and Statistics

   nn_prediction2
      1   2
  1 809  22
  2   0 794

               Accuracy : 0.9865
                 95% CI : (0.9796, 0.9915)
    No Information Rate : 0.5022
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9729
 Mcnemar's Test P-Value : 7.562e-06

            Sensitivity : 1.0000
            Specificity : 0.9730
         Pos Pred Value : 0.9735
         Neg Pred Value : 1.0000
             Prevalence : 0.4978
         Detection Rate : 0.4978
   Detection Prevalence : 0.5114
      Balanced Accuracy : 0.9865

       'Positive' Class : 1
```

I repeated the steps with the default for nnet function number f iterations – 100:

```
> #model 3 - default number of iterations = 100
> nn_model3 <- nnet(class ~  ., data = shroom_train, size = 2, rang = 0.1, de
cay = 5e-4, maxit = 100 )
# weights:  197
initial  value 4508.098045
iter  10 value 3044.438339
iter  20 value 869.627125
iter  30 value 251.861822
iter  40 value 130.139043
iter  50 value 9.798670
iter  60 value 6.688230
iter  70 value 4.739829
iter  80 value 3.072347
iter  90 value 2.078984
iter 100 value 1.609315
final  value 1.609315
stopped after 100 iterations
> summary(nn_model3)
a 96-2-1 network with 197 weights
options were - entropy fitting  decay=5e-04
  b->h1  i1->h1  i2->h1  i3->h1
```

(the remainder of output is omitted)

Even with 100 iterations the model was able to correctly classify all test samples:

```
> nn_prediction3 <- predict (nn_model3, shroom_testnoclass, type="class")
> nn_table3 <- table(shroom_test$class, nn_prediction1)
> nn_table3
   nn_prediction1
      1   2
  1 831   0
  2   0 794
>
> confusionMatrix(nn_table3)
Confusion Matrix and Statistics

   nn_prediction1
      1   2
  1 831   0
  2   0 794

               Accuracy : 1
                 95% CI : (0.9977, 1)
    No Information Rate : 0.5114
    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 : 0.5114
         Detection Rate : 0.5114
   Detection Prevalence : 0.5114
```

```
      Balanced Accuracy : 1.0000

       'Positive' Class : 1
```

Next, I tried to decrease the size of the network:

```
> #model4 - size =1
> nn_model4 <- nnet(class ~  ., data = shroom_train, size = 1, rang = 0.1, decay = 5e-4, maxit =
500 )
# weights:  99
initial  value 4521.232237
iter  10 value 2389.979281
iter  20 value 1467.400423
```

(part of the output is omitted)

```
iter 360 value 0.484688
final  value 0.484687
converged
> summary(nn_model4)
a 96-1-1 network with 99 weights
options were - entropy fitting  decay=5e-04
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1  i9->h1 i10->h1 i11->h1 i
12->h1 i13->h1
  -0.37    0.28   -0.22   -0
```

(part of the output is omitted)

Nevertheless, the simpler network with 1 hidden layer was able to correctly classify all test samples.

```
> nn_prediction4 <- predict (nn_model4, shroom_testnoclass, type="class")
> nn_table4 <- table(shroom_test$class, nn_prediction4)
> nn_table4
   nn_prediction4
      1   2
  1 831   0
  2   0 794
```

Next, I intentionally tried to overcomplicate the model by including 5 middle layers:

```
> #model5 - size=5
> nn_model5 <- nnet(class ~  ., data = shroom_train, size = 5, rang = 0.1, decay = 5e-4, maxit =
500 )
# weights:  491
initial  value 4504.913184
iter  10 value 657.572283
```

(Part of the output is omitted)

```
iter 500 value 0.244264
final  value 0.244264
stopped after 500 iterations
> summary(nn_model5)
a 96-5-1 network with 491 weights
options were - entropy fitting  decay=5e-04
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1
```

(Part of the output is omitted)

It affected the running time for the model, but the prediction results remained 100% accurate:

```
> nn_prediction5 <- predict (nn_model5, shroom_testnoclass, type="class")
> nn_table5 <- table(shroom_test$class, nn_prediction5)
> nn_table5
   nn_prediction5
      1   2
  1 831   0
  2   0 794
```

Next, I increased the decay parameter, or incremental step used by the model:

```
> #model6 change decay
> nn_model6 <- nnet(class ~  ., data = shroom_train, size = 2, rang = 0.1, decay = 5e-2, maxit =
500 )
# weights:  197
initial  value 4508.425475
iter  10 value 710.218183
```

(part of the output is omitted.)

```
iter 220 value 15.777130
final  value 15.777130
converged
> summary(nn_model6)
a 96-2-1 network with 197 weights
options were - entropy fitting  decay=0.05
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1
```

(Part of the output is omitted.)


```
iter 220 value 15.777130
final  value 15.777130
converged
> summary(nn_model6)
a 96-2-1 network with 197 weights
options were - entropy fitting  decay=0.05
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1
```

Bigger steps obviously decreased the number of iterations before converging and they did have influence on the model precision. This time, two samples were misclassified:

```
> nn_prediction6 <- predict (nn_model6, shroom_testnoclass, type="class")
> nn_table6 <- table(shroom_test$class, nn_prediction6)
> nn_table6
   nn_prediction6
      1   2
  1 831   0
  2   0 794
```

However, considering the size of the test dataset it still resulted in the 99.77% accuracy rate:

```
> confusionMatrix(nn_table6)
Confusion Matrix and Statistics

   nn_prediction6
      1   2
  1 831   0
  2   0 794

               Accuracy : 1
                 95% CI : (0.9977, 1)
    No Information Rate : 0.5114
    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 : 0.5114
         Detection Rate : 0.5114
   Detection Prevalence : 0.5114
      Balanced Accuracy : 1.0000

       'Positive' Class : 1
```

Next, I simultaneously increased decay and limited maximum number of iterations:

```
> #model7 change decay and decrease max number of iterations
> nn_model7 <- nnet(class ~  ., data = shroom_train, size = 2, rang = 0.1, decay = 5e-2, maxit =
50 )
# weights:  197
initial  value 4498.593920
iter  10 value 789.630516
iter  20 value 411.695008
```

```
iter  30 value 308.499061
iter  40 value 233.410026
iter  50 value 182.115040
final  value 182.115040
stopped after 50 iterations
> summary(nn_model7)
a 96-2-1 network with 197 weights
options were - entropy fitting  decay=0.05
  b->h1  i1->h1  i2->h1  i3->
```

(Part of the output is omitted.)

Surprisingly, it still resulted in a model with 100% prediction accuracy correctly classifying all test observations:

```
> nn_prediction7 <- predict (nn_model7, shroom_testnoclass, type="class")
> nn_table7 <- table(shroom_test$class, nn_prediction7)
> nn_table7
   nn_prediction7
      1   2
  1 831   0
  2   0 794
```

I continued experimenting with the model and simultaneously changed decay, maximum number of iterations and the size of the model:

```
> #model8 change decay, decrease max number of iterations and size
> nn_model8 <- nnet(class ~  ., data = shroom_train, size = 1, rang = 0.1, decay = 5e-2, maxit =
40 )
# weights:  99
initial  value 4520.011015
iter  10 value 3453.630382
iter  20 value 2418.671629
iter  30 value 1688.795022
iter  40 value 1407.054056
final  value 1407.054056
stopped after 40 iterations
> summary(nn_model8)
a 96-1-1 network with 99 weights
options were - entropy fitting  decay=0.05
  b->h1  i1->h1  i2->h
```

This time, the model demonstrated considerably decreased predictive power as is misclassified 96 test observations:

```
> nn_prediction8 <- predict (nn_model8, shroom_testnoclass, type="class")
> nn_table8 <- table(shroom_test$class, nn_prediction8)
> nn_table8
   nn_prediction8
```

```
     1   2
  1 831   0
  2  96 698
```

It means 94.09% accuracy as demonstrated by the confusion matrix below:

```
> confusionMatrix(nn_table8)
Confusion Matrix and Statistics

   nn_prediction8
      1   2
  1 831   0
  2  96 698

               Accuracy : 0.9409
                 95% CI : (0.9283, 0.9519)
    No Information Rate : 0.5705
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8815
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8964
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.8791
             Prevalence : 0.5705
         Detection Rate : 0.5114
   Detection Prevalence : 0.5114
      Balanced Accuracy : 0.9482

       'Positive' Class : 1
```

For the next model I change the rang parameter, or initial weights used by the model:

```
> #model9 change rang
> nn_model9 <- nnet(class ~  ., data = shroom_train, size = 2, rang = 0.5, decay = 5e-4, maxit =
500 )
# weights:  197
initial  value 4571.822461
iter  10 value 867.783060
```

(Part of the output is omitted.)

```
final  value 0.292983
converged
> summary(nn_model9)
a 96-2-1 network with 197 weights
options were - entropy fitting  decay=5e-04
  b->h1  i1->h1  i2->h1  i3->h1
```

(Part of the output is omitted.)

This step did not have any effect of the classification precision when model was applied to the testing data set:

```
> nn_prediction9 <- predict (nn_model9, shroom_testnoclass, type="class")
> nn_table9 <- table(shroom_test$class, nn_prediction9)
> nn_table9
   nn_prediction9
      1   2
  1 831   0
  2   0 794
```

Finally, I decided to try a model in which four parameters differ from my initial model 1 – decreased size, 0.5 initial weights, increased decay and limited maximum number of iterations.

```
> #model10 change size,  rang, decay and maxit at the same time
> nn_model10 <- nnet(class ~  ., data = shroom_train, size = 1, rang = 0.5, decay = 5e-2, maxit =
50 )
# weights:  99
initial  value 4500.944601
iter  10 value 529.776080
iter  20 value 391.673440
iter  30 value 351.372461
iter  40 value 332.278913
iter  50 value 105.208174
final  value 105.208174
stopped after 50 iterations
> summary(nn_model10)
a 96-1-1 network with 99 weights
options were - entropy fitting  decay=0.05
  b->h1  i1->h1  i2->h1  i3->h1
```

As the output below shows, the model incorrectly classified 11 samples:

```
> nn_prediction10 <- predict (nn_model10, shroom_testnoclass, type="class")
> nn_table10 <- table(shroom_test$class, nn_prediction10)
> nn_table10
   nn_prediction10
      1   2
  1 826   5
  2   6 788
```

Which still resulted in 99.32% accuracy given the size of the test sample:

```
> confusionMatrix(nn_table10)
Confusion Matrix and Statistics

   nn_prediction10
      1   2
```

```
  1 826   5
  2   6 788

               Accuracy : 0.9932
                 95% CI : (0.9879, 0.9966)
    No Information Rate : 0.512
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9865
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9928
            Specificity : 0.9937
         Pos Pred Value : 0.9940
         Neg Pred Value : 0.9924
             Prevalence : 0.5120
         Detection Rate : 0.5083
   Detection Prevalence : 0.5114
      Balanced Accuracy : 0.9932

       'Positive' Class : 1
```

Conclusions:

Overall, both SVM and neural network algorithms yielded impressively 100% accurate prediction results to the point of making my skeptical about the correctness of my models. In this assignment, I had to do the opposite to my usual actions. Instead of trying to improve a model, I was trying to "break it", or to find parameters that would make is less precise.

I used different kernels in my SVM models, and all of them but one (sigmoid) showed 100% accuracy on the testing data set. The SVM with the sigmoid kernel had the worst result – "only" 99.14% accuracy.

I had to use the same "breaking" logic for my neural network model. I change parameters to decrease the initial 100% accuracy. I manipulate complexity (size, or the number of hidden layers), maximum allowed number of iterations, initial weights, and decay (step). The changes in the parameters were neither exhaustive, not methodic, I was just trying to find factors that negatively impact the predictive power of a neural network.

Decreasing the number of maximum allowed iterations to 50 resulted in misclassification of 22 samples (or 98.65% accuracy). At the same time, limiting the number of iterations and decreasing decay surprisingly still resulted in 100% accuracy. Simultaneously increasing decay, limiting the maximum number of iterations and the network's size led to the worst result – 94.00% accuracy. However, decreasing size, changing initial weights and limiting the maximum number of iterations lead to 99.32% accuracy. So, I was not able to detect any straightforward (ex., linear) dependencies between the accuracy and the model's parameters. The most influential factor in this particular seems to be limiting the maximum number of iterations, and its combined effect with a bigger model step.

Overall, both SVM and ANNs turned out to yield very precise results on the mushroom dataset, potentially due to preprocessing (one hot full rank encoding) of the categorical variable.