MSDS 680 - Machine Learning, Week 5
Alla Topp

Artificial Neural Network and Support Vector Machine

Objective: Are the mushrooms safe to eat?

We'll work with the mushroom data set located at http://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/. we have 8124 mushrooms in the dataset. And each observation consists of 23 variables.

We'll first load the data into R.

```
#Load the data - we downloaded the data from the website
mushroom_df <- read.csv(url("http://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.data"),
                        header = FALSE, sep = ",")
str(mushroom_df)
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20
1	p	x	s	n	t	p	f	c	n	k	e	e	s	s	w	w	p	w	o	p
2	e	x	s	y	t	a	f	c	b	k	e	c	s	s	w	w	p	w	o	p
3	e	b	s	w	t	l	f	c	b	n	e	c	s	s	w	w	p	w	o	p
4	p	x	y	w	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	p
5	e	x	s	g	f	n	f	w	b	k	t	e	s	s	w	w	p	w	o	e

```
> str(mushroom_df)
'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 ...
```

Next, we should add attribute names to the dataset.

```
> #Rename the variables (colums), wo we have names for each column
> colnames(mushroom_df) <- c("edibility", "cap_shape", "cap_surface", "cap_color", "bruises", "odor",
+                          "grill_attachement", "grill_spacing", "grill_size", "grill_color",
+                          "stalk_shape", "stalk_root", "stalk_surface_above_ring",
+                          "stalk_surface_below_ring", "stalk_color_above_ring",
+                          "stalk_color_below_ring", "veil_type", "veil_color",
+                          "ring_number", "ring_type", "spore_print_color", "population", "habitat")
> str(mushroom_df)
'data.frame':   8124 obs. of  23 variables:
 $ edibility                : 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 ...
```

We can how we have data frame with column names now.

Because veil_type is a factor with only one level, we'll remove it from the data set.

```
sum(is.na(mushroom_df)) # checking how many values are missing in our dataset
mushroom_df <- subset(mushroom_df, select = -veil_type) #removing the column with 1 level (veil_type)
```

We have about 2480 of the missing values which are "NA" in the column stalk_root after applying formula is.na() We can see the column with missing values below:

```
stalk_shape stalk_root  stalk_surface_above_ring stalk_surface_below_ring stalk_color_above_ring stalk_color_be
e:3516      b    :3776  f: 552                   f: 600                   w       :4464          w       :4384
t:4608      c    : 556  k:2372                   k:2304                   p       :1872          p       :1872
            e    :1120  s:5176                   s:4936                   g       : 576          g       : 576
            r    : 192  y:  24                   y: 284                   n       : 448          n       : 512
            NA's:2480                                                     b       : 432          b       : 432
                                                                          o       : 192          o       : 192
                                                                          (Other): 140           (Other): 156
```

We are going to replace missing values with a character "u" which means unknown, we are going to avoid removing all the columns with the missing values:

```
sum(is.na(mushroom_ann))
#mushroom[ mushroom == "u" ] <- NA
levels <- levels(mushroom_ann$stalk_root)
levels[length(levels) + 1] <- "U"
mushroom_ann$stalk_root <- factor(mushroom_ann$stalk_root, levels = levels)
mushroom_ann$stalk_root[is.na(mushroom_ann$stalk_root)] <-"U"
summary(mushroom_ann)
sum(is.na(mushroom_ann))
```

After running summary, we can see that those missing values are substituted:

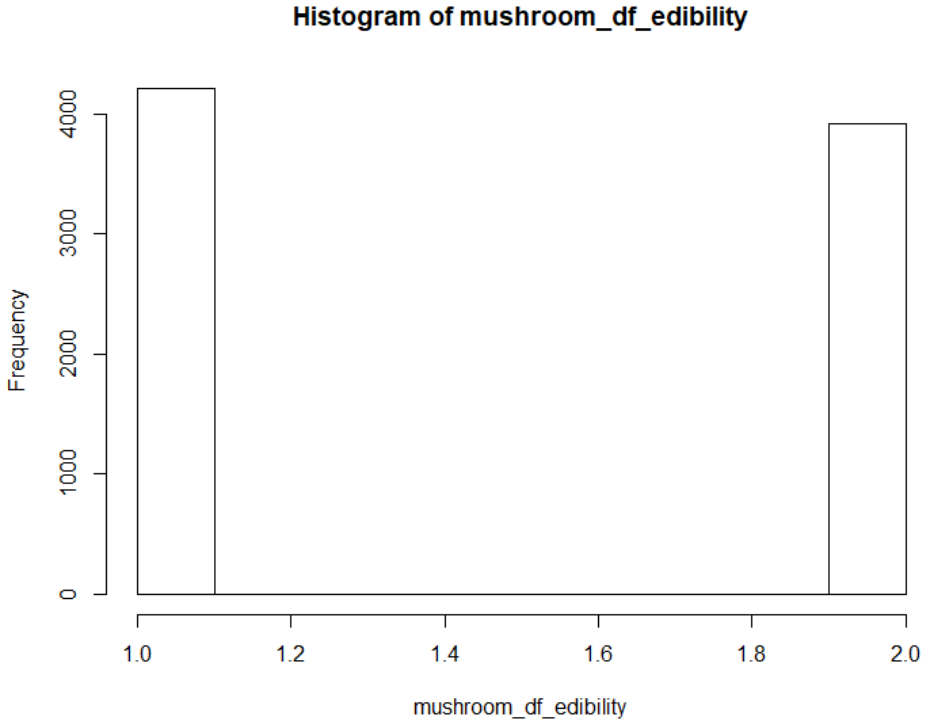
```
> summary(mushroom_df)
 edibility   cap_shape      cap_surface   cap_color       bruises      odor       grill_attachement grill_spacing grill_size  grill_color
 e:4208    Min.   :1.000   f:2320      n      :2284   f:4748   n      :3528   a: 210            c:6812        b:5612     b      :1728
 p:3916    1st Qu.:3.000   g:   4      g      :1840   t:3376   f      :2160   f:7914            w:1312        n:2512     p      :1492
           Median :4.000   s:2556      e      :1500            s      : 576                                              w      :1202
           Mean   :4.348   y:3244      y      :1072            y      : 576                                              n      :1048
           3rd Qu.:6.000               w      :1040            a      : 400                                              g      : 752
           Max.   :6.000               b      : 168            l      : 400                                              h      : 732
                                       (Other): 220            (Other): 484                                              (Other):1170
 stalk_shape stalk_root stalk_surface_above_ring stalk_surface_below_ring stalk_color_above_ring stalk_color_below_ring veil_color
 e:3516      b:3776     f: 552                   f: 600                   w      :4464           w      :4384           n:   96
 t:4608      c: 556     k:2372                   k:2304                   p      :1872           p      :1872           o:   96
             e:1120     s:5176                   s:4936                   g      : 576           g      : 576           w:7924
             r: 192     y:  24                   y: 284                   n      : 448           n      : 512           y:    8
             U:2480                                                       b      : 432           b      : 432
                                                                          o      : 192           o      : 192
                                                                          (Other): 140           (Other): 156
```

Let's now change all the factor variables to numeric variables so the SVM can run.

```
#change all the factor variables to numeric variables
mushroom_df$cap_shape <- as.numeric(mushroom_df$cap_shape)
mushroom_df$cap_surface <- as.numeric(mushroom_df$cap_surface)
mushroom_df$cap_color <- as.numeric(mushroom_df$cap_color)
mushroom_df$bruises <- as.numeric(mushroom_df$bruises)
mushroom_df$odor <- as.numeric(mushroom_df$odor)
mushroom_df$grill_attachement <- as.numeric(mushroom_df$grill_attachement)
mushroom_df$grill_spacing <- as.numeric(mushroom_df$grill_spacing)
mushroom_df$grill_size <- as.numeric(mushroom_df$grill_size)
mushroom_df$grill_color <- as.numeric(mushroom_df$grill_color)
mushroom_df$stalk_shape <- as.numeric(mushroom_df$stalk_shape)
mushroom_df$stalk_root <- as.numeric(mushroom_df$stalk_root)
mushroom_df$stalk_surface_above_ring <- as.numeric(mushroom_df$cap_shape)
mushroom_df$stalk_surface_below_ring <- as.numeric(mushroom_df$cap_shape)
mushroom_df$stalk_color_above_ring <- as.numeric(mushroom_df$stalk_color_above_ring)
mushroom_df$stalk_color_below_ring <- as.numeric(mushroom_df$stalk_color_below_ring)
mushroom_df$veil_color <- as.numeric(mushroom_df$veil_color)
mushroom_df$ring_number <- as.numeric(mushroom_df$ring_number)
mushroom_df$ring_type <- as.numeric(mushroom_df$ring_type)
mushroom_df$spore_print_color <- as.numeric(mushroom_df$spore_print_color)
mushroom_df$population <- as.numeric(mushroom_df$population)
mushroom_df$habitat <- as.numeric(mushroom_df$habitat)
```

After we changed all variables to numeric, we would like to look at the histogram to explore the levels of mushroom edibility.

```
> #hisrogram
> mushroom_df_edibility <- as.numeric(mushroom_df$edibility)
> hist(mushroom_df_edibility)
> table(mushroom_df_edibility)
mushroom_df_edibility
   1    2
4208 3916
```



Looking at the table we can see that we have 4208 edible mushrooms of the 8124 and 2916 not edible mushroom of the 8124. To figure out the percentage of it we need to do some calculations.

```
> #quick calculation, to see what is % of the poisonous and edible
> 4208/8124
[1] 0.5179714
> 3916/8124
[1] 0.4820286
```

We can see above, approx. 52% of the mushroom entries in the mushroom data set are edible, and approx 48% are poisonous (or unclear and not definitely safe to eat).

Our next step is to build the SVM model and we are going to start with splitting data into training and testing sets. We'll experiment with different kernels and compare the accuracy. We should first split the data into training and testing sets. We'll use 70% of the data for training and save 30% for testing.

```
> #splitting data into train and test sets
> 8124*0.7
[1] 5686.8
> 8124*0.3
[1] 2437.2
> s <- sample(8124, 5687)
> mushroom_train <- mushroom_df[s, ]
> mushroom_test <- mushroom_df[-s, ]
> dim(mushroom_train)
[1] 5687   22
> dim(mushroom_test)
[1] 2437   22
```

To build the SVM model we need to use the library e1071 package. We will build a few models with different kernels.

The first model will be with kernel = "linear" and cost =1

```
#building SVM model #1
library(e1071)
str(mushroom_train) #data frame of train data
svm_model_mush <- svm(edibility ~ ., data = mushroom_train, kernel = 'linear', cost = 1, scale = FALSE)
print(svm_model_mush)

svm.pred = predict(svm_model_mush, mushroom_test[, !names(mushroom_test) %in% c("edibility")])
svm.table = table(svm.pred, mushroom_test$edibility)
svm.table
```

```
> svm_model_mush <- svm(edibility ~ ., data = mushroom_train, kernel = 'linear', cost = 1, scale = FALSE)
> print(svm_model_mush) #print the model

Call:
svm(formula = edibility ~ ., data = mushroom_train, kernel = "linear", cost = 1, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.04761905

Number of Support Vectors:  831
```

```
> summary(svm_model_mush) #summary of the model

Call:
svm(formula = edibility ~ ., data = mushroom_train, kernel = "linear", cost = 1, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.04761905

Number of Support Vectors:  831

 ( 416 415 )


Number of Classes:  2

Levels:
 e p
```

```
> confusionMatrix(svm.table) #use a confusion matrix to measure the performance of the model
Confusion Matrix and Statistics


svm.pred    e    p
       e 1191   62
       p   39 1145

               Accuracy : 0.9586
                 95% CI : (0.9499, 0.9661)
    No Information Rate : 0.5047
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9171
 Mcnemar's Test P-Value : 0.02859

            Sensitivity : 0.9683
            Specificity : 0.9486
         Pos Pred Value : 0.9505
         Neg Pred Value : 0.9671
             Prevalence : 0.5047
         Detection Rate : 0.4887
   Detection Prevalence : 0.5142
      Balanced Accuracy : 0.9585

       'Positive' Class : e
```

We can see tat accuracy level of SVM is 96% which is a great result already, but we should check out the model with different parameter like cost and kernel.

To improve the model, we can try to change the values of cost, which was 1 and now we are going to try the model with cost = 100.

```
#SVM model #2
svm2 <- svm(edibility ~ ., data = mushroom_train, kernel = 'linear', cost = 100, scale = FALSE)
summary(svm2)

svm.pred2 = predict(svm2, mushroom_test[, !names(mushroom_test) %in% c("edibility")])
svm.table2 = table(svm.pred2, mushroom_test$edibility)
svm.table2

confusionMatrix(svm.table2)
```

```
> summary(svm2)

Call:
svm(formula = edibility ~ ., data = mushroom_train, kernel = "linear", cost = 100, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  100
      gamma:  0.04761905

Number of Support Vectors:  852

 ( 441 411 )


Number of Classes:  2

Levels:
 e p

> confusionMatrix(svm.table2)
Confusion Matrix and Statistics


svm.pred2    e    p
        e 1184   20
        p   46 1187

               Accuracy : 0.9729
                 95% CI : (0.9657, 0.979)
    No Information Rate : 0.5047
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9458
 Mcnemar's Test P-Value : 0.002089

            Sensitivity : 0.9626
            Specificity : 0.9834
         Pos Pred Value : 0.9834
         Neg Pred Value : 0.9627
             Prevalence : 0.5047
         Detection Rate : 0.4858
   Detection Prevalence : 0.4941
      Balanced Accuracy : 0.9730

       'Positive' Class : e
```

Looking at the second model (the attempt to improve it) with cost = 100, we can see that it slightly improves with the accuracy level 97%. Next, we will try with different kernel “radial” and will see how it goes.

```
#SVM model #3 with kernel = "radial"
svm3 <- svm(edibility ~ ., data = mushroom_train, kernel = 'radial', cost = 100, scale = FALSE)
summary(svm3)

svm.pred3 = predict(svm3, mushroom_test[, !names(mushroom_test) %in% c("edibility")])
svm.table3 = table(svm.pred3, mushroom_test$edibility)
svm.table3

confusionMatrix(svm.table3)
```

```
> summary(svm3)

Call:
svm(formula = edibility ~ ., data = mushroom_train, kernel = "radial", cost = 100, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  100
      gamma:  0.04761905

Number of Support Vectors:  773

 ( 424 349 )


Number of Classes:  2

Levels:
 e p

> confusionMatrix(svm.table3)
Confusion Matrix and Statistics


svm.pred3    e    p
        e 1260    0
        p    0 1177

               Accuracy : 1
                 95% CI : (0.9985, 1)
    No Information Rate : 0.517
    P-Value [Acc > NIR] : < 2.2e-16

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

            Sensitivity : 1.000
            Specificity : 1.000
         Pos Pred Value : 1.000
         Neg Pred Value : 1.000
             Prevalence : 0.517
         Detection Rate : 0.517
   Detection Prevalence : 0.517
      Balanced Accuracy : 1.000

       'Positive' Class : e
```

The radial SVM predicted the test set edibility labels with 100% accuracy. There is no way to improve the model, but just to compare we are going to build the model with kernel = “polynomial”. And see if we can get a different result.

```
#SVM model #4 with kernel = "polynomial"
svm4 <- svm(edibility ~ ., data = mushroom_train, kernel = 'polynomial', cost = 100, scale = FALSE)
summary(svm4)

svm.pred4 = predict(svm4, mushroom_test[, !names(mushroom_test) %in% c("edibility")])
svm.table4 = table(svm.pred4, mushroom_test$edibility) #set the table to produce it
svm.table4 #print the table

confusionMatrix(svm.table4)
```

```
> summary(svm4)

Call:
svm(formula = edibility ~ ., data = mushroom_train, kernel = "polynomial", cost = 100, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  100
     degree:  3
      gamma:  0.04761905
     coef.0:  0

Number of Support Vectors:  122

 ( 71 51 )


Number of Classes:  2

Levels:
 e p
```

```
> confusionMatrix(svm.table4)
Confusion Matrix and Statistics


svm.pred4    e    p
        e 1230    0
        p    0 1207

               Accuracy : 1
                 95% CI : (0.9985, 1)
    No Information Rate : 0.5047
    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 : 0.5047
         Detection Rate : 0.5047
   Detection Prevalence : 0.5047
      Balanced Accuracy : 1.0000

       'Positive' Class : e
```

So, as we thought the last attempt gave us also 100% accuracy in the SVM model! Now we can move on to neural network algorithms.

ANN Model

```
#_____________________________________
#Nueral Network algorithm

mushroom_ann <- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.data",
                           na.strings="?", sep = "," )
str(mushroom_ann)
#Rename the variables (colums), wo we have names for each column
colnames(mushroom_ann) <- c("edibility", "cap_shape", "cap_surface", "cap_color", "bruises", "odor",
                            "grill_attachement", "grill_spacing", "grill_size", "grill_color",
                            "stalk_shape", "stalk_root", "stalk_surface_above_ring",
                            "stalk_surface_below_ring", "stalk_color_above_ring",
                            "stalk_color_below_ring", "veil_type", "veil_color",
                            "ring_number", "ring_type", "spore_print_color", "population", "habitat")
```

Next, we transform the factors into dummy variables and create training ang testing sets for the data.

```
#creating train and test sets and converting into dummy variables
splitting <- createDataPartition(mushroom_ann$edibility, p = .7, list = FALSE) #splitting data into 70/30
dummy <- subset(mushroom_ann, select = -edibility)
mush_ann_dummy <- dummyVars(~., data = dummy, sep = ".") #creating a full set of dummy variables
mush_ann_dummy <- data.frame(predict(mush_ann_dummy, dummy))
mush_ann_dummy$edibility <- mushroom_ann$edibility #dummy vars of edibility

train_ann <- mush_ann_dummy[splitting,] #split into .7
test_ann <- mush_ann_dummy[-splitting,] #split into .3
testLabels <- subset(test_ann, select = edibility) #selecting variables from edibility
testset <- subset(test_ann, select = -edibility) #creating tetsset of edibility
```

	cap_shape.b	cap_shape.c	cap_shape.f	cap_shape.k	cap_shape.s	cap_shape.x	cap_surface.f	cap_surface.g	cap_surface.s
1	0	0	0	0	0	1	0	0	1
2	0	0	0	0	0	1	0	0	1
4	0	0	0	0	0	1	0	0	0
5	0	0	0	0	0	1	0	0	1
6	0	0	0	0	0	1	0	0	0
7	1	0	0	0	0	0	0	0	1
10	1	0	0	0	0	0	0	0	1
11	0	0	0	0	0	1	0	0	0

```
> net <- nnet(edibility ~ ., data = train_ann, size = 2, rang = 0.1, maxit = 200) #train the neural network with nnet
# weights:  227
initial  value 3944.449514
iter  10 value 407.025062
iter  20 value 42.834183
iter  30 value 2.253616
iter  40 value 0.032950
iter  50 value 0.001209
iter  60 value 0.000160
iter  70 value 0.000159
final  value 0.000088
converged
> summary(net) #obtain information about the trained neural network
a 111-2-1 network with 227 weights
options were - entropy fitting
   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
   -2.33     0.38    -0.88    -0.74     0.16     0.07    -0.99     4.14    -5.67     0.25    -0.79    -0.76    13.00
 i13->h1  i14->h1  i15->h1  i16->h1  i17->h1  i18->h1  i19->h1  i20->h1  i21->h1  i22->h1  i23->h1  i24->h1  i25->h1
   -5.57    -5.56    -5.83    -4.42     9.66     7.55    -5.93    -4.41     0.02    -2.32     7.73    -9.12   -25.13
 i26->h1  i27->h1  i28->h1  i29->h1  i30->h1  i31->h1  i32->h1  i33->h1  i34->h1  i35->h1  i36->h1  i37->h1  i38->h1
    5.78    10.80    10.30     7.37    -4.62    -5.28    -2.31     0.08    -1.71    -0.44     7.29    -9.48    -9.97
 i39->h1  i40->h1  i41->h1  i42->h1  i43->h1  i44->h1  i45->h1  i46->h1  i47->h1  i48->h1  i49->h1  i50->h1  i51->h1
    2.60     2.90     2.07     4.37     2.89    -0.73     3.76   -10.18    -3.04     3.78    -0.73    -9.02     6.77
 i52->h1  i53->h1  i54->h1  i55->h1  i56->h1  i57->h1  i58->h1  i59->h1  i60->h1  i61->h1  i62->h1  i63->h1  i64->h1
    7.13   -10.12     4.11    -3.35    -3.47    -2.38     0.17     3.48    -6.84    10.79     1.47     4.82    -5.73
 i65->h1  i66->h1  i67->h1  i68->h1  i69->h1  i70->h1  i71->h1  i72->h1  i73->h1  i74->h1  i75->h1  i76->h1  i77->h1
   -1.29    -2.36     1.80    -5.15    -7.75    10.80     2.00     4.60    -4.57    -1.23    -1.47     0.25    -4.62
 i78->h1  i79->h1  i80->h1  i81->h1  i82->h1  i83->h1  i84->h1  i85->h1  i86->h1  i87->h1  i88->h1  i89->h1  i90->h1
   -0.59    -0.65     4.12    -5.05    10.90     0.04   -13.14   -16.82    23.81   -22.34    10.91     2.02    -0.16
 i91->h1  i92->h1  i93->h1  i94->h1  i95->h1  i96->h1  i97->h1  i98->h1  i99->h1 i100->h1 i101->h1 i102->h1 i103->h1
   -1.19     8.17     8.52    -0.28   -50.26    10.05    23.19    -0.17    -3.13    -0.40     4.23    -1.03    -1.13
i104->h1 i105->h1 i106->h1 i107->h1 i108->h1 i109->h1 i110->h1 i111->h1
```

Above we demonstrated steps to train a neural network model with the nnet package. We first used nnet to train the neural network. With this function, we can set the classification formula, source of data, number of hidden units in the size parameter, initial random weight in the rang parameter, parameter for weight decay in the decay parameter, and the maximum iteration in the maxit parameter. As we set maxit to 200, the training process repeatedly runs till the value of the fitting criterion plus the decay term converge. Finally, we used the summary function to obtain information about the built neural network, which reveals that the model is built with 11-2-1 networks with 227 weights. Also, the model shows a list of weight transitions from one node to another at the bottom of the printed message.

As we have trained a neural network with nnet in previously, we can now predict the labels of the testing dataset based on the trained neural network. Furthermore, we can assess the model with a confusion matrix adapted from the caret package.

```
#prediction ann
mush_ann.predict <- predict(net, testset, type = "class") #Generate the predictions of the testing dataset
net.table <- table(test_ann$edibility, mush_ann.predict) #Generate a classification table based on the prec
net.table
confusionMatrix(net.table) # generate a confusion matrix based on the classification table
```

```
> confusionMatrix(net.table) # generate a confusion matrix based on the classification table
Confusion Matrix and Statistics

   mush_ann.predict
       e    p
  e 1260    2
  p    0 1174

               Accuracy : 0.9992
                 95% CI : (0.997, 0.9999)
    No Information Rate : 0.5172
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9984
 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 1.0000
            Specificity : 0.9983
         Pos Pred Value : 0.9984
         Neg Pred Value : 1.0000
             Prevalence : 0.5172
         Detection Rate : 0.5172
   Detection Prevalence : 0.5181
      Balanced Accuracy : 0.9991

       'Positive' Class : e
```

ANN with size = 1

```
> #using library nnet to construct the ANN
> net <- nnet(edibility ~ ., data = train_ann, size = 1, rang = 0.1, maxit = 200) #train the neural network with nnet
# weights:  119
initial  value 3943.368187
iter  10 value 3366.682253
iter  20 value 1405.046431
iter  30 value 1404.966551
final  value 1404.966107
converged
> summary(net) #obtain information about the trained neural network
a 116-1-1 network with 119 weights
options were - entropy fitting
   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
   -2.55   -11.94    -1.19     3.12    -1.71     6.46     2.76     3.55    -0.29    -0.95    -4.88    -9.86     2.28    11.30
 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  i28->h1
    7.31   -12.27     4.79     5.82    -7.33    -8.25     6.47    -8.83    -6.67   -23.41   -14.34    -7.60    -1.66    84.18
 i30->h1  i31->h1  i32->h1  i33->h1  i34->h1  i35->h1  i36->h1  i37->h1  i38->h1  i39->h1  i40->h1  i41->h1  i42->h1  i43->h1
  -11.37   -10.15     3.36    -5.90   -27.77    25.33     4.55    -7.08   -15.45     5.16    -2.34     0.43    -5.38     2.41
 i45->h1  i46->h1  i47->h1  i48->h1  i49->h1  i50->h1  i51->h1  i52->h1  i53->h1  i54->h1  i55->h1  i56->h1  i57->h1  i58->h1
    5.59    -9.15    12.10     3.69    -0.45   -34.37    31.91     7.42   -31.20     4.34    -8.71    25.93    -6.43     0.64
 i60->h1  i61->h1  i62->h1  i63->h1  i64->h1  i65->h1  i66->h1  i67->h1  i68->h1  i69->h1  i70->h1  i71->h1  i72->h1  i73->h1
   -1.13    11.75     9.36     5.19   -28.65     0.95    -1.73     6.13    16.57     0.96     4.15   -10.05   -15.97    -3.29
 i75->h1  i76->h1  i77->h1  i78->h1  i79->h1  i80->h1  i81->h1  i82->h1  i83->h1  i84->h1  i85->h1  i86->h1  i87->h1  i88->h1
   -1.79     7.17    15.25    -5.32     4.10    -5.35    -7.35   -10.47     1.23     2.89    -3.39    -3.32    -1.71   -14.75
 i90->h1  i91->h1  i92->h1  i93->h1  i94->h1  i95->h1  i96->h1  i97->h1  i98->h1  i99->h1 i100->h1 i101->h1 i102->h1 i103->h1 i
  -25.48    21.74    -3.46    -1.72     6.40     1.57     1.46     1.52     1.47     0.95   -23.59     9.20     2.65     2.07
i105->h1 i106->h1 i107->h1 i108->h1 i109->h1 i110->h1 i111->h1 i112->h1 i113->h1 i114->h1 i115->h1 i116->h1
    8.04    -2.20   -24.70     5.12    14.56    12.75    -9.91     5.14   -30.67     3.60     7.28     9.14
  b->o  h1->o
  1.73  -7.75
```

```
> confusionMatrix(net.table) # generate a confusion matrix based on the classification table
Confusion Matrix and Statistics

   mush_ann.predict
       e    p
  e 1042  220
  p    2 1172

               Accuracy : 0.9089
                 95% CI : (0.8967, 0.92)
    No Information Rate : 0.5714
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8187
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9981
            Specificity : 0.8420
         Pos Pred Value : 0.8257
         Neg Pred Value : 0.9983
             Prevalence : 0.4286
         Detection Rate : 0.4278
   Detection Prevalence : 0.5181
      Balanced Accuracy : 0.9200

       'Positive' Class : e
```

We can see that accuracy rate of ANN is extremely close to 100% which is an outstanding result. When we compare all the models, we analyzed like svm with different parameters and ann, we can see pretty much the same result with a very high accuracy rate from 97-100. Although, in the last model where we tried size =1 (hidden layer), we can see a huge regress in accuracy. This is not the way to go with a model. I assume that the best architecture of ANN would be with size 2. The most accurate output was with the last svm model and ANN algorithm.