DA5030: Final Project

Code

Virly Ananda

8/17/21

PREDICTING EARLY STAGE OF DEMENTIA

Load and Explore Dataset

Hide

```
# Download the dataset fromt the official website locally

# Click on the zipped file: Normally it is stored in Downloads

# Load CSV file to R
alzheimers<-read.csv("/Users/virlyananda/Downloads/alzheimer.csv")

# Check the dimension and properties of the dataset
str(alzheimers)
```

```
'data.frame':   373 obs. of  10 variables:
 $ Group: Factor w/ 3 levels "Converted","Demented",..: 3 3 2 2 2 3 3 3 3 3 ...
 $ M.F  : Factor w/ 2 levels "F","M": 2 2 2 2 2 1 1 2 2 2 ...
 $ Age  : int  87 88 75 76 80 88 90 80 83 85 ...
 $ EDUC : int  14 14 12 12 12 18 18 12 12 12 ...
 $ SES  : int  2 2 NA NA NA 3 3 4 4 4 ...
 $ MMSE : int  27 30 23 28 22 28 27 28 29 30 ...
 $ CDR  : num  0 0 0.5 0.5 0.5 0 0 0 0.5 0 ...
 $ eTIV : int  1987 2004 1678 1738 1698 1215 1200 1689 1701 1699 ...
 $ nWBV : num  0.696 0.681 0.736 0.713 0.701 0.71 0.718 0.712 0.711 0.705 ...
 $ ASF  : num  0.883 0.876 1.046 1.01 1.034 ...
```

ABSTRACT

This dataset is obtained from OASIS (Open Access Series of Imaging Studies) commonly used for machine learning application to detect early stages of dementia. For the purpose of this project, the author specify “Group” column as the target variable. The “Group” column is a class-based category of whether the 373 subjects are on the stages of dementia or not.

Group –> Class
Age –> Age
EDUC –> Years of Education
SES –> Socioeconomic Status / 1-5

MMSE –> Mini Mental State Examination
CDR –> Clinical Dementia Rating
eTIV –> Estimated total intracranial volume
nWBV –> Normalize Whole Brain Volume
ASF –> Atlas Scaling Factor

From the dataset shown above, we know that there are 2 factors indicating the stages of whether a patient has dementia and which gender each subject belongs to.

Based on the structure of the dataset we also notice that most of the variables are numerics/integers, this could be helpful for us in analyzing the correlation among variables in the later section. Proceeding to column 5, we also see missing values exist within the dataset. Thus, we should impute them accordingly.

Before proceeding to the next step, we first need to convert all the factor variables into numerics to obtain correlation.

FEATURE ENGINEERING

ONE HOT CODING:

Hide

```
# Convert gender column M.F to one hot coding
alzheimers$M.F<-ifelse(alzheimers$M.F == "M", 0, 1) # Male = 0, Female = 1

# Convert target value column Group to one hot coding: Since we have 3 levels, we com
bine "converted" and "demented" into 1 level due to their early stage of dementia rep
resentation.
levels(alzheimers$Group)
```

```
[1] "Converted"   "Demented"    "Nondemented"
```

Hide

```
levels(alzheimers$Group)<-c("Demented", "Demented", "Nondemented")
alzheimers$Group<-ifelse(alzheimers$Group == "Nondemented", 0, 1) # Nondemented = 0,
Demented = 1

# Check the structure of the updated dataset
str(alzheimers)
```

```
'data.frame':   373 obs. of  10 variables:
 $ Group: num  0 0 1 1 1 0 0 0 0 0 ...
 $ M.F  : num  0 0 0 0 0 1 1 0 0 0 ...
 $ Age  : int  87 88 75 76 80 88 90 80 83 85 ...
 $ EDUC : int  14 14 12 12 12 18 18 12 12 12 ...
 $ SES  : int  2 2 NA NA NA 3 3 4 4 4 ...
 $ MMSE : int  27 30 23 28 22 28 27 28 29 30 ...
 $ CDR  : num  0 0 0.5 0.5 0.5 0 0 0 0.5 0 ...
 $ eTIV : int  1987 2004 1678 1738 1698 1215 1200 1689 1701 1699 ...
 $ nWBV : num  0.696 0.681 0.736 0.713 0.701 0.71 0.718 0.712 0.711 0.705 ...
 $ ASF  : num  0.883 0.876 1.046 1.01 1.034 ...
```

MISSING VALUE IMPUTATION:

Hide

```
# Check which columns have missing values
names(which(sapply(alzheimers, anyNA)))
```

```
[1] "SES"  "MMSE"
```

Based on the result, SES (Socioeconomic Status) and MMSE (Mini Mental State Examination) have missing values. Since both of these variables are integers, we apply mean imputation:

Hide

```
# MEAN APPLIED
alzheimers$SES[is.na(alzheimers$SES)]<-mean(alzheimers$SES[!is.na(alzheimers$SES)]) #
SES

alzheimers$MMSE[is.na(alzheimers$MMSE)]<-mean(alzheimers$MMSE[!is.na(alzheimers$MMSE)
]) # MMSE
```

Hide

```
# Check whether the updated dataset has missing values
anyNA(alzheimers)
```

```
[1] FALSE
```

UNDERSTAND CORRELATION BETWEEN VARIABLES

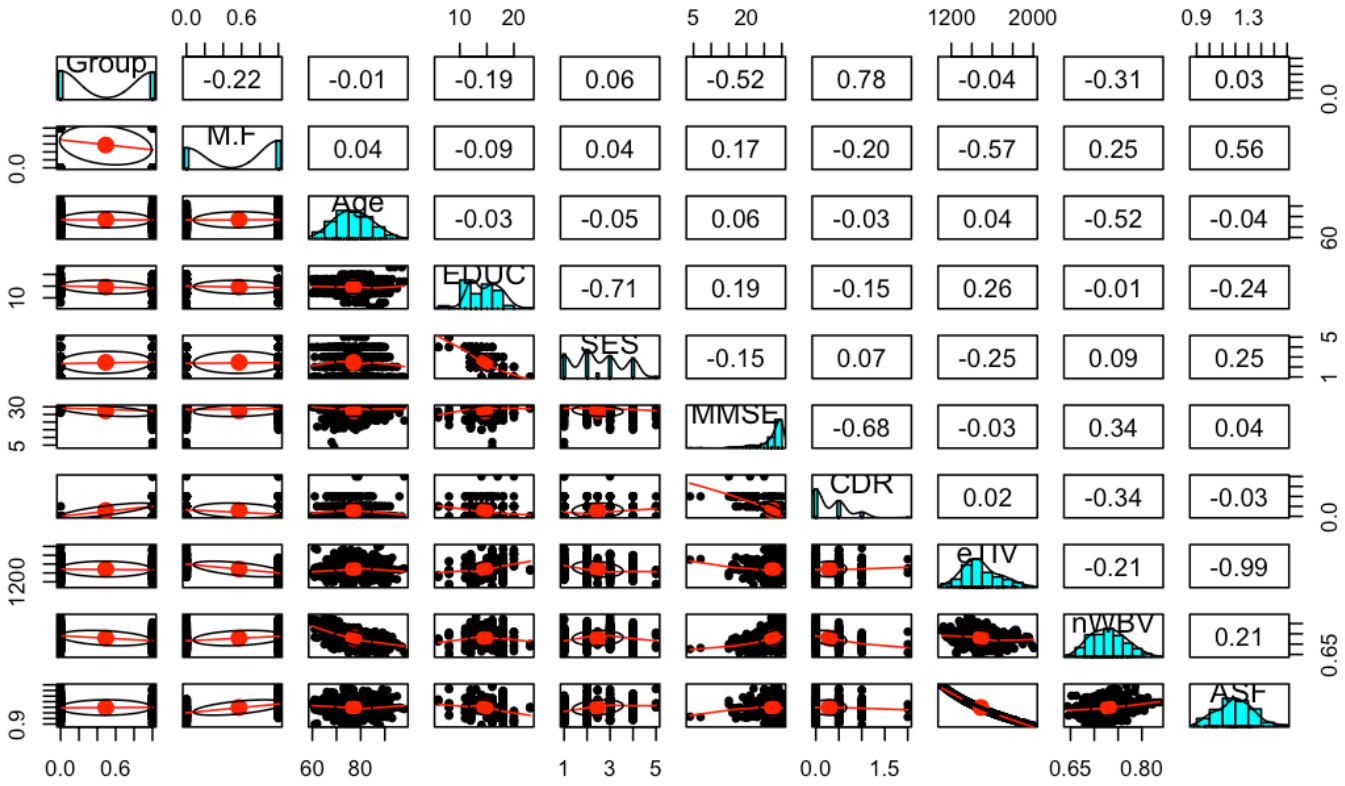
Before proceding into the rescaling features part, we first need to understand how the data is being distributed and how the target variable correlates with the other supporting features. To do this, we load pysch package and implement pairs.panels to study the relationship of the variables:

Hide

```
# install.packages("pysch")
library(psych)
```

Hide

```
pairs.panels(alzheimers)
```



From a slight observation obtained based on the pairs.panels, we can see that there are about 5 features that seem to have bell-shaped distribution (e.g, Age, EDUC, eTIV, nWBV, ASF), while the rest are non-normal. Although visualizing the histogram seems like a decent way to detect how the data is ditributed, we further need to detect if those data contain outliers.

Summary of features with medium-strong correlation with Group column:

- Group & M.F = -0.22
- Group & EDUC = -0.19
- Group & MMSE = -0.52
- Group & CDR = 0.78 (Fairly strong)
- Group & nWBV = -0.31

Summary of features with weak-no correlation with Group column:

- Group & Age = -0.01
- Group & SES = 0.06
- Group & eTIV = -0.04
- Group & ASF = 0.03

As we notice, the correlation between Group and CDR is fairly strong multidisciplinary. To avoid multicolinearity, we can remove CDF feature along with the features that have weak correlation shown above.

Hide

```
# Load dplyr
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

Hide

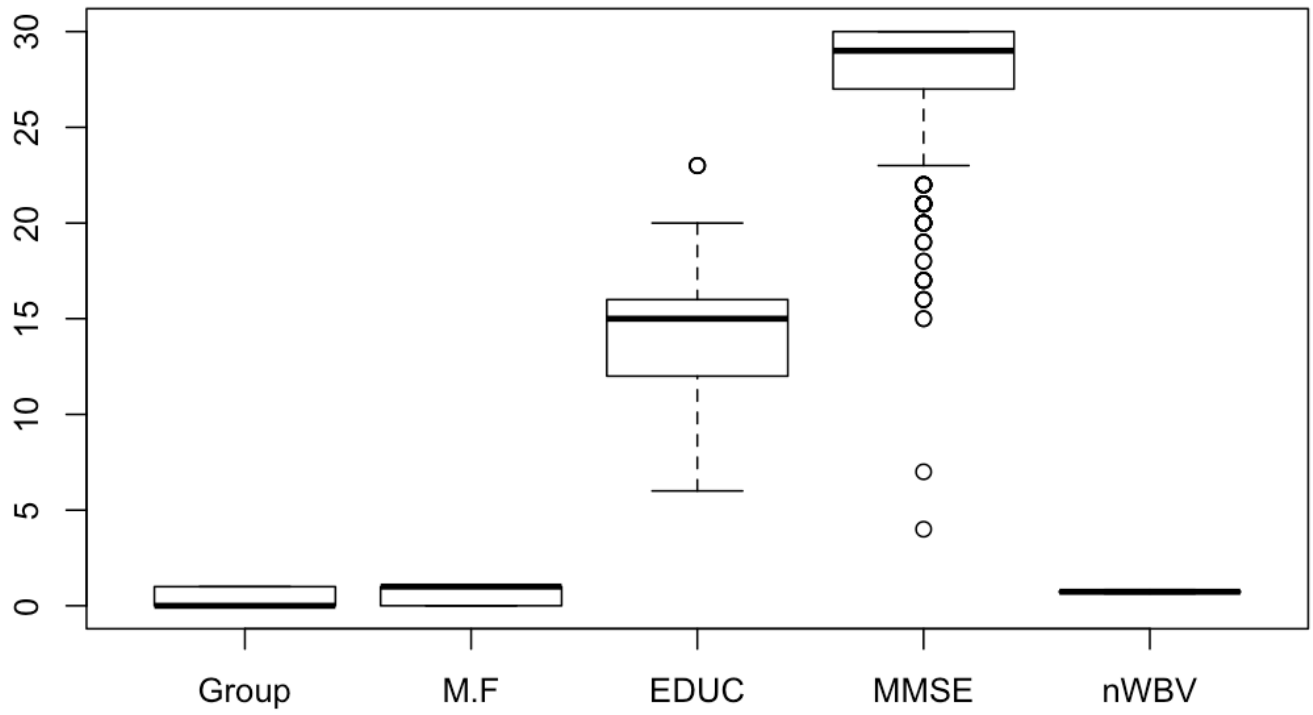
```
# Remove unnecessary features
alzheimersdf<-alzheimers %>%
  select(-c(CDR, Age, SES, eTIV, ASF))

# Check the structure of updated dataset
str(alzheimersdf)
```

```
'data.frame':   373 obs. of  5 variables:
 $ Group: num  0 0 1 1 1 0 0 0 0 0 ...
 $ M.F  : num  0 0 0 0 0 1 1 0 0 0 ...
 $ EDUC : int  14 14 12 12 12 18 18 12 12 12 ...
 $ MMSE : num  27 30 23 28 22 28 27 28 29 30 ...
 $ nWBV : num  0.696 0.681 0.736 0.713 0.701 0.71 0.718 0.712 0.711 0.705 ...
```

DETECT OUTLIERS

Hide

```
# Apply boxplot to detect outliers
boxplot(alzheimersdf)
```



From the boxplot visualization above, we can see that EDUC and MMSE seem to have outliers points. We can eliminate those outliers points with IQR estimation:

Hide

```
# Remove outliers through IQR: EDUC
summary(alzheimersdf$EDUC)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    6.0    12.0    15.0    14.6    16.0    23.0 
```

Hide

```
IQR_EDUC<- 16 - 12
upfen_EDUC<- 16 + 1.5*IQR_EDUC
upfen_EDUC
```

```
[1] 22
```

Hide

```
# Remove outliers through IQR: MMSE
summary(alzheimersdf$MMSE)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   4.00   27.00   29.00   27.34   30.00   30.00 
```

Hide

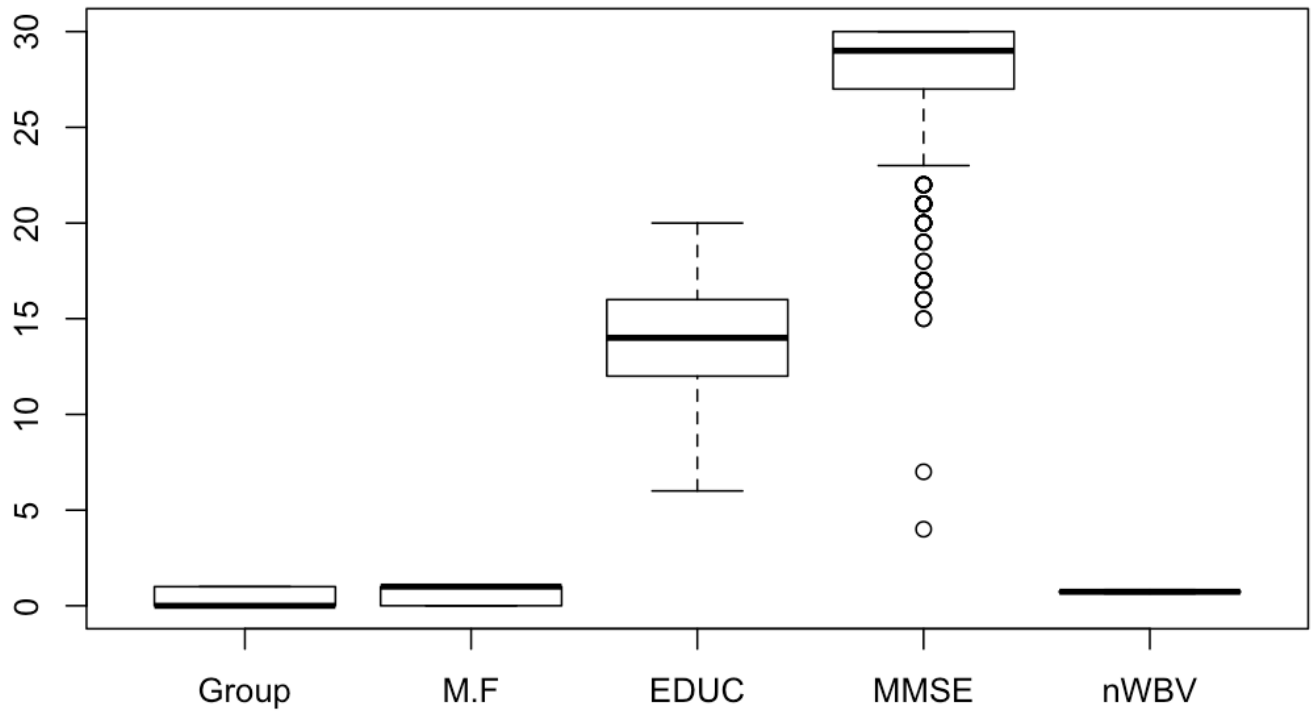
```
IQR_MMSE<- 30 - 27
upfen_MMSE<- 30 + 1.5*IQR_MMSE
upfen_MMSE
```

```
[1] 34.5
```

CREATE A NEW DATAFRAME WITHOUT OUTLIERS

Hide

```
# Remove outliers in the dataset
alzheimersdf.clean<-subset(alzheimersdf, Group<=1 & M.F<=1 & EDUC<=22 & MMSE<=34.5 & nWBV<=0.8370)

# visualize dataset
boxplot(alzheimersdf.clean)
```



Based on the visualization shown above, as we remove the outliers through IQR estimation, we were able to remove outliers within EDUC but not with MMSE, this could be due to how the data is distributed. MMSE seems to be a non-normal data in this case as the outliers points could be a representation of how non-normal the data is.

DATA TRANSFORMATION

In this section, our features should be scaled to obtain better algorithm. Since our data looks slightly non-normal based on the outliers detection shown previously, we try applying normalization and see whether our feature min-max return 0 and 1:

Hide

```
normalize <- function(x) {
    return((x - min(x)) / (max(x) - min(x)))
  }
```

Hide

```
# Only normalize continous variables
alzheimers_norm <- as.data.frame(lapply(alzheimersdf.clean[,3:5], normalize))

summary(alzheimers_norm)
```

```
      EDUC             MMSE             nWBV        
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.4286   1st Qu.:0.8846   1st Qu.:0.2915  
 Median :0.5714   Median :0.9615   Median :0.4404  
 Mean   :0.6093   Mean   :0.8972   Mean   :0.4448  
 3rd Qu.:0.7143   3rd Qu.:1.0000   3rd Qu.:0.5803  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
```

Hide

```
# Combine our categorized variables and normalized continuous variables as one datase
t
alzheimersdf.clean2<-alzheimersdf.clean %>%
  select(c(Group, M.F))

alzheimers_norm<-cbind(alzheimersdf.clean2, alzheimers_norm)

summary(alzheimers_norm)
```

```
     Group             M.F              EDUC             MMSE             nWBV        
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.4286   1st Qu.:0.8846   1st Qu.:0.2915  
 Median :0.0000   Median :1.0000   Median :0.5714   Median :0.9615   Median :0.4404  
 Mean   :0.4946   Mean   :0.5676   Mean   :0.6093   Mean   :0.8972   Mean   :0.4448  
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.7143   3rd Qu.:1.0000   3rd Qu.:0.5803  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
```

CALCULATE PCA

In this section, we implement PCA (Principal Component Analysis) to better analyze the dataset information.

Hide

```
# Apply prcomp function to calculate PCA
alzheimers_pca<-prcomp(alzheimers_norm, scale. = F) # Here we specify scale to F beca
use we have already normalized our features beforehand

# square root of eigenvalues
alzheimers_pca$sdev
```

```
[1] 0.5590382 0.4458684 0.1927927 0.1792177 0.1134499
```

Hide

```
# Check the rotation
alzheimers_pca$rotation
```

```
              PC1         PC2         PC3         PC4          PC5
Group  0.72650855 -0.65129283 -0.10719988  0.15620683 -0.110048152
M.F   -0.65999392 -0.74214160 -0.08452618 -0.08049359  0.003160664
EDUC  -0.02219483  0.12768572 -0.98380661  0.04239642  0.116322158
MMSE  -0.12294243  0.09186645 -0.08942581  0.21726590 -0.959814706
nWBV  -0.14486514  0.01733256  0.07410188  0.95922796  0.230443704
```

Hide

```
# Check the score
head(alzheimers_pca$x)
```

```
          PC1        PC2        PC3         PC4         PC5
1  0.04305073 0.73431801  0.1263467 -0.20412083  0.01986264
2  0.04012402 0.74357089  0.1102691 -0.25360309 -0.10879534
3  0.76162032 0.05424331  0.1888064  0.11140757  0.08862119
4  0.75524127 0.06984439  0.1627783  0.03887733 -0.12342076
5  0.79261976 0.04756677  0.1788076 -0.07090207  0.08374683
6 -0.63852147 0.03344866 -0.2373313 -0.19456348  0.03605838
```

The author focuses on evaluating the overall evaluation from the concepts mentioned above including the interpretation of correlation matrix as well as pairs. panels that show how each variables are distributed within the dataset.

DATA SPLITTING: TRAINING & TEST SET

Hide

```
library(caret)
```

In this section, dataset is split with a ratio of 80:20. From the codes below, we apply sample to randomly generate rows from the dataset and split into 80% to training and 20% to test dataset. Ratio of 80:20 is commonly known as Pareto principle and used mainly for splitting dataset that contains in various range of observations. Since the data is considered small, 80:20 can gives a more unbiased result.

Hide

```
# Set seed to create a reproducible set
set.seed(123)

# Generate random rows with ratio of 80:20
RandomRows<- sort(sample(nrow(alzheimers_norm), nrow(alzheimers_norm)*.80))
alzheimers_train<-alzheimers_norm[RandomRows,] # Train 80%
alzheimers_test<-alzheimers_norm[-RandomRows,] # Test 20%
```

Hide

```
# Check target variables to labelled Y and N
alzheimers_train$Group<-factor(alzheimers_train$Group, levels=c(0,1), labels=c("N","Y
")) # 0 for N and 1 for Y
alzheimers_test$Group<-factor(alzheimers_test$Group, levels = c(0,1), labels = c("N",
"Y"))
```

Hide

```
# Adjust control to be applied later on for Caret training model implementation
control<-trainControl(method = "cv", number = 3, savePredictions = "final", classProb
s = TRUE)
```

Hide

```
# Define
Predictors<-c("M.F", "EDUC", "MMSE", "nWBV") # Supporting variables
Outcome<-'Group' # Target variable
```

BUILD SVM Model: LINEAR KERNEL HYPER-PARAMETER TUNING

In this section, we build SVM model based on linear kernel hyper-parameter. To do this, we apply train() from Caret package:

Hide

```
# Install kernlab
 install.packages("kernlab")
# Load kernlab to apply ksvm
library(kernlab)
# Load gmodels to apply crossTable
library(gmodels)
```

Hide

```
# Apply ksvm to train model on data: linear kernel
svmModel<-train(alzheimers_train[,Predictors], alzheimers_train[,Outcome], method = "
svmLinear", trControl = control, tuneLength = 3)
```

EVALUATE SVM MODEL PERFORMANCE: LINEAR

Hide

```
svmModel
```

```
Support Vector Machines with Linear Kernel

296 samples
  4 predictor
  2 classes: 'N', 'Y'

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 197, 197, 198
Resampling results:

  Accuracy   Kappa
  0.7972583  0.5934046

Tuning parameter 'C' was held constant at a value of 1
```

Since we used train() function, Caret automatically estimate the prediction along with Accuracy and Kappa score.

For the next implementation, we apply ANOVA RBF hyper-parameter tuning with our SVM model from kernlab package and see whether our model accuracy changes:

BUILD SVM WITH KERNLAB: ANOVA KERNEL

Hide

```
# Apply ksvm to train model on data: anova kernel
svmModel_Anova<-ksvm(Group ~., data = alzheimers_train, kernel = "anovadot")
```

```
 Setting default kernel parameters
```

Hide

```
svmModel_Anova
```

```
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1

Anova RBF kernel function.
 Hyperparameter : sigma =  1 degree =  1

Number of Support Vectors : 133

Objective Function Value : -117.7175
Training error : 0.175676
```

EVALUATE SVM MODEL PERFORMANCE: ANOVA RBF

Hide

```
# Predict our KSVM model: ANOVA RBF
svmModel_Anova_Pred<-predict(svmModel_Anova, alzheimers_test)
```

CALCULATE ACCURACY SVM MODEL: ANOVA RBF

Hide

```
CrossTable(alzheimers_test$Group, svmModel_Anova_Pred)
```

```
   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  74


                     | svmModel_Anova_Pred
alzheimers_test$Group |         N |         Y | Row Total |
----------------------|-----------|-----------|-----------|
                    N |        29 |         8 |        37 |
                      |     4.050 |     4.765 |           |
                      |     0.784 |     0.216 |     0.500 |
                      |     0.725 |     0.235 |           |
                      |     0.392 |     0.108 |           |
----------------------|-----------|-----------|-----------|
                    Y |        11 |        26 |        37 |
                      |     4.050 |     4.765 |           |
                      |     0.297 |     0.703 |     0.500 |
                      |     0.275 |     0.765 |           |
                      |     0.149 |     0.351 |           |
----------------------|-----------|-----------|-----------|
         Column Total |        40 |        34 |        74 |
                      |     0.541 |     0.459 |           |
----------------------|-----------|-----------|-----------|
```

Hide

```
confusionMatrix(alzheimers_test$Group, svmModel_Anova_Pred)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  N  Y
         N 29  8
         Y 11 26

               Accuracy : 0.7432
                 95% CI : (0.6284, 0.8378)
    No Information Rate : 0.5405
    P-Value [Acc > NIR] : 0.0002685

                  Kappa : 0.4865

 Mcnemar's Test P-Value : 0.6463552

            Sensitivity : 0.7250
            Specificity : 0.7647
         Pos Pred Value : 0.7838
         Neg Pred Value : 0.7027
             Prevalence : 0.5405
         Detection Rate : 0.3919
   Detection Prevalence : 0.5000
      Balanced Accuracy : 0.7449

       'Positive' Class : N
```

From the result shown above, we can see that the kernlab SVM model was able to predict the out come with lower accuracy of 74.3%. Based on the error rate given from crossTable, kernlab SVM model obtain 25.7%.

To summarize:

Caret SVM = 79.7%
Kernlab SVM = 74.3%

BUILD KNN MODEL

Similar to what have been implemented with SVM, below we build KNN model from Caret and Class package:

TRAIN KNN WITH CARET

Hide

```
# Train kNN
kNNModel<-train(alzheimers_train[,Predictors], alzheimers_train[,Outcome], method = "
knn", trControl = control, tuneLength = 3)
```

EVALUATE KNN WITH CARET

Hide

```
kNNModel
```

```
k-Nearest Neighbors

296 samples
  4 predictor
  2 classes: 'N', 'Y'

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 197, 197, 198
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.7939944  0.5873161
  7  0.7634508  0.5256517
  9  0.7497767  0.4981666

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 5.
```

BUILD KNN CLASS MODEL

Hide

```
# Class Package
library(class)
kNNClass_Model<-knn(train = alzheimers_train[,-1], test = alzheimers_test[,-1], cl =
alzheimers_train[,1], k = 1)
kNNClass_Model
```

```
 [1] Y N Y N Y N N Y N N Y Y Y N Y N Y Y Y N Y Y N N N Y Y N N Y N N N N Y Y Y N N Y
Y Y Y Y N Y Y N N N Y N
[53] N Y Y Y Y Y Y N N N N N N N Y Y Y N N N Y Y
Levels: N Y
```

EVALUATE KNN CLASS MODEL

Hide

```
CrossTable(x = alzheimers_test$Group, y = kNNClass_Model, prop.chisq = FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  74


                     | kNNClass_Model
alzheimers_test$Group |         N |         Y | Row Total |
----------------------|-----------|-----------|-----------|
                    N |        28 |         9 |        37 |
                      |     0.757 |     0.243 |     0.500 |
                      |     0.778 |     0.237 |           |
                      |     0.378 |     0.122 |           |
----------------------|-----------|-----------|-----------|
                    Y |         8 |        29 |        37 |
                      |     0.216 |     0.784 |     0.500 |
                      |     0.222 |     0.763 |           |
                      |     0.108 |     0.392 |           |
----------------------|-----------|-----------|-----------|
         Column Total |        36 |        38 |        74 |
                      |     0.486 |     0.514 |           |
----------------------|-----------|-----------|-----------|
```

Hide

```
confusionMatrix(alzheimers_test$Group, kNNClass_Model)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  N  Y
         N 28  9
         Y  8 29

               Accuracy : 0.7703
                 95% CI : (0.6579, 0.8601)
    No Information Rate : 0.5135
    P-Value [Acc > NIR] : 4.981e-06

                  Kappa : 0.5405

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.7778
            Specificity : 0.7632
         Pos Pred Value : 0.7568
         Neg Pred Value : 0.7838
             Prevalence : 0.4865
         Detection Rate : 0.3784
   Detection Prevalence : 0.5000
      Balanced Accuracy : 0.7705

       'Positive' Class : N
```

Based on both Kernlab and Caret evaluation, Caret was able to obtain higher accuracy with score of 79.4% while Class was able to obtain 77%.

In conclusion, below summarized our findings:

Class k-value = 1 | Accuracy = 77%
Caret k-value = 5 | Accuracy = 79%

Below, we creat a loop based on Class package:

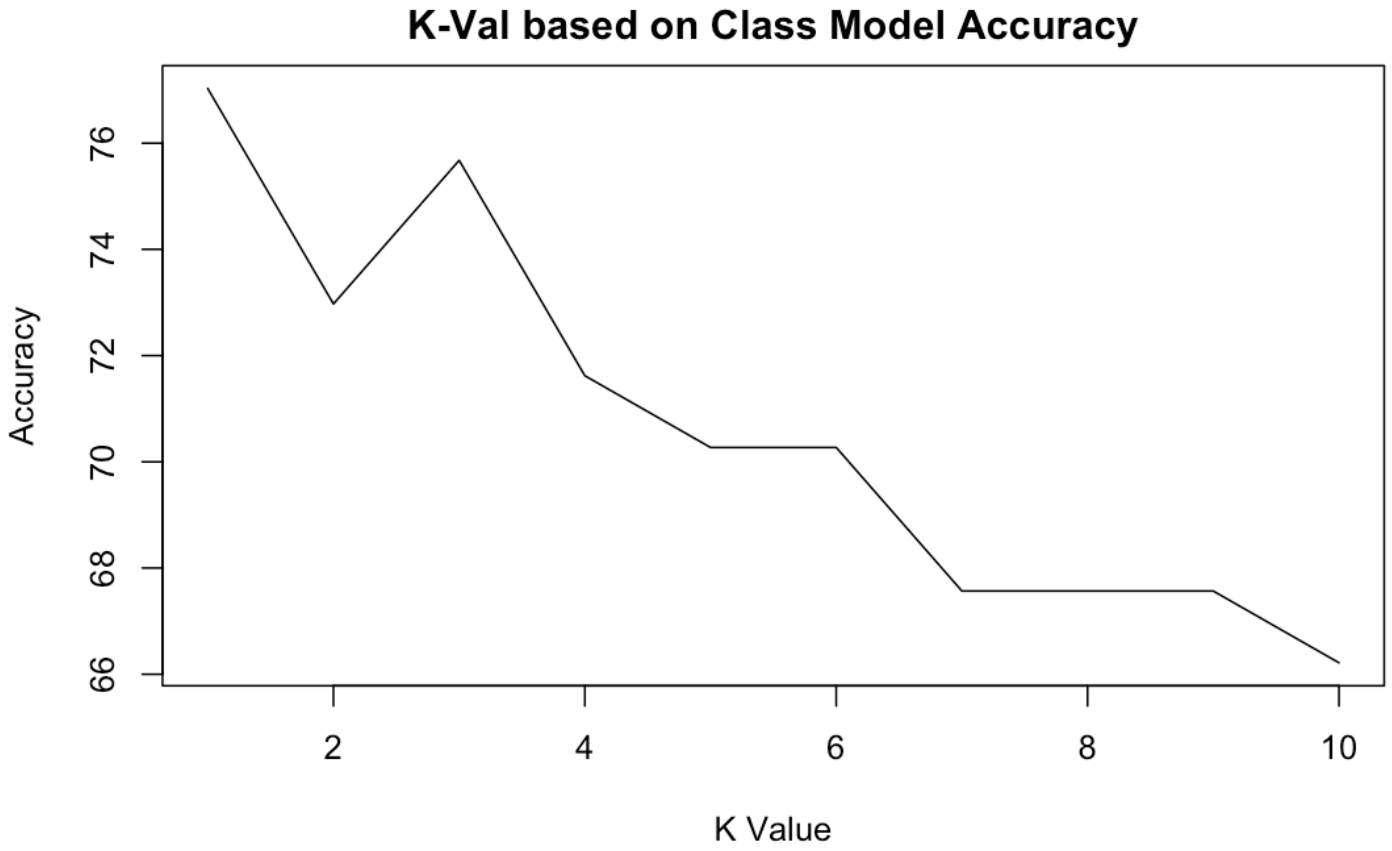
Hide

```
# Create a loop for KNN Model: Accuracy might change when code chunk is re-run. Set.s
eed was implemented but did not make any change.
i<-1                          # Intiate loop
optimal.k<-1                     # Initiate loop
for (i in 1:10){
    knn.model <-  knn(train=alzheimers_train[,-1], test=alzheimers_test[-1], cl = alz
heimers_train[,1], k=i)
    optimal.k[i] <- 100 * sum(alzheimers_test[,1] == knn.model)/NROW(alzheimers_test[
,1])
    k=i
    cat(k,'=',optimal.k[i],'\n')       # Print Accuracy in percentage
}
```

```
1 = 77.02703
2 = 72.97297
3 = 75.67568
4 = 71.62162
5 = 70.27027
6 = 70.27027
7 = 67.56757
8 = 67.56757
9 = 67.56757
10 = 66.21622
```

Hide

```
# Visualize optimal K
plot(optimal.k, type="l", xlab="K Value",ylab="Accuracy", main = "K-Val based on Clas
s Model Accuracy")  # Plot k-value with accuracy
```



As we can see from the observations, Class KNN model was able to obtain the highest accuracy. However, due to the hyper-parameters added to Caret train control, we were able to even obtain higher accuracy than Class Model.

BUILD DECISION TREE C5.0 MODEL

Hide

```
# Train C5 with Caret
C5Model<-train(alzheimers_train[,Predictors], alzheimers_train[,Outcome] ,method = "C
5.0", trControl = control, tuneLength = 3)
```

```
'trials' should be <= 4 for this object. Predictions generated using 4 trials'trials' should be <= 4 for this object. Predictions generated using 4 trials'trials' should be <= 4 for this object. Predictions generated using 4 trials'trials' should be <= 4 for this object. Predictions generated using 4 trials'trials' should be <= 3 for this object. Predictions generated using 3 trials'trials' should be <= 3 for this object. Predictions generated using 3 trials'trials' should be <= 3 for this object. Predictions generated using 3 trials'trials' should be <= 3 for this object. Predictions generated using 3 trials'trials' should be <= 5 for this object. Predictions generated using 5 trials'trials' should be <= 5 for this object. Predictions generated using 5 trials'trials' should be <= 5 for this object. Predictions generated using 5 trials'trials' should be <= 5 for this object. Predictions generated using 5 trials'trials' should be <= 5 for this object. Predictions generated using 5 trials'trials' should be <= 5 for this object. Predictions generated using 5 trials'trials' should be <= 5 for this object. Predictions generated using 5 trials
```

EVALUATE DECISION TREE C5 MODEL

Hide

```
C5Model
```

```
C5.0

296 samples
  4 predictor
  2 classes: 'N', 'Y'

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 198, 197, 197
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.7635195  0.5260977
  rules  FALSE   10      0.7736549  0.5458552
  rules  FALSE   20      0.7702536  0.5386854
  rules   TRUE    1      0.7904899  0.5789423
  rules   TRUE   10      0.7769876  0.5522320
  rules   TRUE   20      0.7769876  0.5522320
  tree   FALSE    1      0.7567168  0.5122807
  tree   FALSE   10      0.7837903  0.5666048
  tree   FALSE   20      0.7837903  0.5662457
  tree    TRUE    1      0.7904899  0.5789423
  tree    TRUE   10      0.7769876  0.5519679
  tree    TRUE   20      0.7769876  0.5519679

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were trials = 1, model = rules and winnow = TRUE.
```

If we compare each of the model' response from train() function, we see that the result is different for every model. As shown above, C5 Decisiion Tree was able to rule out multiple trials, model and winnow to obtain the highest accuracy though not in order.

Hide

```
# Install C5.0
install.packages("C50")
library(C50)
```

Hide

```
# Train Decision Tree Model with C5.0 package
C5Model_OG<-C5.0(alzheimers_train[-1], alzheimers_train$Group)
C5Model_OG
```

```
Call:
C5.0.default(x = alzheimers_train[-1], y = alzheimers_train$Group)

Classification Tree
Number of samples: 296
Number of predictors: 4

Tree size: 2

Non-standard options: attempt to group attributes
```

Based on the result shown above, we can see that we only have 2 decision deep. Here, we apply summary to check the model:

Hide

```
summary(C5Model_OG)
```

```
Call:
C5.0.default(x = alzheimers_train[-1], y = alzheimers_train$Group)


C5.0 [Release 2.07 GPL Edition]     Tue Aug 17 20:17:49 2021
-------------------------------

Class specified by attribute `outcome'

Read 296 cases (5 attributes) from undefined.data

Decision tree:

MMSE <= 0.8977815: Y (100/8)
MMSE > 0.8977815: N (196/54)


Evaluation on training data (296 cases):

        Decision Tree
      ----------------
      Size      Errors

         2   62(20.9%)   <<


       (a)   (b)    <-classified as
      ----  ----
       142     8    (a): class N
        54    92    (b): class Y


    Attribute usage:

    100.00% MMSE


Time: 0.0 secs
```

According to the confusion matrix shown above, the Errors shows the model was unable to classify 62 out of 296 training cases with an error rate of 20.9%.

To summarize this C5.0 model:

False Positives = 8 | False Negatives = 54

EVALUATE OG (ORIGINAL) C5.0 MODEL PERFORMANCE

Hide

```
C5Model_OGPred<-predict(C5Model_OG, alzheimers_test)
```

CHECK ACCURACY OF C5.0 MODEL

Hide

```
CrossTable(alzheimers_test$Group, C5Model_OGPred,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('Actual Group', 'Predicted Group'))
```

```
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  74


             | Predicted Group
Actual Group |         N |         Y | Row Total |
-------------|-----------|-----------|-----------|
           N |        35 |         2 |        37 |
             |     0.473 |     0.027 |           |
-------------|-----------|-----------|-----------|
           Y |        16 |        21 |        37 |
             |     0.216 |     0.284 |           |
-------------|-----------|-----------|-----------|
Column Total |        51 |        23 |        74 |
-------------|-----------|-----------|-----------|
```

From the crosstable shown above, out of 74 observation in the test set, our model accurately predicted 35 patients are nondemented (does not have dementia), and 21 patients are considered to have dementia (demented).
This C5.0 results in 75.7% accurate with an error rate of 24.3%.

IMPROVING C5.0 OG MODEL PERFORMANCE

To improve our C5.0 model, we add trials as an additional boost to better train the model. Here, we specify the trials to 10 due to de facto standard:

Hide

```
C5Model_OGBoost10<-C5.0(alzheimers_train[-1], alzheimers_train$Group,
                      trials = 10)
C5Model_OGBoost10
```

```

Call:
C5.0.default(x = alzheimers_train[-1], y = alzheimers_train$Group, trials = 10)

Classification Tree
Number of samples: 296
Number of predictors: 4

Number of boosting iterations: 10
Average tree size: 2.8

Non-standard options: attempt to group attributes
```

Hide

```
summary(C5Model_OGBoost10)
```

```

Call:
C5.0.default(x = alzheimers_train[-1], y = alzheimers_train$Group, trials = 10)


C5.0 [Release 2.07 GPL Edition]     Tue Aug 17 20:18:56 2021
-------------------------------

Class specified by attribute `outcome'

Read 296 cases (5 attributes) from undefined.data

-----  Trial 0:  -----

Decision tree:

MMSE <= 0.8977815: Y (100/8)
MMSE > 0.8977815: N (196/54)

-----  Trial 1:  -----
```

```
Decision tree:

MMSE <= 0.8461539: Y (58/1.7)
MMSE > 0.8461539:
:...M.F <= 0: Y (107.6/42.7)
    M.F > 0: N (130.4/45.4)

-----  Trial 2:  -----

Decision tree:

MMSE <= 0.8461539: Y (50.4/2.1)
MMSE > 0.8461539:
:...MMSE <= 0.9230769: Y (80.4/30.5)
    MMSE > 0.9230769: N (165.2/63.1)

-----  Trial 3:  -----

Decision tree:

MMSE <= 0.8076923: Y (32.9)
MMSE > 0.8076923:
:...nWBV <= 0.4611399: Y (137.2/49.9)
    nWBV > 0.4611399: N (125.9/43.4)

-----  Trial 4:  -----

Decision tree:

MMSE <= 0.8461539: Y (40.6/2.1)
MMSE > 0.8461539: N (255.4/118.8)

-----  Trial 5:  -----

Decision tree:

MMSE <= 0.8076923: Y (26.3)
MMSE > 0.8076923:
:...EDUC <= 0.7142857: Y (205.8/89.1)
    EDUC > 0.7142857: N (63.9/24.6)

-----  Trial 6:  -----

Decision tree:

MMSE <= 0.8076923: Y (23.8)
```

```
MMSE > 0.8076923:
:...MMSE <= 0.8977815: Y (47.6/17.2)
    MMSE > 0.8977815: N (224.6/102.3)

-----  Trial 7:  -----

Decision tree:

MMSE <= 0.8461539: Y (30.9)
MMSE > 0.8461539:
:...nWBV > 0.7202073: N (18.3/0.9)
    nWBV <= 0.7202073:
    :...M.F <= 0: Y (126.1/51.2)
        M.F > 0: N (110.7/49.6)

-----  Trial 8:  -----

Decision tree:

MMSE <= 0.8977815: Y (63.5/6.5)
MMSE > 0.8977815: N (207.5/71.2)

-----  Trial 9:  -----

Decision tree:

MMSE <= 0.8977815: Y (50)
MMSE > 0.8977815:
:...M.F <= 0: Y (150.5/65.2)
    M.F > 0: N (54.5)


Evaluation on training data (296 cases):

Trial       Decision Tree
-----     ----------------
      Size      Errors

   0         2   62(20.9%)
   1         3   81(27.4%)
   2         3   66(22.3%)
   3         3   83(28.0%)
   4         2   78(26.4%)
   5         3  123(41.6%)
   6         3   62(20.9%)
   7         4   77(26.0%)
   8         2   62(20.9%)
```

```
   9         3   71(24.0%)
boost            63(21.3%)   <<


       (a)   (b)    <-classified as
      ----  ----
       138    12    (a): class N
        51    95    (b): class Y


    Attribute usage:

    100.00% MMSE
     81.76% EDUC
     81.76% nWBV
     76.35% M.F


Time: 0.0 secs
```

Hide

```
C5Model_OGBoostPred10 <- predict(C5Model_OGBoost10, alzheimers_test)
CrossTable(alzheimers_test$Group, C5Model_OGBoostPred10,
            prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
            dnn = c('Actual Group', 'Predicted Group'))
```

```
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  74


             | Predicted Group
Actual Group |         N |         Y | Row Total |
-------------|-----------|-----------|-----------|
           N |        34 |         3 |        37 |
             |     0.459 |     0.041 |           |
-------------|-----------|-----------|-----------|
           Y |        13 |        24 |        37 |
             |     0.176 |     0.324 |           |
-------------|-----------|-----------|-----------|
Column Total |        47 |        27 |        74 |
-------------|-----------|-----------|-----------|
```

To summarize the improvement:

- C5.0 Model 1 Error Rate = 24.3%
- C5.0 Boosted Model 1 Error Rate = 21.7%

Our boosted model has reduced the error rate to 21.7%.

To summarize the C5.0 model observations:

Caret Accuracy = 79%
C5 Accuracy = 78.3%

To sum up, the 3 models considering SVM, KNN and C5.0, Caret seems to play an essential role due to its hyperparameters in increasing the models' productivity. Before we proceed to ensemble learning, we implement another model with Random Forest:

BUILD MODEL WITH RANDOMFOREST

In this section, we implement ensemble model to see whether the ensembled model create higher accuracy compared to the models done previously.

Hide

```
# Installl Random Forest
install.packages("randomForest")
# Load Random Forest
library(randomForest)
```

Hide

```
# Build Random Forest Model: Group as target variable
RFModel_OG<-randomForest(x = alzheimers_train[-1], y = alzheimers_train$Group, ntree
= 200, random_state = 0)
```

EVALUATE RANDOM FOREST MODEL

Hide

```
RFPredict_Group<-predict(RFModel_OG, newdata = alzheimers_test[-1])
```

CHECK ACCURACY RANDOM FOREST MODEL

Hide

```
CrossTable(alzheimers_test$Group, RFPredict_Group,
             prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
             dnn = c('Actual Group', 'Predicted Group'))
```

```
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  74


             | Predicted Group
Actual Group |         N |         Y | Row Total |
-------------|-----------|-----------|-----------|
           N |        28 |         9 |        37 |
             |     0.378 |     0.122 |           |
-------------|-----------|-----------|-----------|
           Y |        10 |        27 |        37 |
             |     0.135 |     0.365 |           |
-------------|-----------|-----------|-----------|
Column Total |        38 |        36 |        74 |
-------------|-----------|-----------|-----------|
```

Hide

```
confusionMatrix(RFPredict_Group, alzheimers_test$Group)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  N  Y
         N 28 10
         Y  9 27

               Accuracy : 0.7432
                 95% CI : (0.6284, 0.8378)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 1.688e-05

                  Kappa : 0.4865

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.7568
            Specificity : 0.7297
         Pos Pred Value : 0.7368
         Neg Pred Value : 0.7500
             Prevalence : 0.5000
         Detection Rate : 0.3784
   Detection Prevalence : 0.5135
      Balanced Accuracy : 0.7432

       'Positive' Class : N
```

Based on the ensemble concept of Random Forest, we were able to obtain a fairly strong mode with 74.3% accuracy and an error rate of 25.7%.

Since the author decides to apply Random Forest from Caret as an ensemble stacking of the ensemble model. We proceed to calculate the 3 models probabilities shown below:

ENSEMBLE LEARNING: COMBINE MULTIPLE MODELS

Hide

```
library(caTools)
library(caretEnsemble)
```

Hide

```
set.seed(123)
# Correlation
ModResults<-resamples(list(svmModel, kNNModel, C5Model))
modelCor(ModResults)
```

```
          Model1      Model2      Model3
Model1 1.0000000  0.77907398  0.56928872
Model2 0.7790740  1.00000000 -0.07190639
Model3 0.5692887 -0.07190639  1.00000000
```

Hide

```
# Predict Fold Prediction Probabilities
alzheimers_train$OOF_SVM<-svmModel$pred$Y[order(svmModel$pred$rowIndex)]
alzheimers_train$OOF_kNN<-kNNModel$pred$Y[order(kNNModel$pred$rowIndex)]
alzheimers_train$OOF_C5<-C5Model$pred$Y[order(C5Model$pred$rowIndex)]
```

Hide

```
#Predicting probabilities for the test data
alzheimers_test$OOF_SVM<-predict(svmModel,alzheimers_test[Predictors],type='prob')$Y
alzheimers_test$OOF_kNN<-predict(kNNModel,alzheimers_test[Predictors],type='prob')$Y
alzheimers_test$OOF_C5<-predict(C5Model,alzheimers_test[Predictors],type='prob')$Y
```

Hide

```
# Build Top Layers
top_layers<-c('OOF_SVM','OOF_kNN','OOF_C5')
```

Hide

```
# Apply Ensemble Stacking through Random Forest
RFModel<- train(alzheimers_train[,top_layers],alzheimers_train[,Outcome],method='rf',
trControl=control,tuneLength=2)
```

Hide

```
# Predict with ensemble stacking: RF top layer model
alzheimers_test$rf_stacked<-predict(RFModel,alzheimers_test[,top_layers])
```

CHECK ACCURACY RANDOM FOREST ENSEMBLE

Hide

```
CrossTable(alzheimers_test$Group, alzheimers_test$rf_stacked,
             prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
             dnn = c('Actual Group', 'Predicted Group'))
```

```
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  74


             | Predicted Group
Actual Group |         N |         Y | Row Total |
-------------|-----------|-----------|-----------|
           N |        30 |         7 |        37 |
             |     0.405 |     0.095 |           |
-------------|-----------|-----------|-----------|
           Y |        11 |        26 |        37 |
             |     0.149 |     0.351 |           |
-------------|-----------|-----------|-----------|
Column Total |        41 |        33 |        74 |
-------------|-----------|-----------|-----------|
```

Hide

```
confusionMatrix(alzheimers_test$rf_stacked, alzheimers_test$Group)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  N  Y
         N 30 11
         Y  7 26

               Accuracy : 0.7568
                 95% CI : (0.6431, 0.849)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 5.546e-06

                  Kappa : 0.5135

 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 0.8108
            Specificity : 0.7027
         Pos Pred Value : 0.7317
         Neg Pred Value : 0.7879
             Prevalence : 0.5000
         Detection Rate : 0.4054
   Detection Prevalence : 0.5541
      Balanced Accuracy : 0.7568

       'Positive' Class : N
```

Based on our stacked model above, we combined 3 models (SVM, KNN, C5.0) that are based on Caret package due to their high accuracy and identical train control hyper-parameters adjustments with a Random Forest model.

According to the result, ensemble stacked model is surprisingly lower in accuracy compared to the 3 models individually.

Notice that within this project, the author did not specify MAE or MSE. This is because the models are based on classification, and most evaluation metrics such as RMSE, MSE, and MAE are better off with regression models. The author tried evaluating MSE for each loop with KNN model, but due to the nature of the classification based data, level of k-value does not correlate efficiently with the MSE where the higher the accuracy, the lower MSE should be.

Instead, the author specify error rate by summing up False Positive and False Negative by the end of the crossTable evaluation.