2021

Advanced Machine Learning

Module number: DALT7012

Student Number: 19132761

MSc Course: Data Analytics

Word count: 2351

Table of Contents

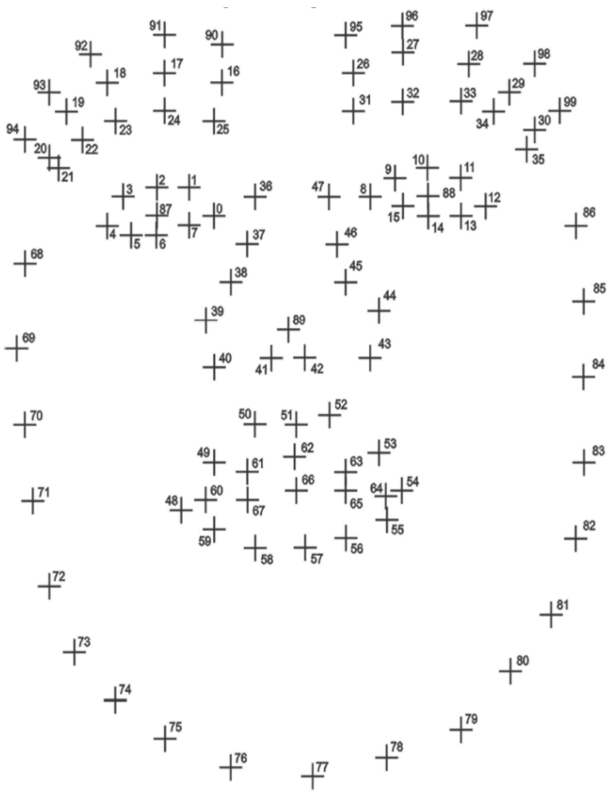
1. Description of the dataset

The data set was developed by Freitas et al. for facial recognition is used in the Brazilian sign language "Libras" and to generate a grammar structure based on machine learning for clear interpretation.

Using Microsoft Kinect sensor, Freitas et al. recorded nine categorical videos for each of two person “a” and “b” (total 18 videos) and divided the data in time frames. The data was tabulated as follows:

- an image of each frame, identified by timestamps.
- a text file containing one hundred coordinates (x, y, z) of points from eyes, nose, eyebrows etc., each line in the data corresponds to points extracted from one frame.

The dataset is organized into 36 files: 18 data point files and 18 target files, one pair for each video composes the dataset. The file’s name refers to each video: the letter corresponding to the person A and person B, the names of grammatical facial expressions and data points or target. Total 100 points on face located using coordinates x and y (given in pixels) and z (in millimeters) (Freitas et al., 2014). The 100 points are marked serially as shown below.



2. Data Preparation

In this coursework, Topics and Conditional facial expressions of person A and person B datasets are chose to do classification.

These data preparation steps would be repeated in Conditional facial expression of person A, Topics and Conditional facial expressions of person B.

Let explore an example of the topics dataset on person A:

```
# Set the working directory
setwd("/grammatical_facial_expression/")

# Load the User A topics data
data1A <- read.delim("a_topics_datapoints.txt", header=TRUE, sep=" ")
target1A <- read.delim("a_topics_targets.txt", header=FALSE)
```

```r
# Check the real number of positives and negatives in the topics dataset person A
dim(data1A)
```

```
## [1] 1796  301
```

```r
dim(target1A)
```

```
## [1] 1796    1
```

```r
table(target1A)
```

```
## target1A
##    0    1
## 1436  360
```

The dataset has 360 (+) frames and 1436 (-) frames, 1796 observations and 301 variables. However, the first column is the timestamp, so we remove it and combine data point and target into a data frame.

```r
# Add the labels (target dataset) onto the data frames
data1A <- cbind(data1A, target1A)

# Remove the first column
data1A <- select(data1A, -X0.0)
```

Split the data into training (75%) & testing datasets (25%) to train SVM on person A.

```r
# Split the data into a training & testing dataset
set.seed(123)
split1A <- sample.split(data1A$V1, SplitRatio = 0.75)
train1A <- subset(data1A, split1A == TRUE)
test1A <- subset(data1A, split1A == FALSE)
```

Because the dataset is not on the same scale, we need to run feature scaling:

```r
# Feature scaling
train1A[,-301] <- scale(train1A[,-301])
test1A[,-301] <- scale(test1A[,-301])
```

3. SVM – Soft Vector Machine on person A – 22%

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labelled training data for each category, they can categorize new text. SVM works very well on smaller data sets or high dimensional spaces. (TopBlog, n.d.)

In the SVM algorithm, we plot each data item as a point in 3-dimensional space, with each feature's value being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes. (B et al., 2008)

a(i) SVM on person A, topics facial expression – 10%

Fitting SVM to the train set and predict results. The number of features is large, in this case, we will use the linear kernel because it is more likely that the data is linearly separable in high dimensional space. (Ray, 2017)

To find the best parameters for the SVM model, Grid Search is applied:

```r
# Applying Grid Search to find the best parameters
svm1A = train(form = V1 ~ . ,
                   data = train1A,
                   method = "svmLinear")
```

```r
# Show the best parameters
svm1A$bestTune
```

```
##   C
## 1 1
```

The best tune is cost = 1. We do cross-validation for its parameters to avoid over-fitting.

```r
# Fitting SVM to the train set 1A
svm1A <- svm(formula = V1~.,
                  data = train1A,
                  type = "C-classification",
                  kernel = "linear",
                  cost = 1,
                  cross = 10)

# Predicting the test set results
predict1A <- predict(svm1A, newdata = test1A[-301])

# Making the Confusion Matrix
confusionMatrix(factor(predict1A), factor(test1A[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 351  10
##          1   8  80
##
##                Accuracy : 0.9599
##                  95% CI : (0.9374, 0.9761)
##     No Information Rate : 0.7996
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8739
##
##  Mcnemar's Test P-Value : 0.8137
##
##             Sensitivity : 0.9777
##             Specificity : 0.8889
##          Pos Pred Value : 0.9723
##          Neg Pred Value : 0.9091
##              Prevalence : 0.7996
##          Detection Rate : 0.7817
##    Detection Prevalence : 0.8040
##       Balanced Accuracy : 0.9333
##
##        'Positive' Class : 0
##
```

We got an accuracy of 95.99% for training SVM on train set and testing on test set of person A topics facial expression. The SVM classifier model seems to present very well.

a(ii) SVM on person A, conditional facial expression – 4%

```r
# Check the real number of positives and negatives in the dataset
dim(data2A)
```

```
## [1] 1907  301
```

```r
table(target2A)
```

```
## target2A
##    0    1
## 1359  548
```

The conditional facial expression dataset of person A has 548 (+) frames and 1359 (-) frames, 1907 observations and also 301 variables.

By applying Grid Search, we have the best tune: cost = 1. Fitting SVM to the train set conditional person A, we got the accuracy of 97.06% for testing on test set for person A conditional expression.

```r
# Making the Confusion Matrix
confusionMatrix(factor(predict2A), factor(test2A[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 335   9
##          1   5 128
##
##                Accuracy : 0.9706
##                  95% CI : (0.9512, 0.9839)
##     No Information Rate : 0.7128
##     P-Value [Acc > NIR] : <2e-16
```

a(iii) SVM implementation – 8%

```r
# SVM implementation
svm.fit = function(X, y, C=NULL) {
  n.samples = nrow(X)
  n.features = ncol(X)
  K = matrix(rep(0, n.samples*n.samples), nrow=n.samples)
  for (i in 1:n.samples){
    for (j in 1:n.samples){
      K[i,j] = X[i,] %*% X[j,] }}
  Dmat = outer(y,y) * K
  Dmat = as.matrix(nearPD(Dmat)$mat)
  dvec = rep(1, n.samples)
  Amat = rbind(y, diag(n.samples), -1*diag(n.samples))
  bvec = c(0, rep(0, n.samples), rep(-C, n.samples))
  res = solve.QP(Dmat,dvec,t(Amat),bvec=bvec, meq=1)
  a = res$solution
  bomega = apply(a*y*X,2,sum)
  return(bomega)
}
```

4. Test SVM on person B – 20%

b(i) Test SVM on person B, topics facial expression – 10%

In this part, we will use the SVM model, which is trained on person A, to test on person B in the same facial expression – topics dataset.

```r
# Check the real number of positives and negatives
dim(data1B)
```

```
## [1] 1825  301
```

```r
table(target1B)
```

```
## target1B
##    0    1
## 1358  467
```

The topics facial expression dataset of person B has 467 (+) frames and 1358 (-) frames, 1825 observations and also 301 variables.

```r
# Predicting the results
predict1B <- predict(svm1A, newdata = data1B[-301])

# Making the Confusion Matrix
confusionMatrix(predict1B, factor(data1B[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1039  153
##          1  319  314
##
##                Accuracy : 0.7414
##                  95% CI : (0.7206, 0.7613)
##     No Information Rate : 0.7441
##     P-Value [Acc > NIR] : 0.6175
```

We got only 74.14% accuracy on testing on person B, which is not high accuracy as expected.

b(ii) Test SVM on person B, conditional facial expression – 5%

Using the SVM model, which is trained on person A, to test on person B in the same facial expression – conditional dataset.

```r
# Check the real number of positives and negatives
dim(data2B)
```

```
## [1] 2034  301
```

```r
table(target2B)
```

```
## target2B
##    0    1
## 1445  589
```

The conditional facial expression dataset of person B has 589 (+) frames and 1445 (-) frames, 2034 observations and also 301 variables.

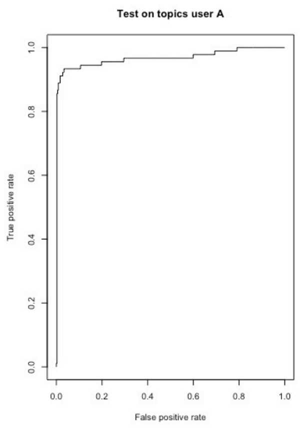
```r
# Predicting the results
predict2B <- predict(svm2A, newdata = data2B[-301])

# Making the Confusion Matrix
confusionMatrix(predict2B, factor(data2B[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1445  589
##          1    0    0
##
##                Accuracy : 0.7104
##                  95% CI : (0.6902, 0.7301)
##     No Information Rate : 0.7104
##     P-Value [Acc > NIR] : 0.5111
```

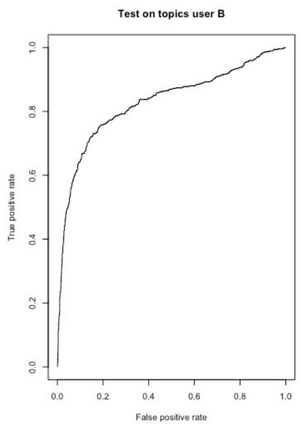
We got similarly 71.04% accuracy. From the table of prediction, SVM classifies almost everything to 0.

b(iii) ROC curve for SVM – 5%

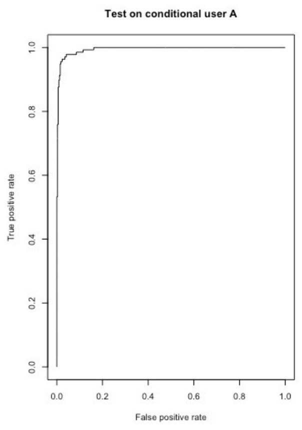
These accuracy numbers cannot say clearly that the model is bad. Therefore, we plot ROC curve for SVM:



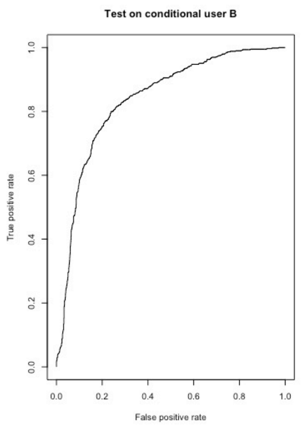
auc = 96.75%



auc = 77.03%



auc = 99.42%



auc = 83.74%

The auc for testing on topics person B data is 77.03%, for testing on conditional person B is 83.74%. These are good classifications.

5. Repeat the analysis by inverting the roles of person A and B – 8%

c(i) SVM on person B then test on person A – 4%

```r
# Fitting SVM to the train set
svm3B <- svm(formula = V1~.,
             data = train3B,
             type = "C-classification",
             kernel = "linear",
             cost = 1,
             cross = 10)
```

We got 89.72% accuracy for train and test on person B, topics dataset.

```r
# Predicting the test set results
predict3B <- predict(svm3B, newdata = test3B[-301])

# Making the Confusion Matrix
confusionMatrix(factor(predict3B), factor(test3B[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 325  32
##          1  15  85
##
##                Accuracy : 0.8972
##                  95% CI : (0.8656, 0.9235)
##     No Information Rate : 0.744
##     P-Value [Acc > NIR] : <2e-16
```

Use the SVM model topics dataset, which is trained on person B, to test on person A, we got 71.88% accuracy. That is lower than 74.14% of training on A and testing on B.

```r
# Predicting the results
predict3A <- predict(svm3B, newdata = data3A[-301])

# Making the Confusion Matrix
confusionMatrix(predict3A, factor(data3A[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1213  282
##          1  223   78
##
##                Accuracy : 0.7188
##                  95% CI : (0.6974, 0.7395)
##     No Information Rate : 0.7996
##     P-Value [Acc > NIR] : 1.000000
```

Doing the same with conditional facial expression, we got 91.34% training and testing on person B, 63.92% when training SVM on person B and testing on person A. It is also lower than 71.04% of training on A and testing on B.

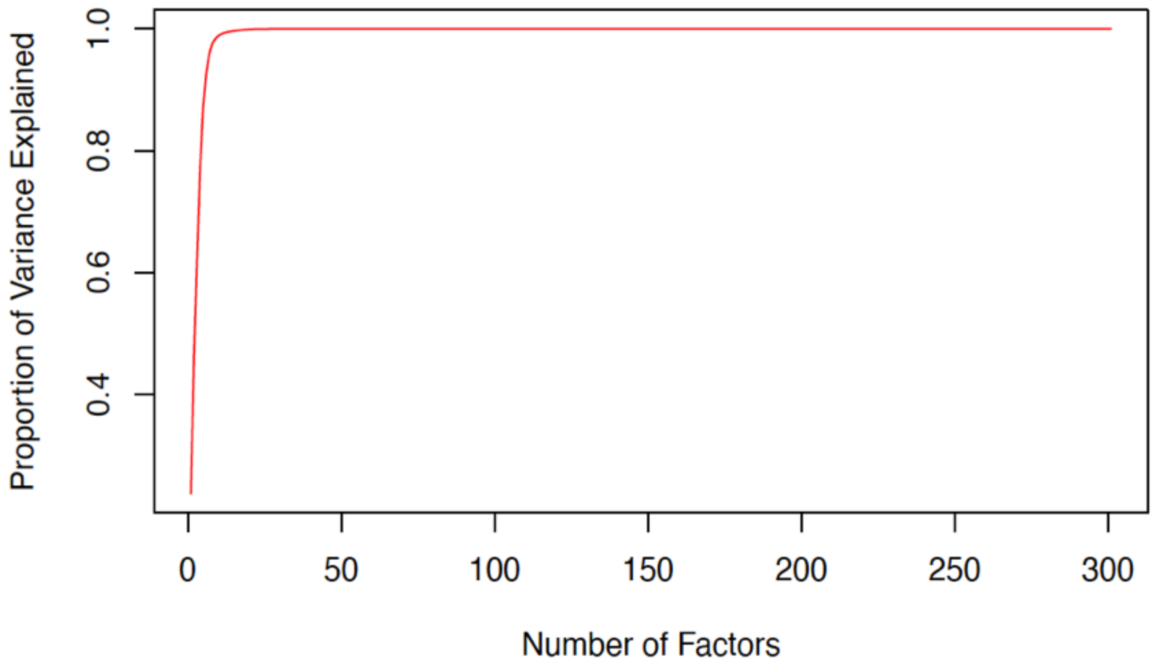
```r
# Predicting the test set results
predict4B <- predict(svm4B, newdata = test4B[-301])

# Making the Confusion Matrix
confusionMatrix(factor(predict4B), factor(test4B[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 341  24
##          1  20 123
##
##                Accuracy : 0.9134
##                  95% CI : (0.8855, 0.9364)
##     No Information Rate : 0.7106
##     P-Value [Acc > NIR] : <2e-16
```

```r
# Predicting the results
predict4A <- predict(svm4B, newdata = data4A[-301])

# Making the Confusion Matrix
confusionMatrix(predict4A, factor(data4A[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1061  390
##          1  298  158
##
##                Accuracy : 0.6392
##                  95% CI : (0.6172, 0.6608)
##     No Information Rate : 0.7126
##     P-Value [Acc > NIR] : 1.0000000
```

c(ii) PCA and SVM on person A then test on person B topics facial expression – 4%



15 PCAs can explain approximately 300 PCAs but can speed up the SVM model pretty much.

```
# Look at 15th variance
variance_explained1A[15, ]
```

```
##    NmbrPCs    CumVar
## 15      15 0.9965087
```

```
# Do PCA for creating 15 factors
pca1A <- preProcess(x = train1A[-301], method = "pca", pcaComp = 15)
train_pca1A <- predict(pca1A, train1A)
test_pca1A <- predict(pca1A, test1A)
str(train_pca1A[1:5,])
```

```
## 'data.frame':    5 obs. of  16 variables:
##  $ V1  : int  0 0 0 0 0
##  $ PC1 : num  -103.43 -72.91 -19.16 -12.66 -5.55
##  $ PC2 : num  -182.1 -75.5 12.5 17.7 20.6
```

After PCA, we apply SVM and test on person A topics facial expression. we got the accuracy of 94.21%, which is slightly lower than 95.99% when run the SVM without PCA.

```
# Predicting the test set results
pred_pca1A <-  predict(svm_pca1A, newdata = test_pca1A[-301])

# Making the confusion matrix
confusionMatrix(factor(pred_pca1A), factor(test_pca1A$V1))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 352  19
##          1   7  71
##
##                Accuracy : 0.9421
##                  95% CI : (0.9163, 0.9618)
##     No Information Rate : 0.7996
##     P-Value [Acc > NIR] : < 2e-16
```

Apply PCA and SVM model's A for person B, we got accuracy 87.18%, which is higher than 74.14% when run the SVM without PCA.

```r
# Predicting the results
pred_pca1B <-  predict(svm_pca1A, newdata = data_set1B[-301])

# Making the confusion matrix
confusionMatrix(factor(pred_pca1B), factor(data_set1B$V1))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1222   98
##          1  136  369
##
##                Accuracy : 0.8718
##                  95% CI : (0.8556, 0.8868)
##     No Information Rate : 0.7441
##     P-Value [Acc > NIR] : < 2e-16
```

Doing the same with conditional facial expression, PCA and SVM on train and test set of person A, we got 97.06%. Apply PCA and SVM model's A for person B, we got accuracy 71.04%, similar when run the SVM without PCA.

```r
# Predicting the test set results
pred_pca2A <-  predict(svm_pca2A, newdata = test_pca2A[-301])

# Making the confusion matrix
confusionMatrix(factor(pred_pca2A), factor(test_pca2A$V1))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 336  12
##          1   4 125
##
##                Accuracy : 0.9665
##                  95% CI : (0.9461, 0.9807)
##     No Information Rate : 0.7128
##     P-Value [Acc > NIR] : < 2e-16
```

```r
# Predicting the results
pred_pca2B <-  predict(svm_pca2A, newdata = data_set2B[-301])

# Making the confusion matrix
confusionMatrix(pred_pca2B, factor(data_set2B$V1))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1445  589
##          1    0    0
##
##                Accuracy : 0.7104
##                  95% CI : (0.6902, 0.7301)
##     No Information Rate : 0.7104
##     P-Value [Acc > NIR] : 0.5111
```

6. Random Forest – 30%

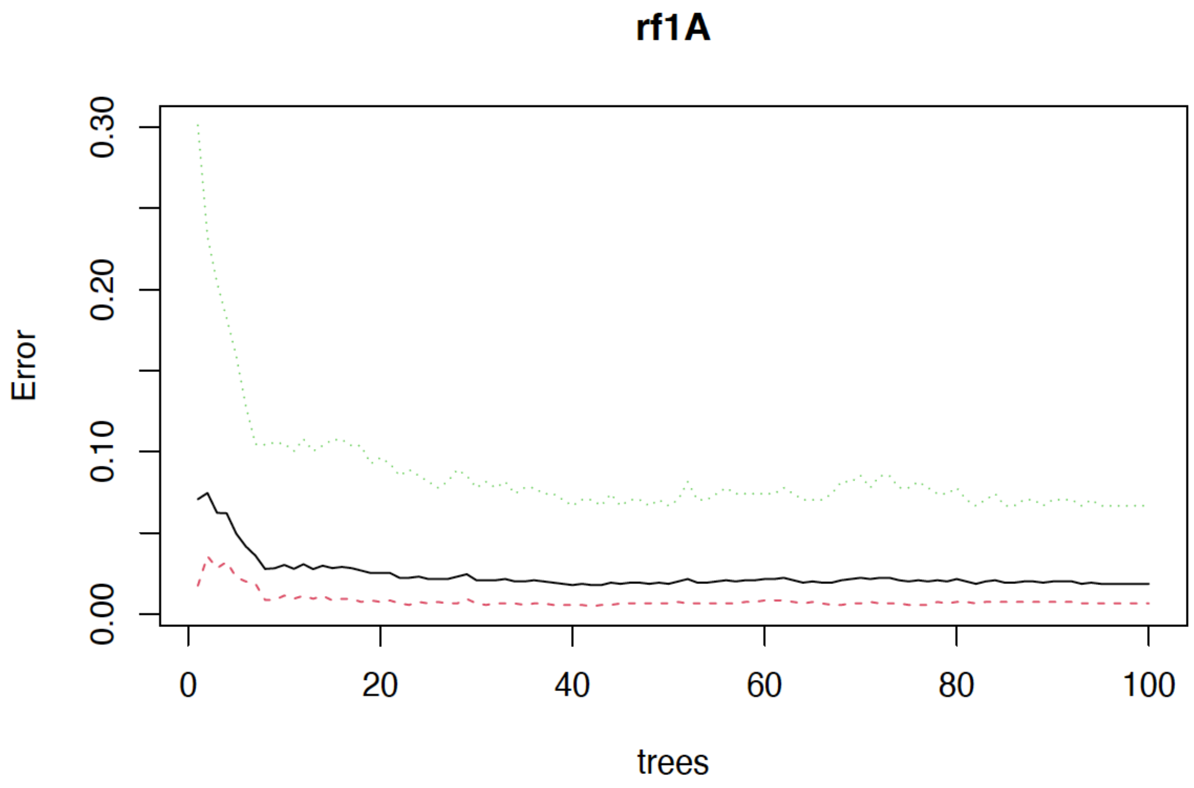
Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithms of the same type i.e., multiple decision trees, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks. (B et al., 2019)

The key principle underlying the random forest approach comprises the construction of many "simple" decisions trees in the training stage and the majority vote (mode) across them in the classification stage (Nigri and Arandjelovic´, n.d.). In the training stage, random forests apply the general technique known as bagging to individual trees in the ensemble. Bagging repeatedly selects a random sample with replacement from the training set and fits trees to these samples. Each tree is grown without any pruning. The number of trees in the ensemble is a free parameter that is readily learned automatically using the so-called out-of-bag error. (D., Dimitriou and Arandjelović, 2021)

d(i) Random Forest on person A, topics facial expression – 8%

Running random forest with numTrain = 500 and numTree = 100 on train dataset topics dataset person A.

```r
# Random Forest
set.seed(123)

# Generate a random sample of "numTrain" indexes
rows1A <- sample(1:nrow(train1A), 500)

# Random forest model
rf1A <- randomForest(train1A, factor(train1A[,301]), ntree=100)
```



```r
# Make prediction
predict1A <- predict(rf1A, test1A)

# Confusion Matrix
confusionMatrix(predict1A, factor(test1A[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 358   8
##          1   1  82
##
##                Accuracy : 0.98
##                  95% CI : (0.9623, 0.9908)
##     No Information Rate : 0.7996
##     P-Value [Acc > NIR] : <2e-16
```

We got the accuracy of 98% of training and testing on the same person A, topics dataset. It is higher than SVM 95.99% accuracy.

d(ii) Test Random Forest on person B, topics facial expression – 4%

Use the random forest model, which is trained on person A, to test on person B. We got good accuracy at 93.86%. Which is significantly higher than by SVM classifier with only 74.14% accuracy.

```r
# Predicting
predict1B <- predict(rf1A, data1B)

# Making the Confusion Matrix
confusionMatrix(predict1B, factor(data1B[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1295   49
##          1   63  418
##
##                Accuracy : 0.9386
##                  95% CI : (0.9266, 0.9492)
##     No Information Rate : 0.7441
##     P-Value [Acc > NIR] : <2e-16
```

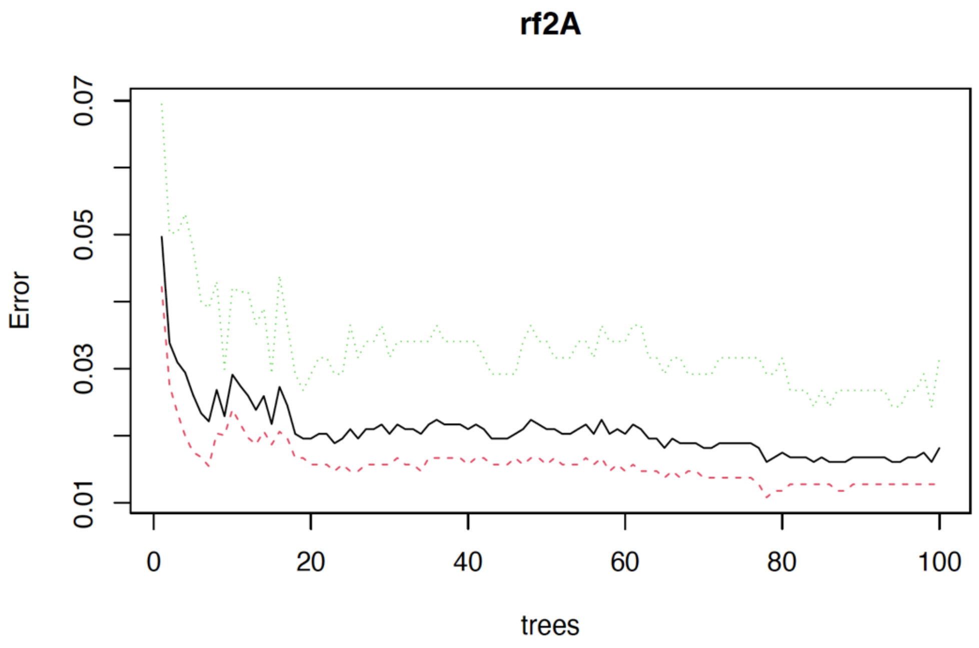
d(iii) Random Forest implementation – 3%

```r
##### d(iii) Own implementation of the second classifier ####
random_forest <- function(train_data, train_formula, method="class",
                          feature_per=0.7, cp=0.01, min_split=20,
                          min_bucket=round(min_split/3),
                          max_depth=30, ntrees = 10)
  {
  target_variable <- as.character(train_formula)[[2]]
  features <- setdiff(colnames(train_data), target_variable)
  n_features <- length(features)

  ncores <- detectCores(logical=FALSE)
  cl <- makeCluster(ncores)
  registerDoParallel(cl)

  rf_model <- foreach(
    icount(ntrees),
    .packages = c("rpart", "Metrics")
  ) %dopar% {
    bagged_features <- sample(features, n_features * feature_per, replace = FALSE)
    index_bag <- sample(nrow(train_data), replace=TRUE)
    in_train_bag <- train_data[index_bag,]
    out_train_bag <- train_data[-index_bag,]
    trControl <- rpart.control(minsplit = min_split, minbucket = min_bucket,
                               cp = cp, maxdepth = max_depth)
    tree <- rpart(formula = train_formula,
                  data = in_train_bag,
                  control = trControl)

    oob_pred <- predict(tree, newdata = out_train_bag, type = "class")
    oob_acc <- accuracy(actual = out_train_bag[, target_variable],
                        predicted = oob_pred)

    list(tree=tree, oob_perf=oob_acc)
  }
  stopCluster(cl)
  rf_model
}
```

d(iv) Random Forest on person A, conditional facial expression – 3%

Running random forest with numTrain = 500 and numTree = 100 on train dataset conditional person A.

```r
## Random Forest
set.seed(123)

# Generate a random sample of "numTrain" indexes
rows2A <- sample(1:nrow(train2A), 500)

# Random forest model
rf2A <- randomForest(train2A, factor(train2A[,301]), ntree=100)

# Plot
plot(rf2A)
```



```r
# Make prediction
predict2A <- predict(rf2A, test2A)

# Confusion Matrix
confusionMatrix(predict2A, factor(test2A[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 339   3
##          1   1 134
##
##                Accuracy : 0.9916
##                  95% CI : (0.9787, 0.9977)
##     No Information Rate : 0.7128
##     P-Value [Acc > NIR] : <2e-16
```

We got excellent the accuracy of 99.16% for the conditional dataset on person A.

d(v) Test Random Forest on person B, conditional facial expression – 3%

```
# Predicting
predict2B <- predict(rf2A, data2B)

# Making the Confusion Matrix
confusionMatrix(predict2B, factor(data2B[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1410  135
##          1   35  454
##
##                Accuracy : 0.9164
##                  95% CI : (0.9035, 0.9281)
##     No Information Rate : 0.7104
##     P-Value [Acc > NIR] : < 2.2e-16
```

Use the random forest model, which is trained on person A conditional dataset, to test on person B. We got the accuracy of 91.64%, which is significantly higher than by SVM classifier 71.04% accuracy.

d(vi) Training Random Forest on person B and testing on person A – 3%

Use random forest model to train and test on person B topics dataset, we got 97.16% accuracy.

```r
# Make prediction
predict3B <- predict(rf3B, test3B)

# Confusion Matrix
confusionMatrix(predict3B, factor(test3B[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 337  10
##          1   3 107
##
##                Accuracy : 0.9716
##                  95% CI : (0.9518, 0.9848)
##     No Information Rate : 0.744
##     P-Value [Acc > NIR] : < 2e-16
```

We got 96.1% accuracy when training on person B and testing on person A with topics dataset facial expression. Pretty similar with training on A and testing on B accuracy 93.86%.

```r
# Predicting the results
predict3A <- predict(rf3B, data3A)

# Making the Confusion Matrix
confusionMatrix(predict3A, factor(data3A[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1423   57
##          1   13  303
##
##                Accuracy : 0.961
##                  95% CI : (0.951, 0.9695)
##     No Information Rate : 0.7996
##     P-Value [Acc > NIR] : < 2.2e-16
```

With conditional dataset facial expression, we got 97.64% accuracy when training and testing on B, 94.65% accuracy when train on person B and test on person A. Slightly higher than training on person A and testing on person B the accuracy of 91.64%.

```r
# Make prediction
predict4B <- predict(rf4B, test4B)

# Confusion Matrix
confusionMatrix(predict4B, factor(test4B[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 361  12
##          1   0 135
##
##                Accuracy : 0.9764
##                  95% CI : (0.9591, 0.9877)
##     No Information Rate : 0.7106
##     P-Value [Acc > NIR] : < 2.2e-16
```

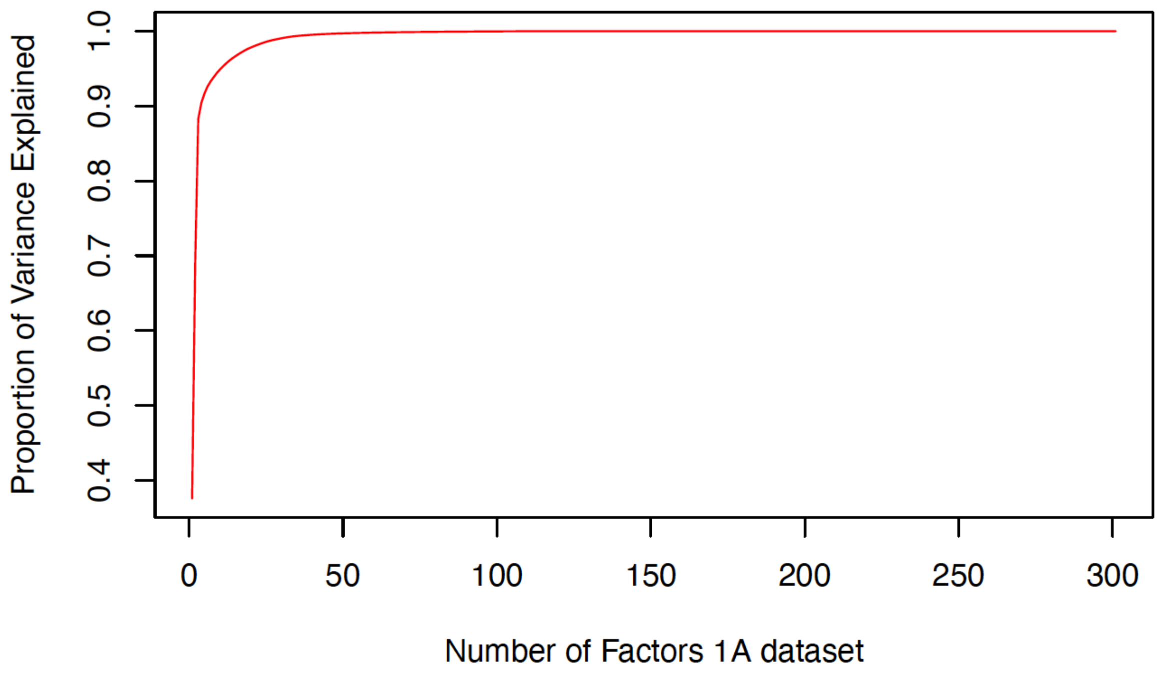
```r
# Predicting
predict4A <- predict(rf4B, data4A)

# Making the Confusion Matrix
confusionMatrix(predict4A, factor(data4A[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1359  102
##          1    0  446
##
##                Accuracy : 0.9465
##                  95% CI : (0.9354, 0.9562)
##     No Information Rate : 0.7126
##     P-Value [Acc > NIR] : < 2.2e-16
```

d(vii) PCA for Random Forest – 3%

```r
# Look at 10th variance
variance_explained1A[10, ]
```

```
##    NmbrPCs    CumVar
## 10      10 0.9485995
```



It shows that 10 PCAs can explain about 95% dataset. We have new train and test sets after PCA for topics dataset person A:

```
# Do PCA for creating 10 factors
pca1A <- preProcess(x = train1A[-301], method = "pca", pcaComp = 10)
train_pca1A <- predict(pca1A, train1A)
test_pca1A <- predict(pca1A, test1A)
head(train_pca1A)
```

```
##     V1         PC1        PC2        PC3      PC4        PC5       PC6
## 1    0 -103.427978 -182.06974 -29.176609 1.033714 14.7049253 10.774456
## 3    0  -72.910288  -75.52997 -15.003642 4.783059  6.8207197  5.451226
## 6    0  -19.159370   12.53354  -7.839942 3.403300 -0.0239336  3.734457
## 7    0  -12.661774   17.65904  -9.780265 2.491799 -0.8497776  2.328911
## 9    0   -5.548527   20.61640 -10.090990 1.420890 -1.2864858  1.767817
## 10   0   -4.154898   22.22218 -11.487416 1.450429 -1.3298780  1.554392
##            PC7         PC8        PC9       PC10
## 1    8.0358487 -0.90692391 -8.2011729 -4.9257149
## 3   -2.6810256 -0.56614380 -0.5382019 -1.3123040
## 6   -1.4821080 -0.06791971  1.7002436 -1.3405662
## 7   -0.9905497  0.62839421  1.5108605 -1.4963540
## 9   -0.5737773  0.46690617  1.5037475 -0.6338052
## 10  -0.4869090  0.97806523  0.8015575 -1.9159111
```

Applying random forest and we got 98% accuracy for train and test on person A topics dataset.

```r
# Make prediction
pred1A <- predict(pca_rf1A, test1A)

# Confusion Matrix
confusionMatrix(pred1A, factor(test1A[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 358   8
##          1   1  82
##
##                Accuracy : 0.98
##                  95% CI : (0.9623, 0.9908)
##     No Information Rate : 0.7996
##     P-Value [Acc > NIR] : <2e-16
```

Doing PCA with person B and use random forest model's person A, we got 93.81% accuracy.

```r
# Predicting the results
pred1B <-  predict(pca_rf1A, data1B)

# Making the confusion matrix
confusionMatrix(factor(pred1B), factor(data_set1B$V1))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1295   50
##          1   63  417
##
##                Accuracy : 0.9381
##                  95% CI : (0.926, 0.9487)
##     No Information Rate : 0.7441
##     P-Value [Acc > NIR] : <2e-16
```

Doing similar with the conditional dataset, we got 99.16% accuracy for training and testing on person A, 91.79% accuracy for training on A and testing on B.

```r
# Make prediction
pred2A <- predict(pca_rf2A, test2A)

# Confusion Matrix
confusionMatrix(pred2A, factor(test2A[,301]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 339   3
##          1   1 134
##
##                Accuracy : 0.9916
##                  95% CI : (0.9787, 0.9977)
##     No Information Rate : 0.7128
##     P-Value [Acc > NIR] : <2e-16
```

```r
# Predicting the results
pred2B <-  predict(pca_rf2A, data2B)

# Making the confusion matrix
confusionMatrix(pred2B, factor(data2B$V1))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1413  135
##          1   32  454
##
##                Accuracy : 0.9179
##                  95% CI : (0.9051, 0.9295)
##     No Information Rate : 0.7104
##     P-Value [Acc > NIR] : < 2.2e-16
```

PCA does not affect Random Forest.

d(viii) ROC curve for Random Forest – 3%

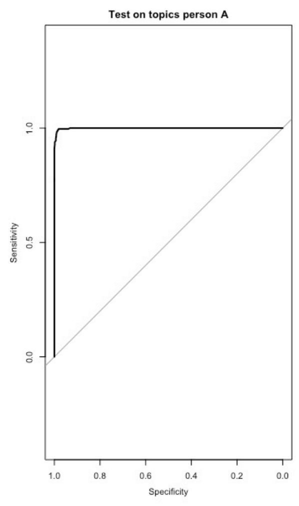
```r
# Plot optimal parameter model's performance on training data 1A
rf.roc1A<-roc(train1A$V1,rf1A$votes[,2])
```

```
## Setting levels: control = 0, case = 1
```

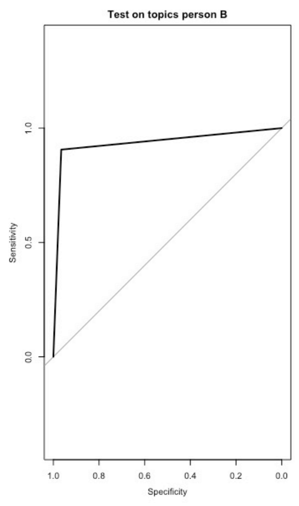
```
## Setting direction: controls < cases
```

```r
auc(rf.roc1A)
```

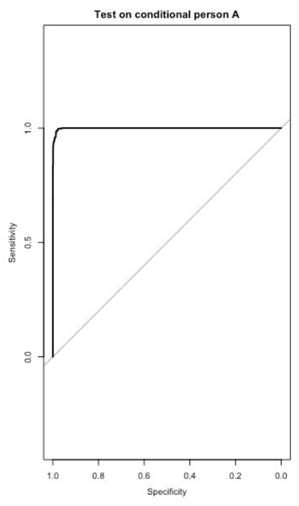
```
## Area under the curve: 0.9989
```



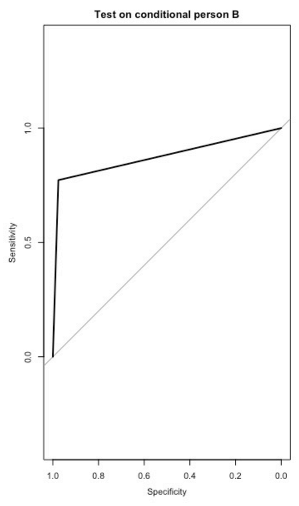
auc: 0.9989



auc: 0.92.43



auc: 0.999



auc: 0.8733

By looking at ROC curve and AUC numbers are excellent, Random Forest got very good predictions.

7. Compare the results of the two classifiers SVM & Random Forest – 20%

		SVM train on person A	SVM train on person B	Random Forest train on person A	Random Forest train on person B	
Topics	SVM test on person A	95.99	**71.88**	98.00	**96.1**	RF test on person A
Conditional		97.06	**63.92**	99.16	**94.65**	
Topics	SVM test on person B	**74.14**	89.72	**93.86**	97.16	RF test on person B
Conditional		**71.04**	91.34	**91.64**	97.64	

From the table, we can see Random Forest works much better than SVM on these datasets. In the research “Do we Need Hundreds of Classifiers to Solve Real World

Classification Problems?” (Fernández-Delgado et al., 2014), after evaluating 179 classifiers arising from 17 families (including Random Forest and Support Vector Machine) for 121 datasets which represent the whole UCI data base (excluding the large-scale problems) and other own real problems, in order to achieve significant conclusions about the classifier behavior, not dependent on the data set collection. The classifiers most likely to be the bests are the random forest versions, implemented in R and accessed via caret.

Support Vector Machine presents as one of the most used robust prediction methods that can be applied to many use cases involving classifications. However, SVMs are known to be fairly insensitive to a very large number of irrelevant features, such application of SVMs likely biases down their performance (Statnikov, Wang and Aliferis, 2008). Applying PCA helps SVM choosing the most important features and we got better results.

Bibliography

TopBlog. (n.d.). *TopBlog*. [online] Available at: https://naivedatascientist.co.in [Accessed 21 Apr. 2021].

Kolte, C.J. and Shrivas, A. (2017). Intelligent knowledge sharing for agricultural information. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/8300906 [Accessed 21 Apr. 2021].

Ray, S. (2017). Sept 13th, 2017. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/ [Accessed 21 Apr. 2021].

B, D.M., B, J., S, G. and N, Mrs.I. (2019). CREDIT CARD FRAUD DETECTION USING RANDOM FOREST. International Research Journal of Engineering and Technology, [online] 6(3). Available at: https://www.irjet.net/archives/V6/i3/IRJET-V6I3710.pdf.

D., P., Dimitriou, N. and Arandjelović, O. (2021). Artificial Intelligence and Deep Learning in Pathology. Artificial Intelligence and Deep Learning in Pathology, [online] pp.149–173. Available at: https://www.sciencedirect.com/science/article/pii/B9780323675383000087?via%3Dihub.

Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D. and Fernández-Delgado, A. (2014). Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? Journal of Machine Learning Research, [online] (15), pp.3133–3181. Available at: https://jmlr.org/papers/volume15/delgado14a/delgado14a.pdf.

Nigri, E. and Arandjelovic´O. (n.d.). Machine Learning Based Detection of Kepler Objects of Interest. [online] Available at: https://research-repository.st-andrews.ac.uk/bitstream/handle/10023/11659/2017_ICMEW_paper1.pdf?isAllowed=y&sequence=1.

Statnikov, A., Wang, L. and Aliferis, C.F. (2008). A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. BMC Bioinformatics, 9(1), p.319.