Emotion Recognition System

Project Report Submitted by

T Baskaran

(In final fulfillment of the Data Science: Capstone course of the HarvardX Professional Certificate Program in Data Science)

May 26, 2019

Table of Contents

Preface

Data Science is a field, which I have chosen after my retirement from Indian Statistical Service of the Government of India. Having inspired by the power of R as a tool for statistical analysis, I decided to become a data scientist, in spite of my age. edX and the HarvardX gave me the opportunity to pursue Data Science and I could complete all the previous eight courses. Joining the **Data Science: Capstone**, the final course in the **HarvardX Professional Certificate Program in Data Science** has resulted in doing this project namely **Emotion Recognition System**, which is the second in my pursuit of Data Science, the first being **Movie Recommendation System**. I hope to continue doing several projects of this kind in this fascinating field of Data Science.

Acknowledgement

Though, I was not knowing anything about Data Science, and not very fluent in R, the course gave me the necessary confidence and brought me to this level. For this, I am indebted to **Prof Rafael A. Irizarry**, whose lectures had given me the necessary insights into Data Science and confidence to take up Data Science as my post-retirement carrier. I thank him for this. The **staff of this course** at the HarvardX gave great inputs at the time of need and encouraged the learners. My thanks are due to them. I am thankful to the **edX platform and the HarvardX** for having made available all the facilities for completing all the courses and this project. I also thank all the **fellow learners** for their wonderful inputs through the discussion forum.

Summary

The project is about conceiving an **Emotion Recognition System** in Arabic conversation utilizing certain acoustic features. This has been accomplished using the **Arabic Natural Audio Dataset (ANAD)** and fitting various models and examining their performance through accuracies. Of all the models examined, The **Logistic Regression model** using the CARET package after cross validation performs well among all the models examined, with an accuracy 97.31%, which is the highest among the models. Therefore, the Logistic Regression model using the CARET package after cross validation is considered the most appropriate among all the models examined, and this is recommended for recognizing emotions in Arabic conversations given the required acoustic features.

Key words: Emotion recognition, Arabic Natural Audio Dataset, ANAD

Emotion Recognition System

1. Introduction

1.1 Objective

The present project is in final fulfilment of the **Data Science: Capstone** course of the **HarvardX Professional Certificate Program in Data Science**. The learner has chosen to generate an emotion recognition system for the **Choose Your Own Project** component of the course. The objective of the project is to predict or recognize the emotion behind a conversation, given certain acoustic features of the conversation.

1.2 Overview

The project **Emotion Recognition System** using the **Arabic Natural Audio Dataset (ANAD)** is another attempt to improve the communication in Arabic language for the hearing impaired people. When two people converse face-to-face, the text as well as the emotions with which they speck can be recorded. However, it would be difficult to recognize the emotion in the conversation when it is not performed face-to-face. Therefore, integration of an effective **Emotion Recognition System** with speech-to-text system helps hearing impaired individuals to make successful conversation especially through phones. It is possible that a hearing-impaired person when types a message, the person on the other side is able to hear the words spoken. In the same way, when the person at one end of the line speaks to a hearing-impaired individual who is at the other end, the words are received as text on say, a mobile phone of the hearing-impaired. However, in this process, the emotion of both the parties is missing. As a result, the readability of the conversation is not complete and there is a need to improve the usefulness of the existing systems.

Here in this project, an attempt has been made to recognize emotions based on the different acoustic features captured in the conversation in Arabic. Emotion

recognition from speech data is an interesting field of research especially with Arabic speech data as very little work has been done[1]. Using the Arabic Natural Audio Dataset (ANAD) an attempt has been made to recognize three emotions: **Happy, Angry and Surprised** - from natural Arabic speech.

1.3 Approach

While developing the **Emotion Recognition System** from the Arabic Natural Audio Dataset, several models of the type k Nearest Neighbours, Logistic Regression, Support Vector Machines (SVM), Random Forests and Neural Network have been examined. Generally the **CARET** package has been used. **Cross Validation** has been attempted in some models. The **Principal Component Analysis (PCA)** has also been attempted for the Logistics Regression. The **e1071** package has been tried for the SVM model and the **randomForest** package for the RF model. As the ANAD contains the normalized data in a file, it is directly used for analysis. The general approach followed in this project is that the data set is first divided into train and test sets so that the model is trained using the training data set, and evaluated or tested using the test data set. The metric used for comparing the performance of the different models is the **overall accuracy** available from the **confusion matrix** from the CARET package. The model with the maximum accuracy is considered as the appropriate model for recognizing the emotions in the conversation.

1.4 Structure of the report

While **Section 1: Introduction** presents the objective of doing this project together with an overview of the project and the approach followed, **Section 2: Data Set** explains the data set used. It also talks of the different features of the data set. **Section 3: Analysis** discusses the data analysis. In this section, the report deals with Exploratory Data Analysis as well as fitting different models. Each model is vividly explained. It also

[1] https://www.ripublication.com/ijaer18/ijaerv13n5_38.pdf

provides the results of the analysis. **Section 4: Conclusion** recommends the final model which can be adopted as the **Emotion Recognition System**.

2. Data Set

As pointed out earlier, the **Arabic Natural Audio Dataset (ANAD)** is used in this project. It is available at: "https://data.mendeley.com/datasets/xm232yxf7t/1". The specific file that is used for our analysis namely **"ANAD_Normalized.csv"** is at the url: "https://data.mendeley.com/datasets/xm232yxf7t/1/files/e535362e-ce98-4729-8c7a-f32fec55cc30/ANAD_Normalized.csv?dl=1". This is the first data set developed to recognize three discrete emotions: **Happy, Angry, and Surprised**[2].

Eight videos of live calls between an anchor and a human outside the studio were downloaded by the developers of the data set from online Arabic talk shows. To label each video, 18 listeners were asked to listen to each video and select whether they perceive a happy, angry or surprised emotion. Each video was then divided into turns: callers and receivers. Silence, laughs and noisy chunks were removed. Every chunk was then automatically divided into 1-sec speech units forming the final corpus composed of 1383 records with 505 happy, 137 surprised and 741 angry units.

Twenty five **low-level descriptors (LLDs)** - a combination of acoustic and spectral features - were extracted from every speech unit. These features are: intensity, zero crossing rates, MFCC 1-12 (Mel-frequency cepstral coefficients), F0 (Fundamental frequency) and F0 envelope, probability of voicing and, LSP (Line spectral pairs) frequency 0-7. On every feature nineteen statistical functions were calculated. These functions are: maximum, minimum, range, absolute position of maximum, absolute position of minimum, arithmetic of mean, Linear Regression1, Linear Regression2, Linear RegressionA, Linear RegressionQ, standard Deviation, kurtosis, skewness, quartiles 1, 2, 3 and, inter-quartile ranges 1-2, 2-3, 1-3. The delta coefficient for every LLD is also computed as an estimate of the first derivative hence leading to a total of 950 features.

[2] https://data.mendeley.com/datasets/xm232yxf7t/1

However, certain ineffective features were removed. Further, two variables - the **name** of the 1-sec speech units and the **Type** of the emotion recorded in terms of 1(happy), 2 (surprised) and 3 (angry) - were added resulting in a new database of 1383 records with 847 features.

The data set contains the following five files:

- **1sec_segmented.zip (157 MB):** Every video in the caller_reciever_turns folder is automatically divided into 1 sec audio files. These files are named as Videoname_chunkNumber (sec_number), for example, V1_1 (3) which refers to the 3rd second of the first turn of video V1.
- **ANAD.csv (9 MB):** Contains the Low Level descriptors features of 1384 audio units.
- **ANAD_Normalized.csv (10 MB):** Contains the normalized values of ANAD.csv
- **caller_Reciever_turns.zip (79 MB):** Each video in the full video folder is manually segmented into caller/ receiver turns. Laughs pause and noise chunks were removed. These video files are denoted by V1_1 for turn1 of video V1, V2_3 for turn 3 of video V2 etc…
- **full videos.zip (296 MB):** Videos are named as V1, V2 till V8.

Of the above, the present project uses the "ANAD_Normalized.csv" file for analysis.

2.1 Data pre-processing

As the ANAD has a file containing the normalized data in **.csv** format, the learner did not do much of pre-processing and used the **ANAD_Normalized.csv** file directly for the analysis. However, it was possible to locate those variables which had near zero variation. Such variables were removed from the data set so as to reduce the number of features. Further, the variables **name** and **Emotion** were not required for the analysis and hence they were removed. The variable **Type**, which is used as the label, is converted into a factor variable, thus resulting in a new data set, we call it, **data** with 1383 observations and 840 features.

```
########################  Initial Settings   ################
# List of packages required for this analysis
pkg <- c("tidyverse", "knitr", "tictoc", "caret", "caTools", "kernlab",
"e1071", "randomForest", "nnet")
```

```
# Check if packages are not installed and assign the
# names of the packages not installed to the variable new.pkg
new.pkg <- pkg[!(pkg %in% installed.packages())]

# If there are any packages in the list that aren't installed,
# install them
if (length(new.pkg)) {
  install.packages(new.pkg, repos = "http://cran.rstudio.com")
}
#Load the libraries
library(tictoc)
library(caret)
library(tidyverse)
library(caTools)
library(kernlab)
library(e1071)
library(randomForest)
library(nnet)
library(knitr)

##############      Data Preparation         ######################
tic()
#Download the data
if (!file.exists("ANAD_Normalized.csv")){

download.file("https://data.mendeley.com/datasets/xm232yxf7t/1/files/e535362e-ce98-4729-8c7a-f32fec55cc30/ANAD_Normalized.csv?dl=1",
"ANAD_Normalized.csv")
  }

#Read the data set
anad <- read.csv("ANAD_Normalized.csv")
dim(anad)
## [1] 1383  847
#Check for the near zero variance variables
nzv <- nearZeroVar(anad)
dim(anad)
## [1] 1383  847
length(nzv)
## [1] 5
```

```
#Remove the nzv columns
data <- anad[,-nzv]
dim(data)
```

```
## [1] 1383  842
```

```
#Remove the "name" and "Emotion" variables as they are not going to be
used in the analysis
data <- data[,-c(1,2)]

# Make the integer variable "Type" as factor
data$Type <- as.factor(data$Type)
str(data)
```

```
## 'data.frame':    1383 obs. of  840 variables:
##  $ Type                        : Factor w/ 3 levels "1","2","3": 2 2 2 2 2 2 2 2 2 2 ...
##  $ pcm_intensity_sma_max           : num  0.286 0.286 0.286 0.286 0.143 ...
##  $ pcm_intensity_sma_range         : num  0.286 0.286 0.286 0.286 0.143 ...
##  $ pcm_intensity_sma_maxPos        : num  0.415 0.404 0.404 0.404 0.606 ...
##  $ pcm_intensity_sma_minPos        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ pcm_intensity_sma_amean         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ pcm_intensity_sma_linregc2      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ pcm_intensity_sma_linregerrA    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ pcm_intensity_sma_stddev        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ pcm_intensity_sma_skewness      : num  0.436 0.418 0.438 0.418 0.404 ...
##  $ pcm_intensity_sma_kurtosis      : num  0.188 0.161 0.179 0.174 0.161 ...
##  $ pcm_intensity_sma_quartile3     : num  0.5 0.5 0 0.5 0.5 0 0.5 0.5 0.5 0 ...
##  $ pcm_intensity_sma_iqr23         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ pcm_intensity_sma_iqr13         : num  0.5 0.5 0 0.5 0.5 0 0.5 0.5 0 0 ...
##  $ pcm_fftMag_mfcc_sma1_max        : num  0.41 0.482 0.43 0.596 0.629 ...
##  $ pcm_fftMag_mfcc_sma1_min        : num  0.144 0.235 0.175 0.167 0.231 ...
##  $ pcm_fftMag_mfcc_sma1_range      : num  0.538 0.512 0.526 0.666 0.633 ...
##  $ pcm_fftMag_mfcc_sma1_maxPos     : num  0.43 0.086 0.968 0.204 0.129 ...
##  $ pcm_fftMag_mfcc_sma1_minPos     : num  0.915 0.927 0.927 0.902 0.951 ...
##  $ pcm_fftMag_mfcc_sma1_amean      : num  0.335 0.355 0.31 0.388 0.35 ...
##  $ pcm_fftMag_mfcc_sma1_linregc1   : num  0.319 0.32 0.325 0.317 0.318 ...
##  $ pcm_fftMag_mfcc_sma1_linregc2   : num  0.558 0.566 0.471 0.623 0.58 ...
##  $ pcm_fftMag_mfcc_sma1_linregerrA : num  0.55 0.536 0.528 0.614 0.645 ...
##  $ pcm_fftMag_mfcc_sma1_linregerrQ : num  0.4 0.324 0.403 0.434 0.484 ...
##  $ pcm_fftMag_mfcc_sma1_stddev     : num  0.596 0.545 0.547 0.627 0.645 ...
##  $ pcm_fftMag_mfcc_sma1_skewness   : num  0.172 0.317 0.184 0.306 0.512 ...
##  $ pcm_fftMag_mfcc_sma1_kurtosis   : num  0.485 0.398 0.481 0.461 0.375 ...
```

```
##  $ pcm_fftMag_mfcc_sma1_quartile1    : num  0.331 0.321 0.307 0.321 0.27 ...
##  $ pcm_fftMag_mfcc_sma1_quartile2    : num  0.38 0.362 0.355 0.418 0.368 ...
##  $ pcm_fftMag_mfcc_sma1_quartile3    : num  0.496 0.47 0.404 0.507 0.448 ...
##  $ pcm_fftMag_mfcc_sma1_iqr12        : num  0.218 0.195 0.214 0.35 0.358 ...
##  $ pcm_fftMag_mfcc_sma1_iqr23        : num  0.492 0.47 0.269 0.402 0.374 ...
##  $ pcm_fftMag_mfcc_sma1_iqr13        : num  0.466 0.435 0.327 0.512 0.5 ...
##  $ pcm_fftMag_mfcc_sma2_min          : num  0.701 0.716 0.704 0.67 0.724 ...
##  $ pcm_fftMag_mfcc_sma2_range        : num  0.381 0.45 0.365 0.444 0.375 ...
##  $ pcm_fftMag_mfcc_sma2_maxPos       : num  0.809 0.798 0.809 0.819 0.777 ...
##  $ pcm_fftMag_mfcc_sma2_minPos       : num  0.234 0.234 0.234 0.0319 0.234 ...
##  $ pcm_fftMag_mfcc_sma2_amean        : num  0.828 0.835 0.841 0.807 0.831 ...
##  $ pcm_fftMag_mfcc_sma2_linregc1     : num  0.427 0.443 0.428 0.44 0.418 ...
##  $ pcm_fftMag_mfcc_sma2_linregc2     : num  0.768 0.655 0.774 0.648 0.837 ...
##  $ pcm_fftMag_mfcc_sma2_linregerrA   : num  0.247 0.3 0.264 0.319 0.323 ...
##  $ pcm_fftMag_mfcc_sma2_linregerrQ   : num  0.0933 0.1269 0.0986 0.1276 0.1334
...
##  $ pcm_fftMag_mfcc_sma2_stddev       : num  0.274 0.409 0.284 0.388 0.286 ...
##  $ pcm_fftMag_mfcc_sma2_skewness     : num  0.648 0.681 0.556 0.653 0.655 ...
##  $ pcm_fftMag_mfcc_sma2_kurtosis     : num  0.445 0.282 0.362 0.271 0.351 ...
##  $ pcm_fftMag_mfcc_sma2_quartile1    : num  0.809 0.682 0.803 0.704 0.751 ...
##  $ pcm_fftMag_mfcc_sma2_quartile2    : num  0.806 0.845 0.833 0.78 0.822 ...
##  $ pcm_fftMag_mfcc_sma2_quartile3    : num  0.839 0.893 0.876 0.883 0.87 ...
##  $ pcm_fftMag_mfcc_sma2_iqr12        : num  0.0809 0.4518 0.155 0.2536 0.2477 ...
##  $ pcm_fftMag_mfcc_sma2_iqr23        : num  0.157 0.202 0.188 0.361 0.2 ...
##  $ pcm_fftMag_mfcc_sma2_iqr13        : num  0.17 0.507 0.252 0.448 0.337 ...
##  $ pcm_fftMag_mfcc_sma3_max          : num  0.694 0.708 0.736 0.708 0.735 ...
##  $ pcm_fftMag_mfcc_sma3_min          : num  0.335 0.387 0.303 0.336 0.281 ...
##  $ pcm_fftMag_mfcc_sma3_range        : num  0.6 0.553 0.68 0.613 0.705 ...
##  $ pcm_fftMag_mfcc_sma3_minPos       : num  0.761 0.772 0.793 0.772 0.837 ...
##  $ pcm_fftMag_mfcc_sma3_amean        : num  0.491 0.53 0.494 0.456 0.509 ...
##  $ pcm_fftMag_mfcc_sma3_linregc1     : num  0.404 0.401 0.397 0.405 0.39 ...
##  $ pcm_fftMag_mfcc_sma3_linregc2     : num  0.518 0.567 0.565 0.476 0.626 ...
##  $ pcm_fftMag_mfcc_sma3_linregerrA   : num  0.485 0.382 0.467 0.515 0.454 ...
##  $ pcm_fftMag_mfcc_sma3_linregerrQ   : num  0.257 0.188 0.268 0.284 0.266 ...
##  $ pcm_fftMag_mfcc_sma3_stddev       : num  0.531 0.475 0.579 0.549 0.632 ...
##  $ pcm_fftMag_mfcc_sma3_skewness     : num  0.252 0.253 0.242 0.329 0.219 ...
##  $ pcm_fftMag_mfcc_sma3_kurtosis     : num  0.335 0.355 0.358 0.286 0.302 ...
##  $ pcm_fftMag_mfcc_sma3_quartile1    : num  0.469 0.506 0.443 0.36 0.386 ...
```

```
##  $ pcm_fftMag_mfcc_sma3_quartile2    : num  0.502 0.544 0.501 0.474 0.552 ...
##  $ pcm_fftMag_mfcc_sma3_quartile3    : num  0.57 0.57 0.572 0.547 0.605 ...
##  $ pcm_fftMag_mfcc_sma3_iqr12        : num  0.15 0.164 0.217 0.372 0.519 ...
##  $ pcm_fftMag_mfcc_sma3_iqr23        : num  0.317 0.188 0.324 0.329 0.274 ...
##  $ pcm_fftMag_mfcc_sma3_iqr13        : num  0.269 0.205 0.313 0.41 0.467 ...
##  $ pcm_fftMag_mfcc_sma4_max          : num  0.535 0.524 0.655 0.519 0.627 ...
##  $ pcm_fftMag_mfcc_sma4_min          : num  0.419 0.415 0.415 0.435 0.439 ...
##  $ pcm_fftMag_mfcc_sma4_range        : num  0.514 0.506 0.655 0.476 0.592 ...
##  $ pcm_fftMag_mfcc_sma4_maxPos       : num  0.9032 0.9032 0.0968 0.0753 0.1075
...
##  $ pcm_fftMag_mfcc_sma4_amean        : num  0.481 0.434 0.501 0.455 0.5 ...
##  $ pcm_fftMag_mfcc_sma4_linregc1     : num  0.527 0.545 0.514 0.528 0.516 ...
##  $ pcm_fftMag_mfcc_sma4_linregc2     : num  0.539 0.363 0.643 0.509 0.629 ...
##  $ pcm_fftMag_mfcc_sma4_linregerrA   : num  0.645 0.535 0.769 0.571 0.673 ...
##  $ pcm_fftMag_mfcc_sma4_linregerrQ   : num  0.386 0.261 0.556 0.316 0.446 ...
##  $ pcm_fftMag_mfcc_sma4_stddev       : num  0.589 0.539 0.734 0.532 0.655 ...
##  $ pcm_fftMag_mfcc_sma4_skewness     : num  0.504 0.601 0.667 0.614 0.71 ...
##  $ pcm_fftMag_mfcc_sma4_kurtosis     : num  0.235 0.228 0.245 0.225 0.277 ...
##  $ pcm_fftMag_mfcc_sma4_quartile1    : num  0.357 0.317 0.321 0.364 0.387 ...
##  $ pcm_fftMag_mfcc_sma4_quartile2    : num  0.592 0.495 0.549 0.512 0.555 ...
##  $ pcm_fftMag_mfcc_sma4_quartile3    : num  0.641 0.57 0.632 0.586 0.591 ...
##  $ pcm_fftMag_mfcc_sma4_iqr23        : num  0.35 0.452 0.466 0.444 0.316 ...
##  $ pcm_fftMag_mfcc_sma4_iqr13        : num  0.688 0.635 0.742 0.57 0.534 ...
##  $ pcm_fftMag_mfcc_sma5_max          : num  0.637 0.594 0.648 0.78 0.606 ...
##  $ pcm_fftMag_mfcc_sma5_min          : num  0.398 0.389 0.381 0.411 0.399 ...
##  $ pcm_fftMag_mfcc_sma5_range        : num  0.52 0.481 0.553 0.67 0.482 ...
##  $ pcm_fftMag_mfcc_sma5_maxPos       : num  0 0.8298 0.0532 0.0957 0.1702 ...
##  $ pcm_fftMag_mfcc_sma5_minPos       : num  0.527 0.527 0.527 0.398 0.527 ...
##  $ pcm_fftMag_mfcc_sma5_amean        : num  0.484 0.445 0.473 0.493 0.455 ...
##  $ pcm_fftMag_mfcc_sma5_linregc1     : num  0.577 0.589 0.582 0.581 0.593 ...
##  $ pcm_fftMag_mfcc_sma5_linregc2     : num  0.596 0.473 0.547 0.572 0.455 ...
##  $ pcm_fftMag_mfcc_sma5_linregerrA   : num  0.741 0.544 0.657 0.657 0.576 ...
##  $ pcm_fftMag_mfcc_sma5_stddev       : num  0.616 0.46 0.546 0.581 0.484 ...
##  $ pcm_fftMag_mfcc_sma5_kurtosis     : num  0.235 0.247 0.237 0.369 0.259 ...
##  $ pcm_fftMag_mfcc_sma5_quartile1    : num  0.339 0.338 0.336 0.35 0.327 ...
##  $ pcm_fftMag_mfcc_sma5_quartile2    : num  0.462 0.44 0.468 0.476 0.457 ...
##   [list output truncated]
toc()
```

```
## 23.59 sec elapsed
```

3. Data Analysis

3.1 Measuring the performance

While analyzing the data, it is proposed to construct several models. It is therefore essential to use a specific yardstick to measure the goodness of the model. This project proposes to use the **Overall Accuracy** as the measure the performance of the model. For this purpose, the **confusionMatrix** function of the **CARET** package is proposed to be used. This function calculates a cross-tabulation of observed and predicted classes with associated statistics and outputs a list of elements including **overall$Accuracy**. The function essentially takes two arguments: $data$ as a factor of predicted classes and $reference$ as a factor of classes of true values.

3.2 Exploratory Data Analysis

Before resorting to actual model building, let us understand the data set by doing some Exploratory Data Analysis (EDA). Let us first look at the frequency distribution and visualize the same.

```
##################           EDA           ####################

# Look at the frequency distribution of the Type of emotion
class(data$Type)
## [1] "factor"
table((data$Type))
##
##  1  2  3
## 505 137 741
# Visual display of the frequency distribution of the "Emotion"
data %>% ggplot(aes(x=Type, fill=Type))+
  geom_bar()+
  scale_fill_discrete(name = "Emotion", labels = c("Angry",
"Surprised", "Happy"))+
  ggtitle("Distribution of Emotions")+
  scale_x_discrete(name = "Type")+
  theme_bw()
```

It is clear from the above that there are 505 Angry, 137 Surprised and 741 Happy emotions in the data.

Let us further find out whether the data set has any missing values.

```
#Check for the missing values. There is no missing observation in the
entire dataset
sum(is.na(anad))
```

```
## [1] 0
```

As the sum is zero, the data set does not contain any missing value.

3.3 Model Development

The project proposes to fit different models for recognizing emotions and the model with the maximum overall accuracy is considered the appropriate. Such a model is recommended for external application.

For the purpose of fitting the model, the data set is partitioned into training and test data sets in the ratio of 80:20. We train the model using the training data set, then test it using the test set and then proceed to calculate the accuracy. The process

continues for all the models in the project. A results table is finally created so that the different methods employed and the corresponding accuracies are properly recorded.

```
################ Model Development ########################

tic()
#Partition the data set "data" in to training (80%) and test (20%) data sets
set.seed(1)
test_index <- createDataPartition(data$Type, times = 1, p = 0.2, list = FALSE)
train_set <- data[-test_index,]
test_set <-  data[test_index,]
dim(train_set)
## [1] 1105  840
dim(test_set)
## [1] 278 840
time_part <- toc()
## 0.09 sec elapsed
toc()
```

3.3.1 k Nearest Neighbours (kNN)

The kNN or k Nearest Neighbours algorithm is one of the popular machine learning algorithms. It is a supervised learning algorithm, where the destination is known, but the path to the destination is not[3]. When we have several groups of labeled samples and the items present in the groups are homogeneous in nature, and when we also have an unlabeled example which needs to be classified into one of the several labeled groups, we can use the kNN algorithm. It stores all available cases and classifies new cases by a majority vote of its k neighbors. Choosing the number of nearest neighbors or in other words determining the value of k, plays a significant role in determining the efficacy of the model. Thus, selection of k will determine how well the data can be utilized to generalize the results of the kNN algorithm. A large k value has

[3] https://www.analyticsvidhya.com/blog/2015/08/learning-concept-knn-algorithms-programming/

benefits which include reducing the variance due to the noisy data; the side effect being developing a bias due to which the model-builder tends to ignore the smaller patterns which may have useful insights.

3.3.1.1 kNN Using CARET with Cross Validation

While developing the kNN model, it is trained using the **train** function of the **CARET** package with the **"knn"** method. The model is then tested with the test set and the predictions are made using the **predict** function and the overall accuracy is calculated. In order to enhance the performance of the model, Cross Validation has been resorted to.

```
###############  k Nearest Neighbours     ###############
#Model 1: kNN model after cross validation

tic()
set.seed(2)
control <- trainControl(method = "cv",
                        number = 5, p = .9)

train_knn_cv <- train(Type ~ .,
                      method = "knn",
                      tuneGrid = data.frame(k = c(3,5,7)),
                      trControl = control,
                      data = train_set)

pred_knn_cv <- predict(train_knn_cv, test_set, type = "raw")

acc_knn_cv <- confusionMatrix(pred_knn_cv,
                              test_set$Type)$overall[["Accuracy"]]
acc_knn_cv
## [1] 0.9136691
time_knn_cv <- toc()
## 49.57 sec elapsed
```

The model after running for 49.57 seconds provides us an accuracy of 0.9136691.

3.3.2 Logistic Regression

Logistic regression is another popular machine learning algorithm. While regression analysis helps to estimate relationships among variables, the problem depends on the nature of the variable to be predicted. Linear Regression is used when the variable to be estimated is continuous, whereas the Logistics Regression is the appropriate model when the variable to be predicted is a classificatory variable,.

3.3.2.1 Logistic Regression Using CARET with Cross Validation

Here in our project, the variable to be predicted is the **Type**, which is a categorical variable. The Logistic Regression has, therefore been chosen. While fitting the Logistic Regression, we train the model using the **train** function of the **CARET** package with the **"LogitBoost"** method. The model is then tested with the test set and the predictions are made using the predict function and the overall accuracy is calculated. As the model uses the **"LogitBoost"** method, it is also called **Boosted Logistic Regression Model**. For enhanced performance the Cross Validation technique has been applied.

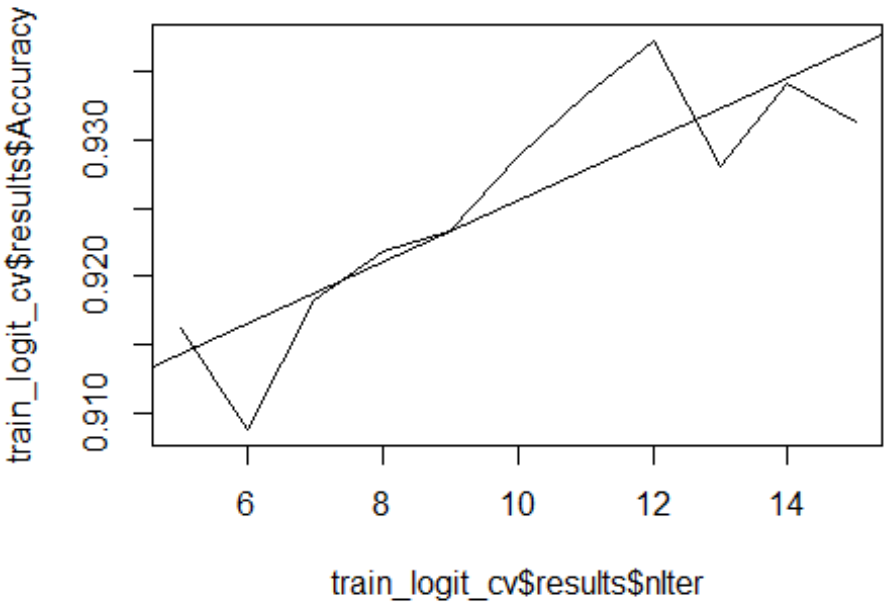
```
#####################   Logistic Regression    ###################
#Model 2: Boosted Logistic Regression with caret::train using the
method "LogitBoost" after Cross Validation

tic()
set.seed(2)
control <- trainControl(method = "cv",
                        number = 5, p = .9)
train_logit_cv <- train(Type ~ .,
                        method = "LogitBoost",
                        tuneGrid = data.frame(nIter=5:15),
                        trControl = control,
                        data = train_set)

plot(x=train_logit_cv$results$nIter,
     y=train_logit_cv$results$Accuracy,
     type='l')

fit_logit_cv <- lm(train_logit_cv$results$Accuracy ~
```

```
train_logit_cv$results$nIter)
abline(fit_logit_cv)
```




```
pred_logit_cv <- predict(train_logit_cv, test_set, type = "raw")

acc_logit_cv <- confusionMatrix(pred_logit_cv,
                                test_set$Type)$overall[["Accuracy"]]

acc_logit_cv
## [1] 0.9730769
time_logit_cv <- toc()
## 25.05 sec elapsed
```

This model has given us an overall accuracy of 0.9730769 after running for 25.05 seconds.

3.3.2.2 Logistic Regression with PCA

The Logistic Regression Model is again taken up after constructing principal components. Here we use the same CARET package with the method "LogitBoost" together with **preProcess = "pca"**.

```
#Model 3: Logistic Regression with PCA

tic()
fit_logit_pca <- train(Type ~ .,
                       data = train_set,
                       method = 'LogitBoost',
                       tuneGrid = data.frame(nIter=50),
                       preProcess = "pca")

pred_logit_pca <- predict(fit_logit_pca,
                          test_set,
                          type = "raw")

acc_logit_pca<- confusionMatrix(pred_logit_pca,
                                test_set$Type)$overall[["Accuracy"]]

acc_logit_pca
## [1] 0.924812
time_logit_pca<- toc()
## 183.68 sec elapsed
```

The model provides an accuracy of 0.924812 with an elapsed time of 183.68 seconds.

3.3.3 Support Vector Machines

In machine learning, Support Vector Machines (SVM) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. It is mostly used in classification problems. In this algorithm, each data item is plotted as a point in n-dimensional space (where n is number of features), with the value of each feature being the value of a particular coordinate. Then, classification is performed by finding the hyper-plane that best differentiates the classes[4].

[4] https://www.geeksforgeeks.org/classifying-data-using-support-vector-machinessvms-in-r/

3.3.3.1 SVM Model Using CARET package

In the process of developing the SVM model, it is trained using the **train** function of the **CARET** package with the **"svmLinear"** method. The model is then tested with the test set and the predictions are generated using the **predict** function and the overall accuracy is then calculated.

```
#####################   Support Vector Machines    ###################

#Model 4: SVM model using train function of the caret package

tic()
train_svm_caret <- train(Type ~ .,
                         method = "svmLinear",
                         data = train_set)

pred_svm_caret <- predict(train_svm_caret,
                          test_set,
                          type = "raw")

acc_svm_caret <- confusionMatrix(pred_svm_caret,
                                 test_set$Type)$overall["Accuracy"]
acc_svm_caret
##  Accuracy
## 0.9568345
time_svm_caret <- toc()
## 33.19 sec elapsed
```

This model has given us an overall accuracy of 0.9568345 after running for 33.19 seconds.

3.3.3.2 SVM Model Using e1071 package

Another attempt is also made to train the model using the **svm** function of the **e1071** package. The model is then tested with the test set and predictions are generated using the **predict** function. The overall accuracy is then calculated.

```
#Model 5: SVM model using the svm function of the e1071 package

tic()
colname <- names(train_set)
```

```
strpred <- paste(colname[!colname %in% "Type"], collapse = " + ")
formula_svm_e1071 <- as.formula(paste("Type ~ ", strpred))

fit_svm_e1071 <- svm(formula_svm_e1071,
                     data = train_set)

pred_svm_e1071 <-  predict(fit_svm_e1071,
                           newdata = test_set)

test_set$Type <- factor(test_set$Type)

acc_svm_e1071 <- confusionMatrix(pred_svm_e1071,
                                 test_set$Type)$overall["Accuracy"]
acc_svm_e1071
##  Accuracy
## 0.9604317
time_svm_e1071 <- toc()
## 5.77 sec elapsed
```

This model has given us an overall accuracy of 0.9604317 after running for 5.77 seconds.

3.3.4 Random Forests

Random Forests Model is one of the most widely used machine learning algorithms for classification among the supervised classification algorithms. It can also be used for regression, but it performs well on classification. Random forests are an ensemble learning method that operate by constructing a lot of decision trees at training time and outputting the class that is the mode of the classes output by individual trees.

3.3.4.1 Random Forests Model using randomForest package

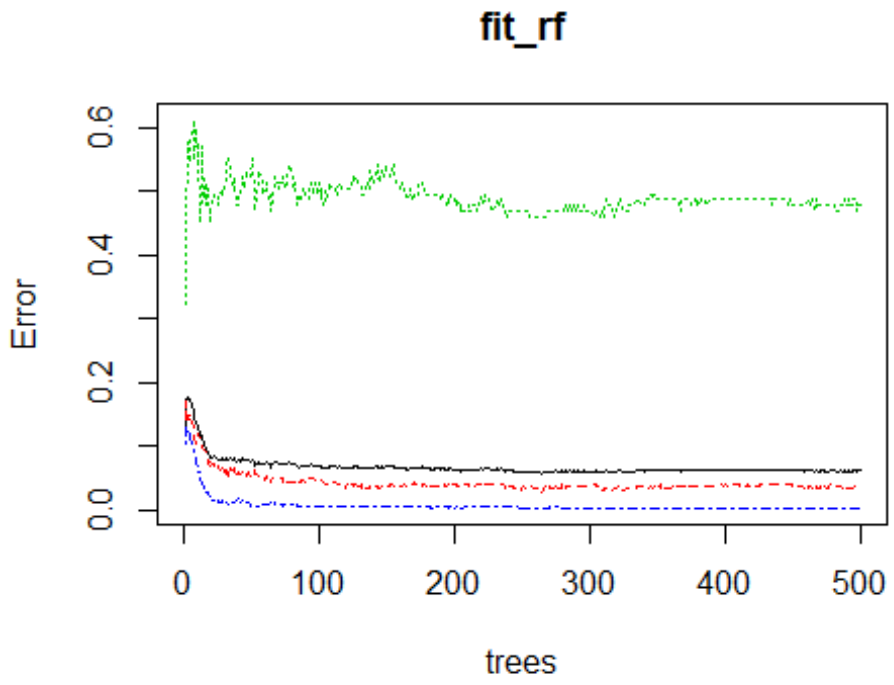
In this Random Forests model, the **randomForest** function of the randomForest packages is used to train the model. The model is then tested with the test set and predictions are made using the **predict** function. The overall accuracy is then calculated.

```
#####################   Random Forests    ###################

#Model 6: Random Forest model with randomForest package
```

```
tic()
fit_rf <- randomForest(Type~.,
                       data = train_set)

plot(fit_rf)
```




```
pred_rf <-  predict(fit_rf,
                    newdata = test_set)

acc_rf <- confusionMatrix(pred_rf,
                          test_set$Type)$overall["Accuracy"]

acc_rf
##  Accuracy
## 0.9604317
time_rf <- toc()
## 35.75 sec elapsed
```

The Random Forests Model has provided us with an accuracy of 0.9604317. It has taken only 35.75 seconds to perform.

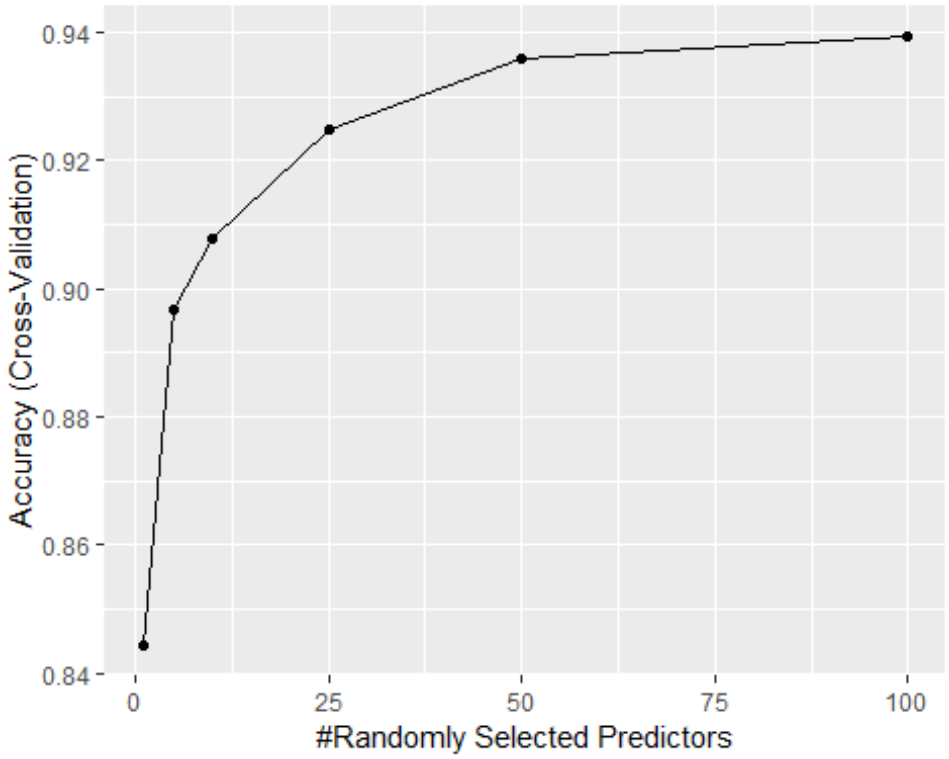
3.3.4.2 Random Forests Model Using CARET with Cross Validation

An attempt has been made to use the **train** function of the **CARET** package with the **"rf"** method and train and evaluate the model as follows after subjecting to Cross Validation.

```
#Model 7: Random Forest after Cross Validation

tic()
set.seed(2)
control <- trainControl(method="cv", number = 5)
grid <- data.frame(mtry = c(1, 5, 10, 25, 50, 100))
train_rf_caret_cv <- train(Type ~ .,
                           method = "rf",
                           ntree = 150,
                           trControl = control,
                           tuneGrid = grid,
                           data = train_set)

ggplot(train_rf_caret_cv)
```




```
train_rf_caret_cv$bestTune
```

```
##  mtry
## 6  100
pred_rf_caret_cv <- predict(train_rf_caret_cv,
                            test_set,
                            type = "raw")
acc_rf_caret_cv <- confusionMatrix(pred_rf_caret_cv,
                                   test_set$Type)$overall["Accuracy"]

acc_rf_caret_cv
##  Accuracy
## 0.9532374
time_rf_caret_cv <- toc()
## 279.19 sec elapsed
```

The model provides an accuracy of 0.9532374 with an elapsed time of 279.19 seconds.

3.3.5 Neural Network

3.3.5.1 Neural Network Model Using CARET

The Neural Network model uses the **train** function of the **CARET** package with the **"nnet"** method to train the model. The model is then tested with the test set. Predictions are made using the **predict** function and the overall accuracy is calculated.

```
############################    Neural Network    ###############

#Model 8: Neural Network Model using caret::train with "nnet" method

tic()
set.seed(2)
library(nnet)

train_nn <- train(Type ~ .,
                  method = "nnet",
                  data = train_set)

pred_nn <- predict(train_nn,
                   test_set,
                   type = "raw")
```

```
acc_nn <- confusionMatrix(pred_nn,
                          test_set$Type)$overall[["Accuracy"]]

acc_nn

time_nn <- toc()
```

The Neural Network Model has provided us with an accuracy of 0.8920863 with an elapsed time of 239.94 seconds.

3.4 Results

In this project, a total of 8 models have been fit to recognize the emotions in the Arabic conversation using the Arabic Natural Audio Dataset (ANAD). Each model has performed differently. The performance or the efficacy of the model has been measured in terms of Accuracy. The different models and the corresponding accuracies are given in the following table:

Results of Analysis

Model	Accuracy
kNN Using CARET after CV	0.9136691
Logistic Regression Using CARET after CV	0.9730769
Logistic Regression after PCA	0.9248120
Support Vector Machines with CARET	0.9568345
Support Vector Machines with e1071	0.9604317
Random Forests Using randomForest	0.9604317
Random Forests Using CARET after CV	0.9532374
Neural Network Using CARET	0.8920863

4. Conclusion

The Logistic Regression Using CARET after CV model fits well with the data. The validation has given us an accuracy of 97.31%, which is the highest among the models examined. It is therefore concluded that the Logistic Regression Using CARET after CV

model is the appropriate model for recognizing or predicting emotions given the required acoustic features.

References

1. Enhancement of an Arabic Speech Emotion Recognition System by Samira Klaylat, Ziad Osman, Lama Hamandi, Lama Hamandi and Rafik Hariri International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 5 (2018) pp. 2380-2389 https://www.ripublication.com/ijaer18/ijaerv13n5_38.pdf
2. Arabic Natural Audio Dataset https://data.mendeley.com/datasets/xm232yxf7t/1, Published: 30 May 2018 | Version 1 | DOI: 10.17632/xm232yxf7t.1, Contributor(s): Samira klaylat, ziad Osman, Rached Zantout, Lama Hamandi
3. http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/112-pca-principal-component-analysis-essentials/
4. Implementation of Neural Network and feature extraction to classify ECG signals by R karthik, Dhruv Tyagi, Amogh Raut, Soumya Saxena, Rajesh Kumar M, Senior IEEE Member, Department of Electronics and Communication Engineering VIT University Vellore, India 632014 https://arxiv.org/ftp/arxiv/papers/1802/1802.06288.pdf
5. Best way to learn kNN Algorithm using R Programming by Analytics Vidhya at https://www.analyticsvidhya.com/blog/2015/08/learning-concept-knn-algorithms-programming/
6. Logistic Regression in R Tutorial at https://www.datacamp.com/community/tutorials/logistic-regression-R
7. Classifying data using Support Vector Machines(SVMs) in R at https://www.geeksforgeeks.org/classifying-data-using-support-vector-machinessvms-in-r/
8. Introduction to Data Science - Data Analysis and Prediction Algorithms with R by Prof. Rafael A. Irizarry (2019-03-27) https://rafalab.github.io/dsbook/
9. https://gist.github.com/primaryobjects/911b50e9cbb2c2a2137b
10. The CARET package, Max Khun at http://topepo.github.io/caret/index.html
11. https://www.r-bloggers.com/creating-nice-tables-using-r-markdown/