US data

Emmanuelle

February 2019

Report

US presidential speech analysis

1. Executive summary

For this project, we will use the “Presidential Debates” dataset (Data_US) composed of 14 variables. The objective is to predict if a candidate will win or lose a speech.

In order reach our goal, we will create a KNN model, which is non parametric supervised machine learning algorithm used for classification and regression. It calculates similarity amongst observations based on a distance function (usually Euclidean).

We will first divide the dataset into two subsets:

- a training subset to train our algorithm, called “train_set” (70%)
- a validation subset to predict the movie ratings, called “validation_set” (30%)

In the first part of the report, we will use techniques such as data exploration and visualization to have an overview of the dataset.

Then, we will train the KNN model and predict the values accordingly.

Finally, we will explain the results and conclude.

All the analysis will be made through R studio and the following packages: caret, e1071,dplyr and tidyverse.

2. Analysis: data description, preparation, exploration and visualization

2.A. Data description

In this section we will take a first look at our datasets and check if data cleaning is necessary.

Please find below the structure of the datasets:

- data_US (complete dataset): 1524 observations of 14 variables
- train_set: 1068 observations of 14 variables
- validation_set: 456 observations of 14 variables

```r
dim(data_US)

## [1] 1524   14

dim(train_set)

## [1] 1068   14

dim(validation_set)

## [1] 456  14
```

It seems that there is no missing data and no data cleaning necessary. However we need to set levels for both training and validation data.

```r
# Setting levels for both training and validation data
levels(train_set$Win.Loss) <- make.names(levels(factor(train_set$Win.Loss)))
levels(validation_set$Win.Loss) <-
make.names(levels(factor(validation_set$Win.Loss)))
```

Please find below the 14 variables: - "Win.Loss": result of the speech - "Optimism" - "Pessimism" - "PastUsed" - "FutureUsed" - "PresentUsed" - "OwnPartyCount" - "OppPartyCount" - "NumericContent" - "Extra" - "Emoti" - "Agree" - "Consc" - "Openn".

In our analysis "Win.Loss" is the dependant variable whereas the 13 others are the independant variables.

Each independant variable represents the use of specific words or expressions in the speech.

2.B. Data exploration

Let's analyse the structure of the 14 variables:

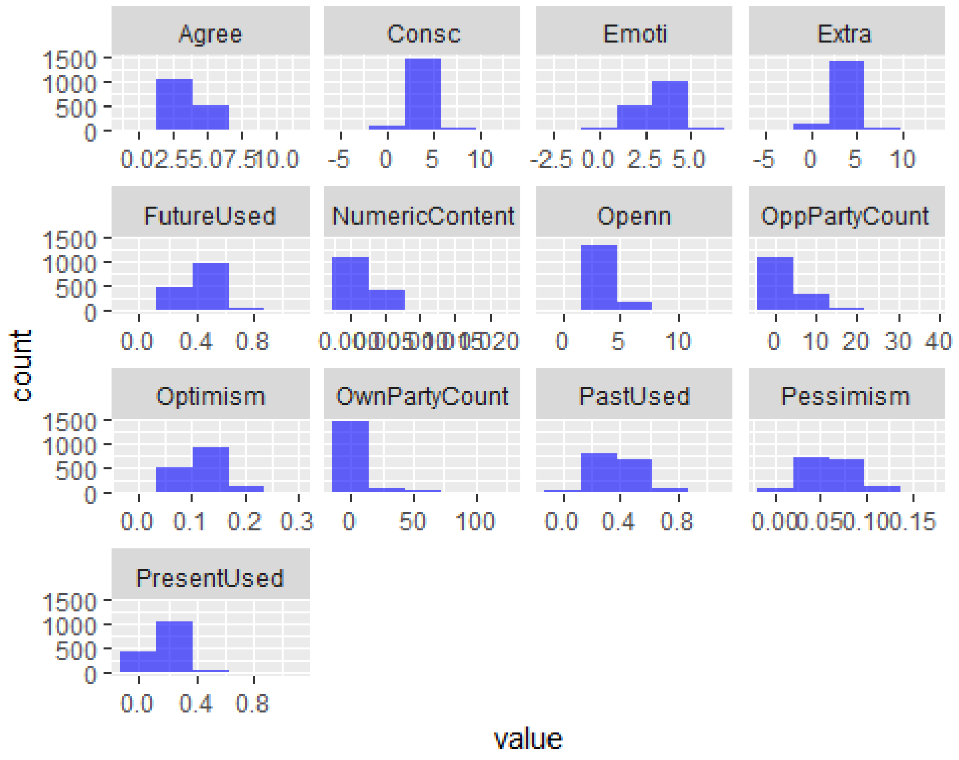
```r
summary(data_US$Win.Loss)

##   0   1
## 595 929

summary(data_US$Optimism)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.09652 0.11585 0.12089 0.14188 0.27132

summary(data_US$Pessimism)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.04448 0.05848 0.06012 0.07425 0.15730

summary(data_US$PastUsed)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.3000  0.3667  0.3834  0.4494  1.0000
```

```
summary(data_US$Futureused)

## Length  Class   Mode
##      0   NULL   NULL

summary(data_US$PresentUsed)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1212  0.1771  0.1871  0.2425  1.0000

summary(data_US$OwnPartyCount)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   3.302   1.000 115.000

summary(data_US$OppPartyCount)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   3.373   5.000  35.000

summary(data_US$NumericContent)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000000 0.001110 0.001775 0.002133 0.002764 0.020752

summary(data_US$Extra)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -3.965   3.184   3.804   3.676   4.337  11.725

summary(data_US$Emoti)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -1.349   2.704   3.272   3.194   3.752   6.468

summary(data_US$Agree)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -1.113   3.279   3.659   3.681   4.087   9.392

summary(data_US$Consc)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -5.048   3.025   3.586   3.543   4.130  10.407

summary(data_US$Win.Openn)

## Length  Class   Mode
##      0   NULL   NULL
```

2.C. Data visualization

Please find below the graphics of the 13 independant variables

```r
ggplot(gather(data_US[,2:ncol(data_US)]), aes(value)) +
  geom_histogram(bins = 5, fill = "blue", alpha = 0.6) +
  facet_wrap(~key, scales = 'free_x')
```



Thanks to the preparation and exploration of the dataset, we are now ready to create our machine learning model.

2.D. Modelling approach

The objective is to define a method that will allow us to train several algorithms in order to identify the best one.

We will proceed in five steps:

- Setting up train controls
- Training of the KNN model (on train_set)
- Predictions (on the validation_set)
- Analysis of the results.
- Conclusion

3. Data analysis and results

Data partition

In order to reach our goal, we divided the Data_US dataset into two subsets:

- the training subset to train our algorithm, called “train_set” (70% of US_data)
- the validation subset to predict the result of the speech, called“validation_set” (30% of US_data)

We are now ready to create our predictive model.

Setting up train controls

Let’s set up train controls and build the KNN model.

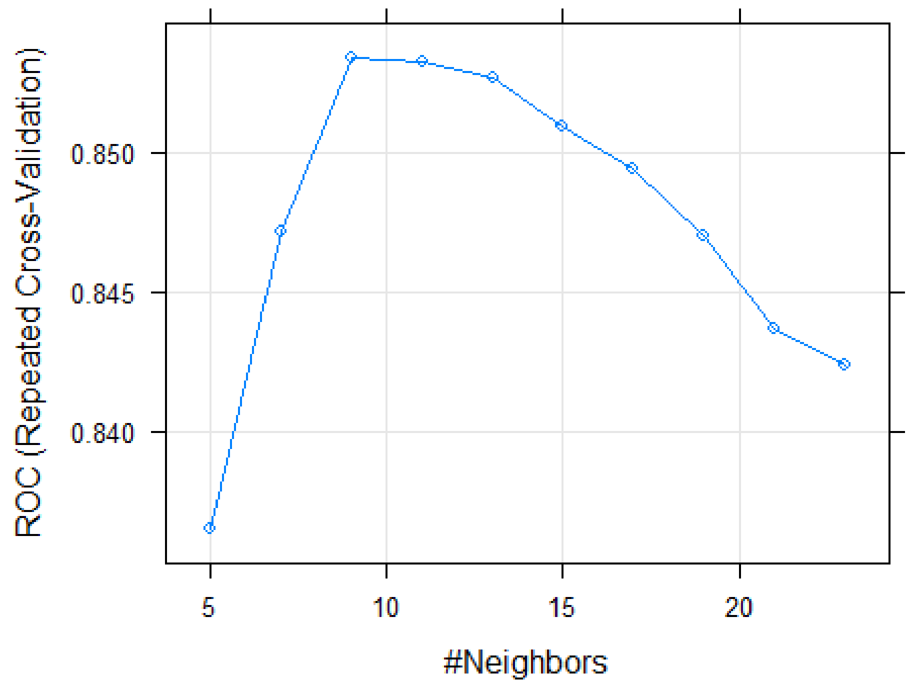
```
repeats = 3
numbers = 10
tunel = 10
set.seed(1234)
x <- trainControl(method = "repeatedcv",
                  number = numbers,
                  repeats = repeats,
                  classProbs = TRUE,
                  summaryFunction = twoClassSummary)
```

Training of the model

We are now ready to train our KNN model:

```
model_knn <- train(Win.Loss~., data= train_set, method="knn",
                preProcess=c("center","scale"),
                trControl=x,metric="ROC",
                tuneLength= tunel)

# Summary of model
model_knn

## k-Nearest Neighbors
##
## 1068 samples
##   13 predictor
##    2 classes: 'X0', 'X1'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 962, 961, 961, 961, 961, 961, ...
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    5  0.8364872  0.6900890  0.8412665
##    7  0.8471507  0.6684475  0.8494250
##    9  0.8534144  0.6587689  0.8525019
##   11  0.8532324  0.6540457  0.8602020
##   13  0.8526851  0.6531940  0.8683994
##    15  0.8509041  0.6491096  0.8607071
##    17  0.8494333  0.6411537  0.8560995
##    19  0.8470142  0.6267325  0.8612432
##    21  0.8436754  0.6146922  0.8637529
##    23  0.8423458  0.6042973  0.8714375
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```r
plot(model_knn)
```



Predictions

```r
# Validation
valid_pred <- predict(model_knn,validation_set, type = "prob")
```

Results analysis

How well does our model perform?

In this section, we are going to evaluate our algorithm thanks to:

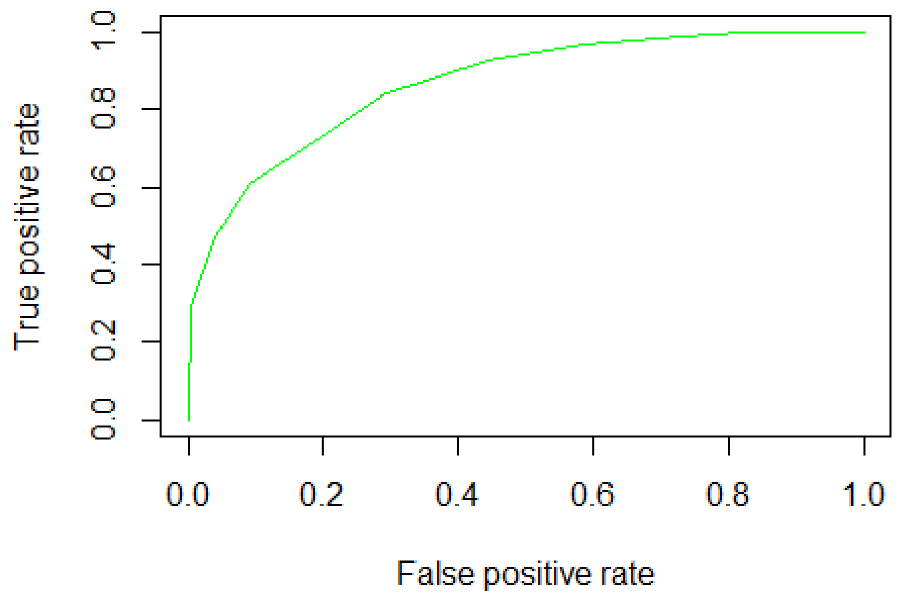
- the AUC (Area under the curve)
- the Kolmogorov-Smirnov (K-S) statistics

First let's calculate the predictions on the validation_set:

```
#Storing Model Performance Scores
library(ROCR)

## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess

pred_val <-prediction(valid_pred[,2],validation_set$Win.Loss)
```

Now let’s calculate and plot the AUC (Area Under the Curve):

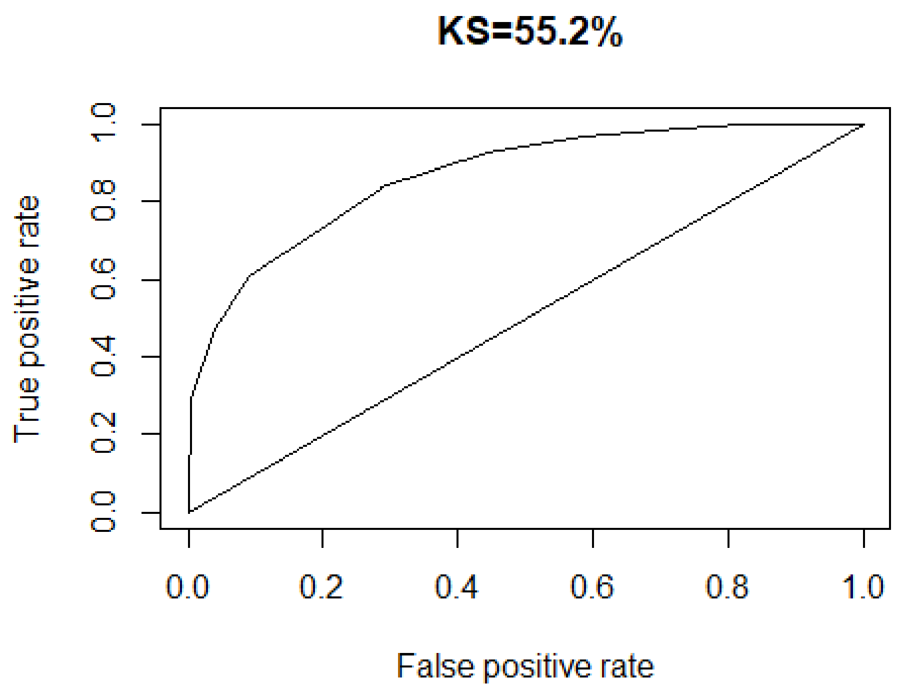
```
#Storing Model Performance Scores
library(ROCR)
pred_val <-prediction(valid_pred[,2],validation_set$Win.Loss)
# Calculating Area under Curve (AUC)
perf_val <- performance(pred_val,"auc")
perf_val

## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.8670378
##
##
## Slot "alpha.values":
## list()

# Plot AUC (x-axis: fpr, y-axis: tpr)
perf_val <- performance(pred_val, "tpr", "fpr")
plot(perf_val, col = "green", lwd = 1.5)
```



The Area under curve (AUC) on validation_set is 0.8670378.

Finally let’s calculate and plot Kolmogorov-Smirnov statistics.

```
ks <- max(attr(perf_val, "y.values")[[1]]-(attr(perf_val, "x.values")[[1]]))
plot(perf_val,main=paste0(' KS=',round(ks*100,1),'%'))
lines(x = c(0,1),y=c(0,1))
```



```
ks

## [1] 0.5516126
```

The value of K-S is 55.2%.

4. Conclusion

To conclude, we can say that the AUC of our model is quite good (0.86). Indeed, AUC value generally lies between 0.5 to 1 where 0.5 denotes a bad classifer and 1 denotes an excellent classifier.

The K-S results (55.2%) is also quite good as it is closer to 1 than to 0.

However we could continue our research in order to find a model with a better accuracy.