SVM & Naïve Bayes On Online News Popularity Prediction

Step 1: Collecting data

```
getwd()
```

```
## [1] "C:/Users/Madhu/Side Projects/Machine_Learning_Course"
```

```
setwd('C:/Users/Madhu/Side Projects/Machine_Learning_Course')

#Read data file

news <- read.csv('OnlineNewsPopularity.csv')

str(news)
```

```
## 'data.frame':    39644 obs. of  61 variables:
##  $ url                          : Factor w/ 39644 levels
"http://mashable.com/2013/01/07/amazon-instant-video-browser/",..: 1 2 3 4 5
6 7 8 9 10 ...
##  $ timedelta                    : num  731 731 731 731 731 731 731 731 731
731 ...
##  $ n_tokens_title               : num  12 9 9 9 13 10 8 12 11 10 ...
##  $ n_tokens_content             : num  219 255 211 531 1072 ...
##  $ n_unique_tokens              : num  0.664 0.605 0.575 0.504 0.416 ...
##  $ n_non_stop_words             : num  1 1 1 1 1 ...
##  $ n_non_stop_unique_tokens     : num  0.815 0.792 0.664 0.666 0.541 ...
##  $ num_hrefs                    : num  4 3 3 9 19 2 21 20 2 4 ...
##  $ num_self_hrefs               : num  2 1 1 0 19 2 20 20 0 1 ...
##  $ num_imgs                     : num  1 1 1 1 20 0 20 20 0 1 ...
##  $ num_videos                   : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ average_token_length         : num  4.68 4.91 4.39 4.4 4.68 ...
##  $ num_keywords                 : num  5 4 6 7 7 9 10 9 7 5 ...
##  $ data_channel_is_lifestyle    : num  0 0 0 0 0 0 1 0 0 0 ...
##  $ data_channel_is_entertainment: num  1 0 0 1 0 0 0 0 0 0 ...
##  $ data_channel_is_bus          : num  0 1 1 0 0 0 0 0 0 0 ...
##  $ data_channel_is_socmed       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ data_channel_is_tech         : num  0 0 0 0 1 1 0 1 1 0 ...
##  $ data_channel_is_world        : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ kw_min_min                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kw_max_min                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kw_avg_min                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kw_min_max                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kw_max_max                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kw_avg_max                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kw_min_avg                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kw_max_avg                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kw_avg_avg                   : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ self_reference_min_shares    : num  496 0 918 0 545 8500 545 545 0 0
...
##  $ self_reference_max_shares    : num  496 0 918 0 16000 8500 16000 16000
0 0 ...
##  $ self_reference_avg_sharess   : num  496 0 918 0 3151 ...
##  $ weekday_is_monday            : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ weekday_is_tuesday           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday_is_wednesday         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday_is_thursday          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday_is_friday            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday_is_saturday          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday_is_sunday            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ is_weekend                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ LDA_00                       : num  0.5003 0.7998 0.2178 0.0286 0.0286
...
##  $ LDA_01                       : num  0.3783 0.05 0.0333 0.4193 0.0288
...
##  $ LDA_02                       : num  0.04 0.0501 0.0334 0.4947 0.0286
...
##  $ LDA_03                       : num  0.0413 0.0501 0.0333 0.0289 0.0286
...
##  $ LDA_04                       : num  0.0401 0.05 0.6822 0.0286 0.8854
...
##  $ global_subjectivity          : num  0.522 0.341 0.702 0.43 0.514 ...
##  $ global_sentiment_polarity    : num  0.0926 0.1489 0.3233 0.1007 0.281
...
##  $ global_rate_positive_words   : num  0.0457 0.0431 0.0569 0.0414 0.0746
...
##  $ global_rate_negative_words   : num  0.0137 0.01569 0.00948 0.02072
0.01213 ...
##  $ rate_positive_words          : num  0.769 0.733 0.857 0.667 0.86 ...
##  $ rate_negative_words          : num  0.231 0.267 0.143 0.333 0.14 ...
##  $ avg_positive_polarity        : num  0.379 0.287 0.496 0.386 0.411 ...
##  $ min_positive_polarity        : num  0.1 0.0333 0.1 0.1364 0.0333 ...
##  $ max_positive_polarity        : num  0.7 0.7 1 0.8 1 0.6 1 1 0.8 0.5 ...
##  $ avg_negative_polarity        : num  -0.35 -0.119 -0.467 -0.37 -0.22 ...
##  $ min_negative_polarity        : num  -0.6 -0.125 -0.8 -0.6 -0.5 -0.4 -
0.5 -0.5 -0.125 -0.5 ...
##  $ max_negative_polarity        : num  -0.2 -0.1 -0.133 -0.167 -0.05 ...
##  $ title_subjectivity           : num  0.5 0 0 0 0.455 ...
##  $ title_sentiment_polarity     : num  -0.188 0 0 0 0.136 ...
##  $ abs_title_subjectivity       : num  0 0.5 0.5 0.5 0.0455 ...
##  $ abs_title_sentiment_polarity : num  0.188 0 0 0 0.136 ...
##  $ shares                       : int  593 711 1500 1200 505 855 556 891
3600 710 ...
```

Step 2: Exploring the data

```
#Using sumary to check the statistics of shares column in the dataset

summary(news$shares)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1     946    1400    3395    2800  843300
```

```
# Adding new variable Popularity based on the market share value. Considering
the median value of 1400 to make a splut in the data as popular and unpopular

news$Popularity <- ifelse( news$shares <= 1400, "Unpopular","Popular")

news$Popularity <- as.factor(news$Popularity)


#Using table to see how many were defaulted and how many were not

table(news$Popularity)
```

```
##
##   Popular Unpopular
##     19562     20082
```

Method1: Support vector machines

Creating the training and testing data set

```
set.seed(12345)

news_rand <- news[order(runif(39644)),]


# Choosing 75% for training set and remaining

news_train <- news_rand[1:29733,]
news_test <- news_rand[29734:39644,]
```

Training model on data

```
# Installing  kernlab

#install.packages('kernlab')


library(kernlab)
```

```
## Warning: package 'kernlab' was built under R version 3.5.2
```

```
news_classifier <- ksvm(Popularity ~ ., data = news_train[c(29:60,62)],
kernel = "vanilladot")
```

```
##  Setting default kernel parameters

news_classifier

## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 25136
##
## Objective Function Value : -25114.41
## Training error : 0.394377
```

There us a training error of 40%

The last and final step 4 of evaluating model performance

```
news_predictions <- predict(news_classifier, news_test)

table(news_predictions, news_test$Popularity)

##
## news_predictions Popular Unpopular
##        Popular      3353      2456
##        Unpopular    1513      2589

agreement <- news_predictions == news_test$Popularity

table(agreement)

## agreement
## FALSE  TRUE
##  3969  5942
```

Model has predicted 5942 correctly and 3970 was misclassified.

Method 2: Naive Bayes Algorithm

Training a model on the data

```
#install.packages('naivebayes')

library(naivebayes)

## Warning: package 'naivebayes' was built under R version 3.5.2

naive <- naive_bayes(Popularity ~ ., data = news_train[c(29:60,62)])
```

```
naive
```

```
## ================================ Naive Bayes
==================================
## Call:
## naive_bayes.formula(formula = Popularity ~ ., data = news_train[c(29:60,
##     62)])
##
## A priori probabilities:
##
##   Popular Unpopular
## 0.4942656 0.5057344
##
## Tables:
##
## self_reference_min_shares   Popular Unpopular
##                       mean  5082.740  3078.151
##                       sd   23296.859 17919.744
##
##
## self_reference_max_shares   Popular Unpopular
##                       mean 12599.096  8048.308
##                       sd   46651.109 33458.052
##
##
## self_reference_avg_sharess   Popular Unpopular
##                        mean  7869.228  4993.733
##                        sd   27464.198 21458.028
##
##
## weekday_is_monday   Popular Unpopular
##               mean 0.1575259 0.1746359
##               sd   0.3643082 0.3796680
##
##
## weekday_is_tuesday   Popular Unpopular
##                mean 0.1707267 0.2001064
##                sd   0.3762828 0.4000931
##
## # ... and 27 more tables
```

Step 4: Evaluating Model performance

```
naive_predict <- table(predict(naive, news_test), news_test$Popularity)
naive_predict
```

```
##
##             Popular Unpopular
##   Popular      1229       728
##   Unpopular    3637      4317
```

```
#calculate accuracy of the model

accuracy <- sum(diag(naive_predict))/sum(naive_predict)*100
accuracy
```

```
## [1] 55.95803
```

```
 prediction <- predict(naive, news_test)
 check <- prediction == news_test$Popularity
 table(check)
```

```
## check
## FALSE  TRUE
##  4365  5546
```

As seen, the two evaulation methods match.