Project Text Categorization

This Text categorization method uses 9 different analytical techniques and they are SVM(Support Vector Machines), GLMNET(Generalized Linear Model), MAXENT(Maximum Entropy), SLDA, Bagging, Boosting, RF(Random Forest), NNET(Neural Networks), TREE(Classification Tree).

Load Required Libraries

```
library(stringi)
library(qdapDictionaries)
library(caTools)
library(SparseM)
```

```
##
## Attaching package: 'SparseM'
```

```
## The following object is masked from 'package:base':
##
##     backsolve
```

```
library(RTextTools)
```

Load Data from csv file

```
My_data<-read.csv(file="C:/Users/Jayan/Documents/Ryerson - Big Data, Analytics, Predictive Analytics/CKME 136 - Capstone Pro
ject/Programming Part/My Data.csv", header=T, sep=",", na.strings=c("","NA"))
```

The Dataset is cleaned by first removing empty records then removing records which are non-english and gibberish

```
My_data<-My_data[complete.cases(My_data),]
is.word  <- function(x) x %in% GradyAugmented
My_data$Response<-tolower(My_data$Response)
split_word<-stri_extract_all_words(My_data$Response, simplify=TRUE)
split_word[split_word==""]<-NA
nonengrownums<-which(apply(split_word, 1, function(x) sum(is.word(x)/sum(!is.na(x))))<0.75)
My_data<-My_data[-nonengrownums,]
```

Randomly shuffle the dataset and making the Action class in the dataset a numeric type

```
My_data<-My_data[sample(nrow(My_data)),]
My_data$Action<-as.numeric(as.factor(My_data$Action))
```

Create the document term matrix

```
doc_matrix <- create_matrix(My_data$Response, language="english", removeNumbers=TRUE,stemWords=TRUE, removePunctuation=TRUE,
 toLower=TRUE, removeSparseTerms=.998)
```

```
## Warning in simple_triplet_matrix(i, j, v, nrow = length(terms), ncol =
## length(corpus), : bytecode version mismatch; using eval
```

Creating the container with 80% Train data and 20% Test data

```
train_size<-round(0.8*(nrow(My_data)), digits=0)
container <- create_container(doc_matrix, My_data$Action, trainSize=1:train_size, testSize=(train_size+1):(nrow(My_data)),vi
rgin=FALSE)
```

Train and classify the model using the allocated train and test sets

```
SVM <- train_model(container,"SVM")
GLMNET <- train_model(container,"GLMNET")
MAXENT <- train_model(container,"MAXENT")
SLDA <- train_model(container,"SLDA")
BOOSTING <- train_model(container,"BOOSTING")
BAGGING <- train_model(container,"BAGGING")
RF <- train_model(container,"RF")
NNET <- train_model(container,"NNET")
TREE <- train_model(container,"TREE")

SVM_CLASSIFY <- classify_model(container, SVM)
GLMNET_CLASSIFY <- classify_model(container, GLMNET)
MAXENT_CLASSIFY <- classify_model(container, MAXENT)
SLDA_CLASSIFY <- classify_model(container, SLDA)
BOOSTING_CLASSIFY <- classify_model(container, BOOSTING)
BAGGING_CLASSIFY <- classify_model(container, BAGGING)
RF_CLASSIFY <- classify_model(container, RF)
NNET_CLASSIFY <- classify_model(container, NNET)
TREE_CLASSIFY <- classify_model(container, TREE)
```

Create the analytics summary data

```
analytics <- create_analytics(container,
                              cbind(SVM_CLASSIFY, SLDA_CLASSIFY,
                                    BOOSTING_CLASSIFY, BAGGING_CLASSIFY,
                                    RF_CLASSIFY, GLMNET_CLASSIFY,
                                    NNET_CLASSIFY, TREE_CLASSIFY,
                                    MAXENT_CLASSIFY))
summary(analytics)
```

```
## ENSEMBLE SUMMARY
##
##        n-ENSEMBLE COVERAGE n-ENSEMBLE RECALL
## n >= 1                1.00              0.74
## n >= 2                1.00              0.74
## n >= 3                1.00              0.74
## n >= 4                0.99              0.74
## n >= 5                0.98              0.75
## n >= 6                0.94              0.76
## n >= 7                0.87              0.79
## n >= 8                0.79              0.81
## n >= 9                0.57              0.83
##
##
## ALGORITHM PERFORMANCE
##
##        SVM_PRECISION           SVM_RECALL           SVM_FSCORE
##              0.73000              0.61250              0.64750
##       SLDA_PRECISION          SLDA_RECALL          SLDA_FSCORE
##              0.61875              0.57125              0.58250
## LOGITBOOST_PRECISION    LOGITBOOST_RECALL    LOGITBOOST_FSCORE
##              0.65750              0.57750              0.60000
##    BAGGING_PRECISION       BAGGING_RECALL       BAGGING_FSCORE
##              0.69125              0.55125              0.60250
##    FORESTS_PRECISION       FORESTS_RECALL       FORESTS_FSCORE
##              0.68500              0.59125              0.62375
##     GLMNET_PRECISION        GLMNET_RECALL        GLMNET_FSCORE
##              0.65875              0.54500              0.58500
##       TREE_PRECISION          TREE_RECALL          TREE_FSCORE
##              0.62000              0.53375              0.55500
##   NNETWORK_PRECISION      NNETWORK_RECALL      NNETWORK_FSCORE
##              0.19250              0.29375              0.23000
## MAXENTROPY_PRECISION    MAXENTROPY_RECALL    MAXENTROPY_FSCORE
##              0.60000              0.54250              0.56250
```

Based on the algorithm performance, it can be seen that SVM, SLDA, Boosting, Bagging, Random Forest, and GLMNET vastly outperform the other algorithms in terms of precision, recall, and F-score. The most optimal algorithm that performed the best is GLMNET which had the highest precision and F-Score.

The ensemble summary above refers to whether multiple algorithms make the same prediction concerning the class of an event (i.e., did SVM and maximum entropy assign the same label to the text?). Coverage simply refers to the percentage of documents that meet the recall accuracy threshold. Considering 80% as an inter-coder reliability standard, one may be comfortable using 6 ensemble agreement with these data because we label 92% of the data with 80% accuracy.

Cross Validation using 10 fold

```
SVM <- cross_validate(container, 10, "SVM", seed=123)
```

```
## Fold 1 Out of Sample Accuracy = 0.7586207
## Fold 2 Out of Sample Accuracy = 0.75
## Fold 3 Out of Sample Accuracy = 0.7620482
## Fold 4 Out of Sample Accuracy = 0.7703927
## Fold 5 Out of Sample Accuracy = 0.7780899
## Fold 6 Out of Sample Accuracy = 0.7822086
## Fold 7 Out of Sample Accuracy = 0.8402367
## Fold 8 Out of Sample Accuracy = 0.7426036
## Fold 9 Out of Sample Accuracy = 0.7660167
## Fold 10 Out of Sample Accuracy = 0.7616099
```

```
MAXENT <- cross_validate(container, 10, "MAXENT", seed=123)
SLDA <- cross_validate(container, 10, "SLDA",seed=123)
```

```
## Fold 1 Out of Sample Accuracy = 0.7356322
## Fold 2 Out of Sample Accuracy = 0.7142857
## Fold 3 Out of Sample Accuracy = 0.7349398
## Fold 4 Out of Sample Accuracy = 0.7673716
## Fold 5 Out of Sample Accuracy = 0.761236
## Fold 6 Out of Sample Accuracy = 0.7668712
## Fold 7 Out of Sample Accuracy = 0.8047337
## Fold 8 Out of Sample Accuracy = 0.7189349
## Fold 9 Out of Sample Accuracy = 0.735376
## Fold 10 Out of Sample Accuracy = 0.74613
```

```
BAGGING <- cross_validate(container, 10, "BAGGING",seed=123)
```

```
## Fold 1 Out of Sample Accuracy = 0.7442529
## Fold 2 Out of Sample Accuracy = 0.7440476
## Fold 3 Out of Sample Accuracy = 0.7560241
## Fold 4 Out of Sample Accuracy = 0.7643505
## Fold 5 Out of Sample Accuracy = 0.761236
## Fold 6 Out of Sample Accuracy = 0.8128834
## Fold 7 Out of Sample Accuracy = 0.8106509
## Fold 8 Out of Sample Accuracy = 0.7011834
## Fold 9 Out of Sample Accuracy = 0.7437326
## Fold 10 Out of Sample Accuracy = 0.7430341
```

```
BOOSTING <- cross_validate(container, 10, "BOOSTING",seed=123)
```

```
## Fold 1 Out of Sample Accuracy = 0.7873563
## Fold 2 Out of Sample Accuracy = 0.7916667
## Fold 3 Out of Sample Accuracy = 0.8042169
## Fold 4 Out of Sample Accuracy = 0.8217523
## Fold 5 Out of Sample Accuracy = 0.8230337
## Fold 6 Out of Sample Accuracy = 0.8558282
## Fold 7 Out of Sample Accuracy = 0.887574
## Fold 8 Out of Sample Accuracy = 0.7721893
## Fold 9 Out of Sample Accuracy = 0.7883008
## Fold 10 Out of Sample Accuracy = 0.7894737
```

```
RF <- cross_validate(container, 10, "RF",seed=123)
```

```
## Fold 1 Out of Sample Accuracy = 0.7787356
## Fold 2 Out of Sample Accuracy = 0.7559524
## Fold 3 Out of Sample Accuracy = 0.7620482
## Fold 4 Out of Sample Accuracy = 0.7794562
## Fold 5 Out of Sample Accuracy = 0.7780899
## Fold 6 Out of Sample Accuracy = 0.791411
## Fold 7 Out of Sample Accuracy = 0.8284024
## Fold 8 Out of Sample Accuracy = 0.7307692
## Fold 9 Out of Sample Accuracy = 0.7576602
## Fold 10 Out of Sample Accuracy = 0.7585139
```

```
NNET <- cross_validate(container, 10, "NNET",seed=123)
```

```
## Fold 1 Out of Sample Accuracy = 0.5747126
## Fold 2 Out of Sample Accuracy = 0.5625
## Fold 3 Out of Sample Accuracy = 0.6536145
## Fold 4 Out of Sample Accuracy = 0.6374622
## Fold 5 Out of Sample Accuracy = 0.6292135
## Fold 6 Out of Sample Accuracy = 0.6564417
## Fold 7 Out of Sample Accuracy = 0.7100592
## Fold 8 Out of Sample Accuracy = 0.6035503
## Fold 9 Out of Sample Accuracy = 0.632312
## Fold 10 Out of Sample Accuracy = 0.6408669
```

```
TREE <- cross_validate(container, 10, "TREE",seed=123)
```

```
## Fold 1 Out of Sample Accuracy = 0.6609195
## Fold 2 Out of Sample Accuracy = 0.6636905
## Fold 3 Out of Sample Accuracy = 0.7018072
## Fold 4 Out of Sample Accuracy = 0.7069486
## Fold 5 Out of Sample Accuracy = 0.738764
## Fold 6 Out of Sample Accuracy = 0.7576687
## Fold 7 Out of Sample Accuracy = 0.7455621
## Fold 8 Out of Sample Accuracy = 0.683432
## Fold 9 Out of Sample Accuracy = 0.6740947
## Fold 10 Out of Sample Accuracy = 0.6749226
```