**A SENTIMENT ANALYSIS CASE STUDY: APPLE COMPUTER TWEETS**

This tutorial will provide a free-of-programming guide on how to do a sentiment analysis using SAS Enterprise Miner (pages 1-15). You can customize if you have some SAS programming knowledge. A tutorial in R programming is also attached for reference (pages 16-25).

**A. Technical settings, data description, task, & outline of methods:**

**1. Technical setting: SAS Enterprise Miner 14.2, SAS Text Miner 14.2**

**2. Data set:** the data set can be downloaded from this site [**https://www.kaggle.com/c/apple-computers-twitter-sentiment2**](https://www.kaggle.com/c/apple-computers-twitter-sentiment2)(you should have a kaggle account).

This is the look into the sentiment around the Apple computers on tweets, containing #AAPL, @apple, etc. Tweets can be positive, neutral or negative.

**Data Description:** In .csv files they are identified as: Positive (5); Neutral (3); Negative (1).

**File descriptions:**

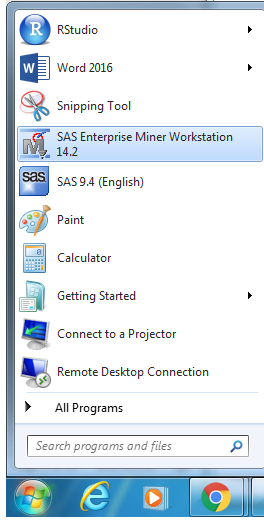
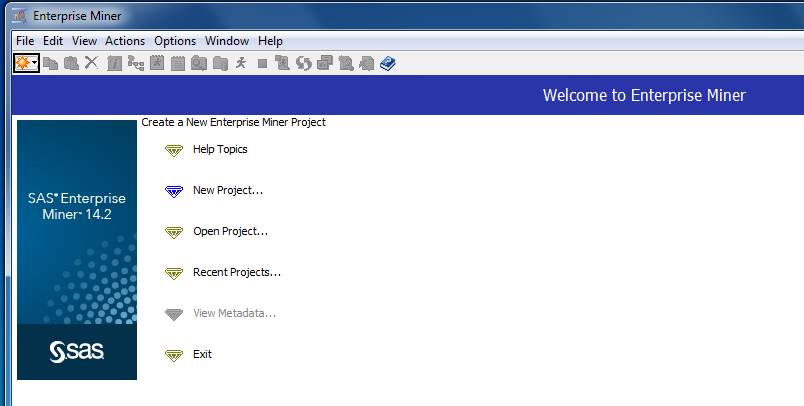
* **train.csv** - the training set: 1887 rows of tweets (608 negative tweets, 1061 neutral tweets, 218 positive tweets), the file’s data fields only have ID & Sentiment.
* **test.csv** - the test set: 1000 rows of tweets whose sentiments are unidentified, data fields include ID & Text
* **sample.csv** - a sample submission file in the correct format, data fields include ID & Sentiment. This is to submit onto kaggle.com to get the result of your predictions of the test data set.

**Data fields: id** - the id of twitter; **sentiment** - sentiment of tweet; **text** - tweet text.

**3. The task** is to identify whether the tweet (in the test set) about Apple computer is positive (5), neutral (3) or negative (1). If the tweet isn't about Apple, identification should be neutral.

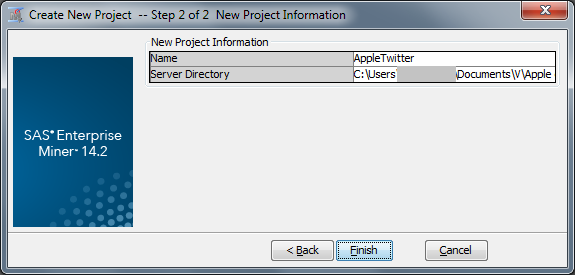
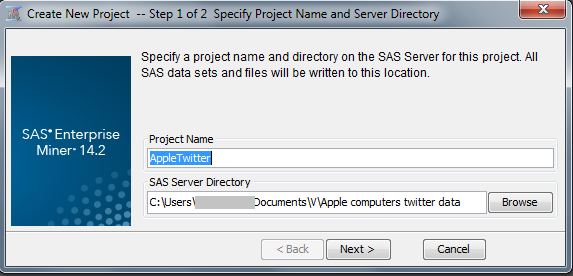
**4. Outline of methods to be used:** In this tutorial, first we explore the power of Text Rule Builder node. Second we build models based on the clusters and topics generated from the train data set. These models will be applied to the test data set to predict or classify the sentiments of the test set.

**B. The First Method − Using SAS Text Rule Builder to predict sentiments:**

** **

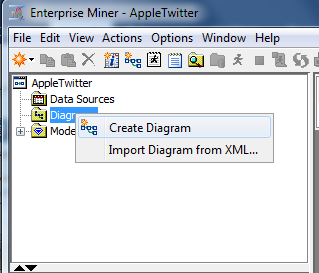
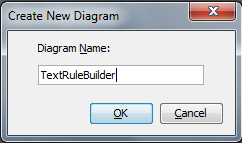
***Fig. 1.a & b.:*** *open SAS Enterprise Miner Workstation 14.2 from Windows icon*

*Step 1:* After opening SAS Enterprise Miner, click on the tab File > New > Project to open a new project. Name it as *AppleTwitter*, or any name you like by filling in the box as following. You should specify a directory that you project is based on by clicking ‘Browse’ associated with the box. You should create a folder beforehand for this step, where you store the data. Click Finish at Step 2.



***Fig. 1.c & 1.d:*** *steps to create a new project in SAS Enterprise Miner*

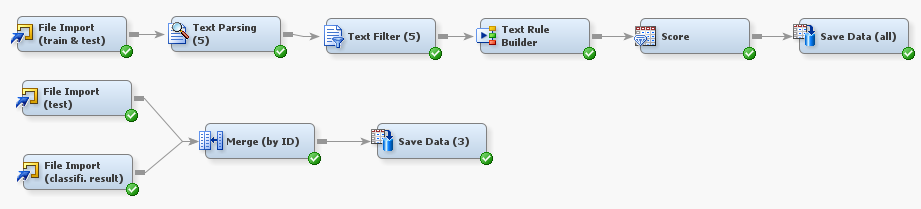
*Step 2:* On the small window located in the top left, right-click on Diagram > Create Diagram to create a new diagram for your project. Name the diagram as *TextRuleBuilder* in the text box appeared. Click OK.

***Fig. 2.a & 2.b:*** *Creating a new diagram*

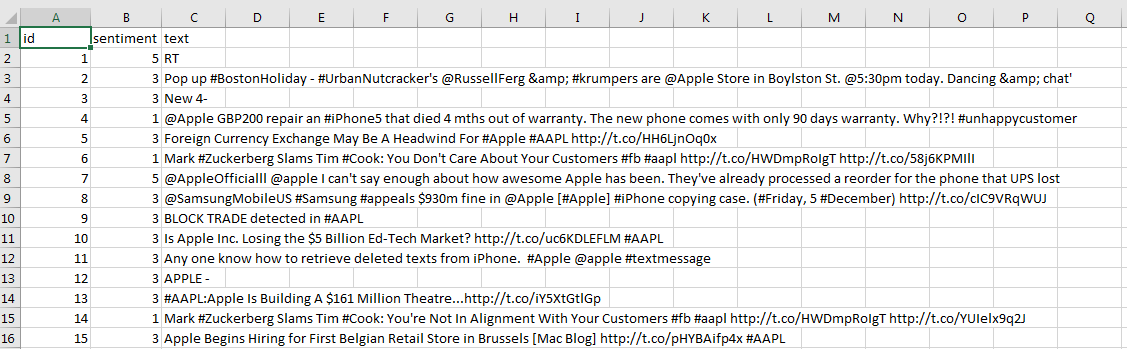
**Create the diagram:**

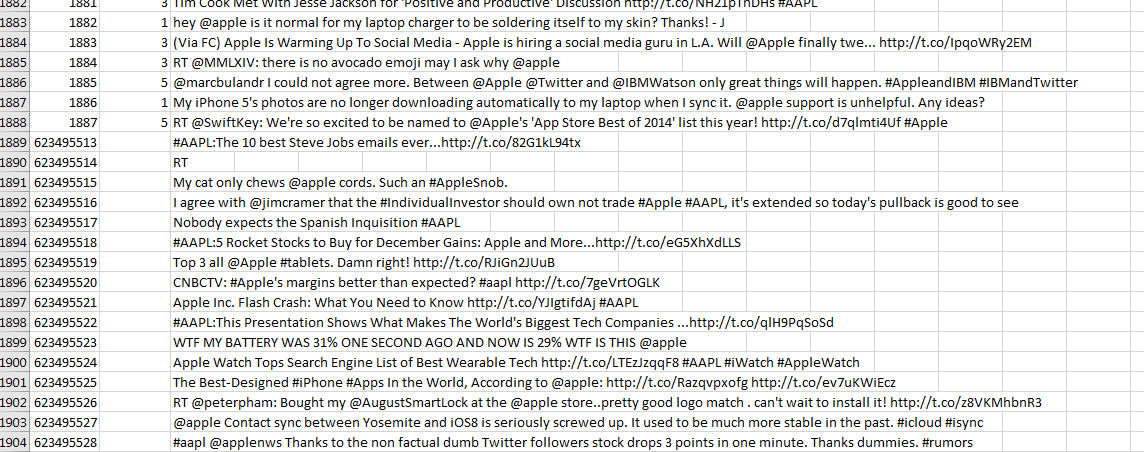
The final diagram should be like this:



***Fig. 3:*** *diagram of the Text Rule Builder model*

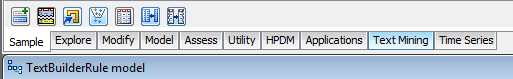
*Step 3: Data preparation:* Create a new .csv data file by concatenating all observations of the test data file to all observations of the train data file. Name the file as ‘alldata.csv’. This file should have 3 columns: id, sentiment, text. The train data file already has text and sentiment while the test data file has id and text. In the combined data, the sentiment values of all the testing observations should be blank, the id values of all the training observations should be added, from 1-1887.



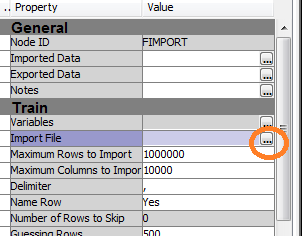
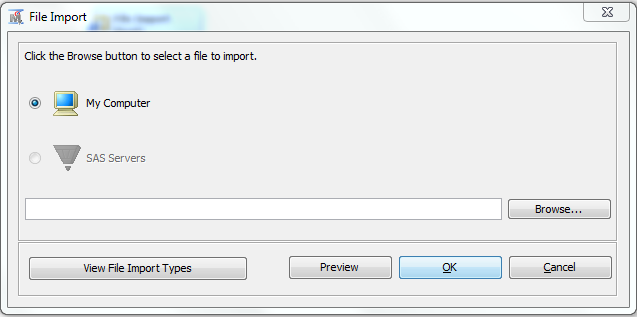


***Fig. 4:*** *data preparation step*

*Step 4:* Drag the icon ‘File Import’ from the ‘Sample’ tab to the diagram editor window and rename it as *File Import (train & test)*. This node will import your combined data set. Left-click on a small icon (with three dots) in the right of the Import File row on the Property window to specify the combined data file location.

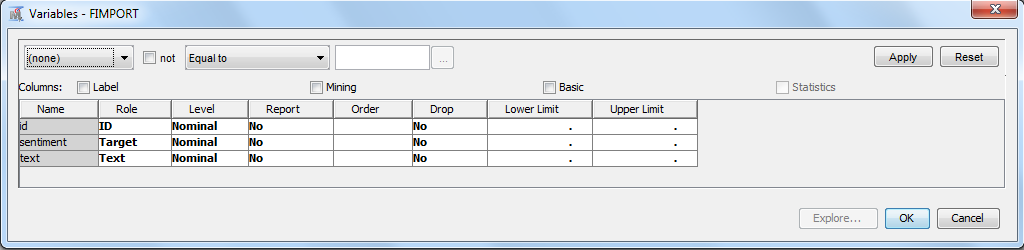


***Fig.5:******Sample*** *tab which includes node icons for creating a diagram*

***Fig. 6.a:*** *Property window of the File Import node;* ***Fig.6.b:*** *specifying the location of the imported file*

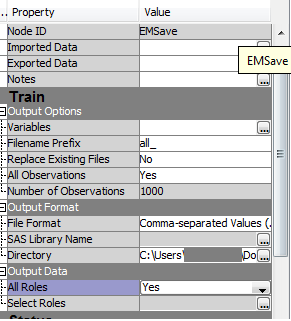
Right-click on the File Import node to choose ‘Edit Variables’, set the Role of ‘sentiment’ variable to ‘Target’, as it is to be the response variable.



***Fig.7:*** *setting roles for variables from the imported file:* ***Sentiment*** *should bet set to* ***Target***

*Step 5:* Drag the icons ‘**Text Parsing’**, ‘**Text Filter**’, ‘**Text Rule Builder**’ from the tab ‘*Text Mining*’, ‘**Score**’ from the tab ‘*Access*’, ‘**Save**’ from the tab ‘*Utility*’ into the diagram editor. Connect all these nodes together and rename them as the above diagram shown.

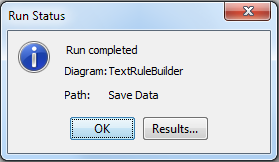
*Step 6:* Click on the ‘Save’ node to set up parameters, on your right side there appears a property window upon clicking. File name prefix should be ‘all\_’. All Roles: yes. File format: csv. Directory should be specified.



***Fig.8:*** *Property window of the* ***Save Data*** *node: specifying parameters*

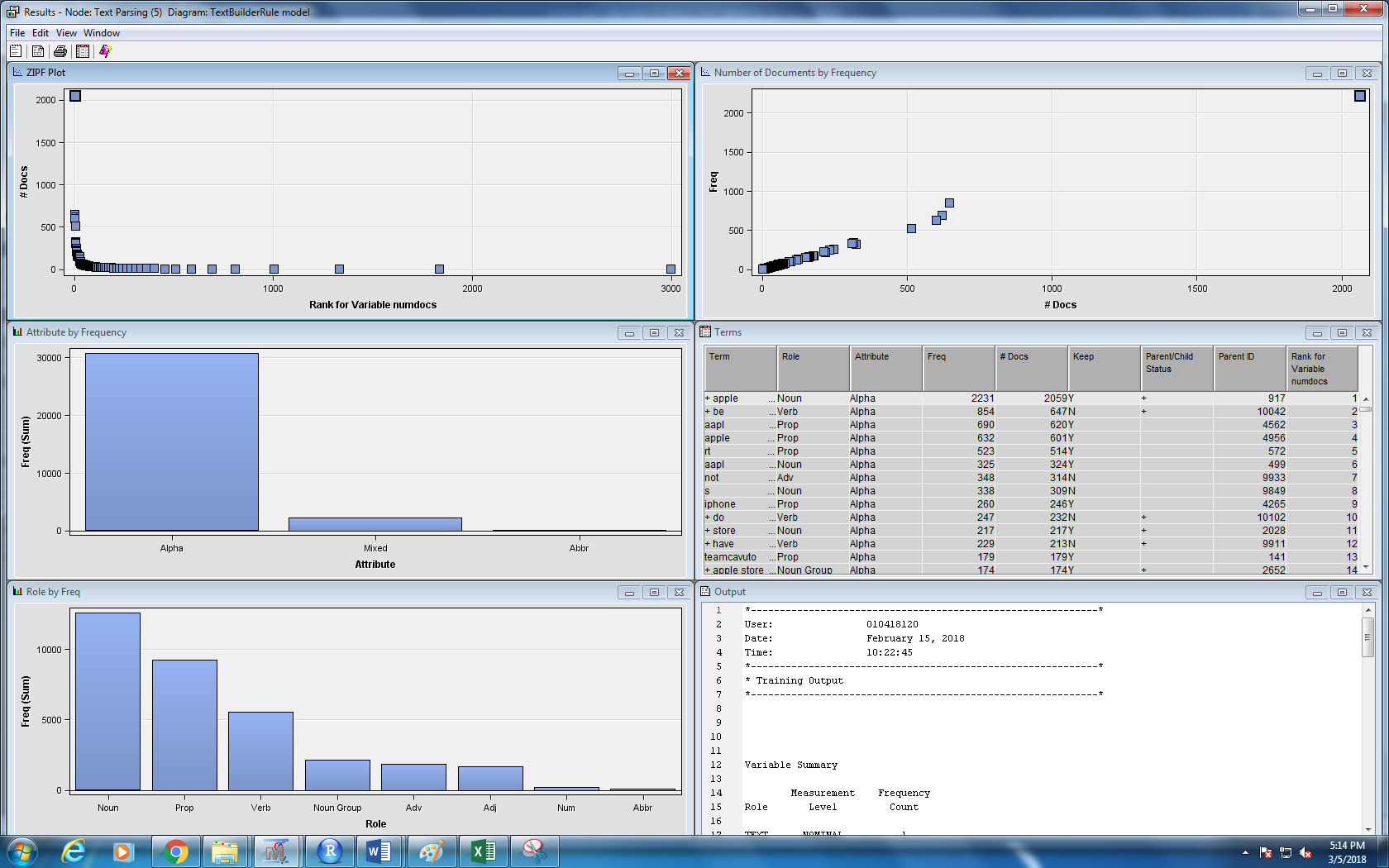
If you click on each of these nodes ‘Text Parsing’, ‘Text Filter’, ‘Text Rule Builder’, or ‘Score’, a corresponding Property Windows will appear on the left. There you can set up parameters for these nodes. However, we will use the SAS default setting for these nodes.

*Step 7:* Next, right-click on ‘Save’ node and click ‘Run’.



***Fig.9:*** *After the run completed, click ‘Results’ to see the results*

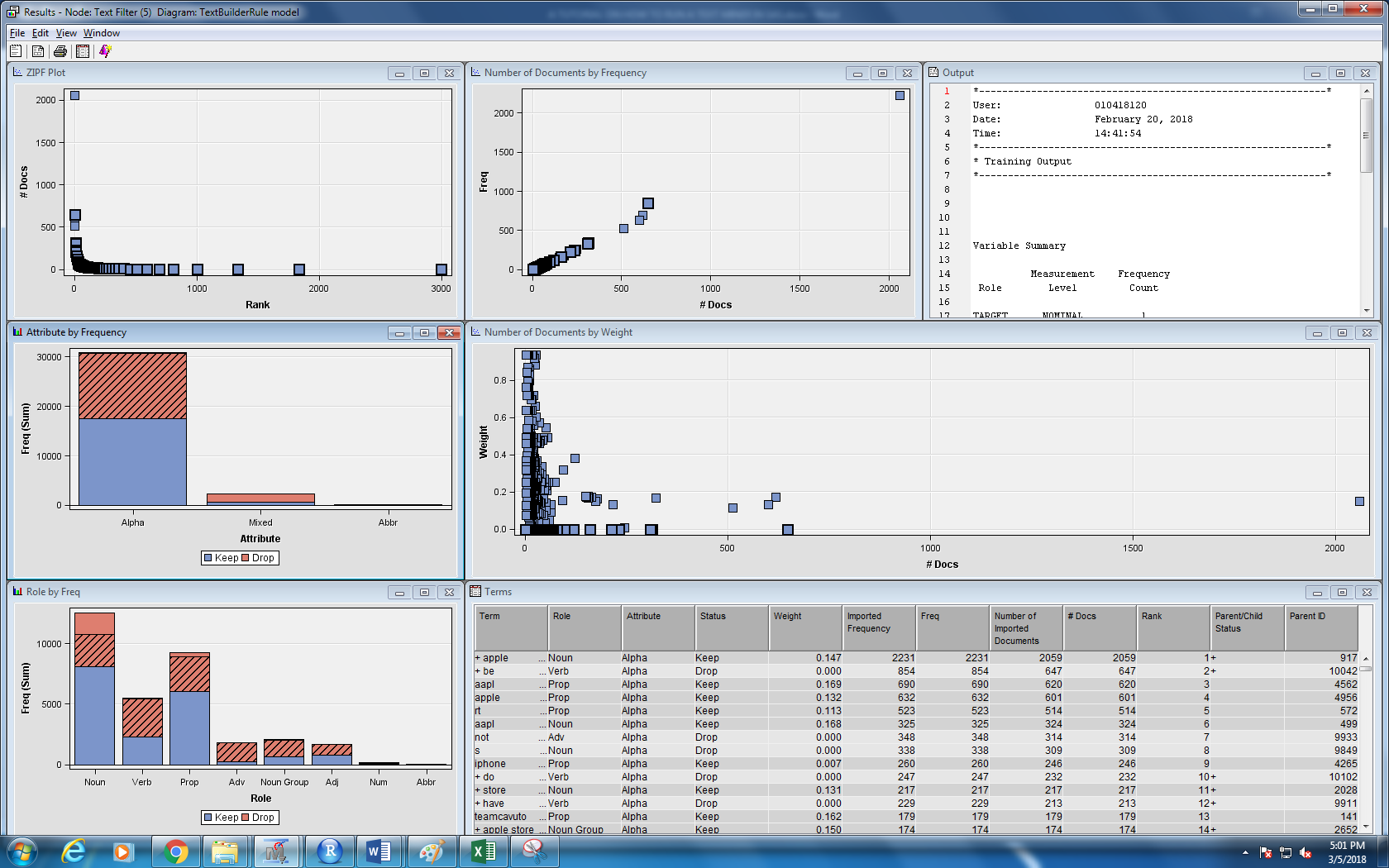
Results of Text Parsing node:



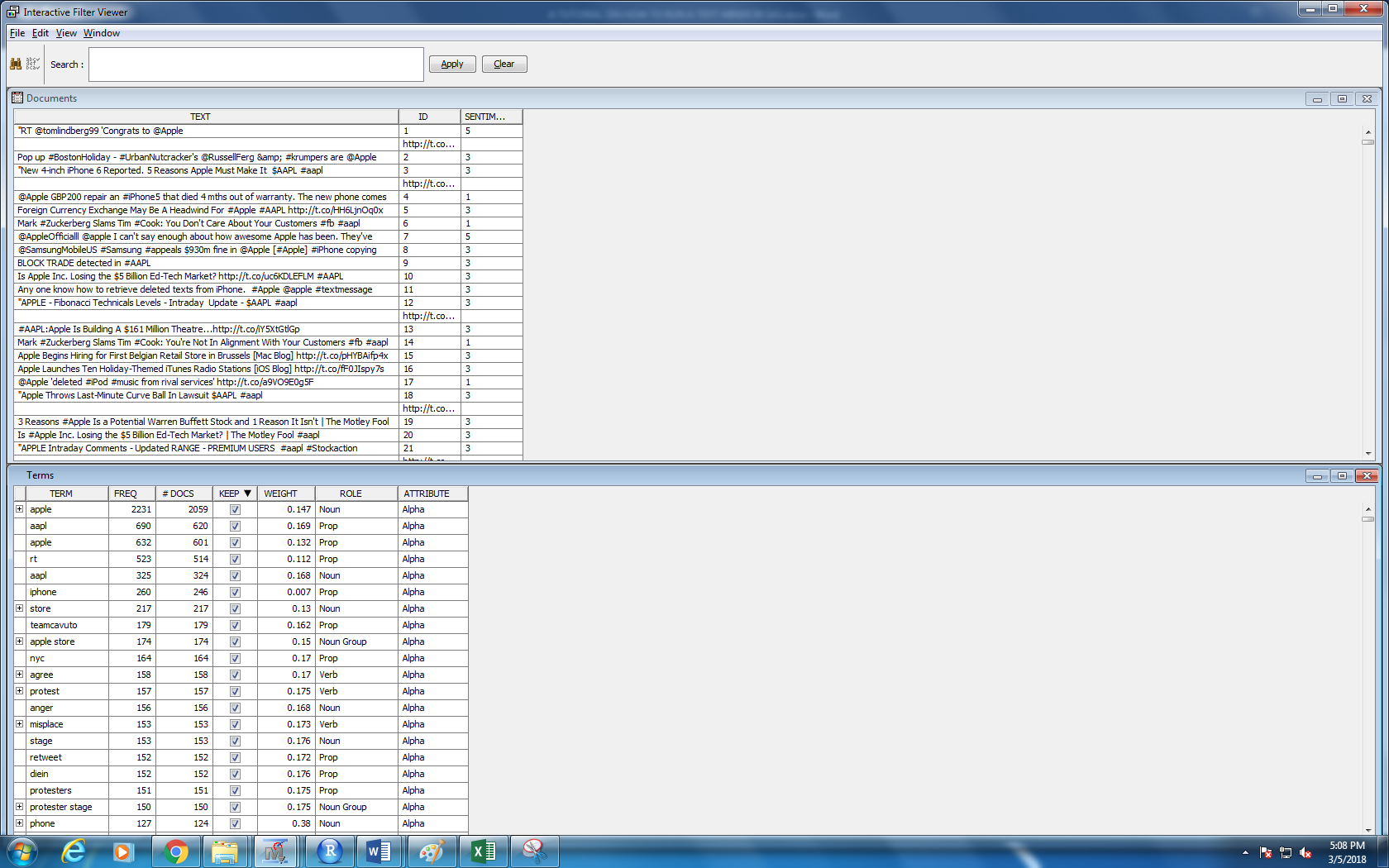
***Fig.10:*** *Results of Text Parsing node*

Results of the Text Filter node:

*Step 8:* If you click to see the Results from the Text Filter node, you will see which words are dropped or kept by the procedure. The results come with seven windows, which are the terms window with its associated graph windows. Maximizing the Terms window, you will see in details which terms are dropped or kept. It is done by using the default dictionary built-in in SAS. There are more for you to play with this node. If you have some linguistic knowledge, you can modify this manually to change the rule. For example, by clicking the Interactive Filter Viewer of the Property windows of this node, you can click/unclick on each word(s) to keep or drop the word(s).



***Fig.11:*** *Results of Text Filter node*



***Fig. 12:*** *clicking the Interactive Filter Viewer of the Property windows of the Text Filter node to see which words are kept or dropped*

What does the Text Rule Builder generate?

*Step 9:* It creates a set of key words used to classify which text is positive, negative, or neutral. By right-clicking on the node ‘Text Rule Builder’ and then ‘Results’, you will see 5 result windows. Maximize the ‘Rules Obtained’ window, you will see what set of key words determines each kind of sentiment.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Target Value | Rule # | Rule | Precision | Recall | F1 score | True Positive/Total |
| 1 | 25 | shit | 1 | 0.060855263 | 0.114728682 | 37/37 |
| 1 | 26 | charger | 0.985294118 | 0.110197368 | 0.198224852 | 34/35 |
| 1 | 27 | fuck | 0.988505747 | 0.141447368 | 0.247482014 | 27/27 |
| 1 | 28 | hate | 0.990196078 | 0.166118421 | 0.284507042 | 16/16 |
| 1 | 29 | fix | 0.983193277 | 0.192434211 | 0.321870702 | 20/21 |
| 1 | 30 | stop | 0.977777778 | 0.217105263 | 0.355316285 | 15/16 |
| 1 | 31 | phone | 0.920212766 | 0.284539474 | 0.434673367 | 58/75 |
| 1 | 32 | fuck | 0.92 | 0.302631579 | 0.455445545 | 18/19 |
| 1 | 33 | fix | 0.91943128 | 0.319078947 | 0.473748474 | 15/16 |
| 1 | 34 | hell | 0.922374429 | 0.332236842 | 0.488512696 | 9/9 |
| 1 | 35 | cord | 0.924778761 | 0.34375 | 0.501199041 | 8/8 |
| 1 | 36 | suck | 0.927350427 | 0.356907895 | 0.51543943 | 10/11 |
| 1 | 37 | apple & ~aapl & ~hire & ~aapl & ~workshop & update | 0.925925926 | 0.370065789 | 0.528789659 | 11/12 |
| 1 | 38 | work | 0.903345725 | 0.399671053 | 0.554161916 | 21/30 |
| 1 | 39 | battery | 0.902527076 | 0.411184211 | 0.564971751 | 15/16 |
| 1 | 40 | fail | 0.903914591 | 0.417763158 | 0.571428571 | 7/7 |
| 1 | 41 | upgrade | 0.902777778 | 0.427631579 | 0.580357143 | 6/7 |
| 1 | 42 | wtf | 0.898648649 | 0.4375 | 0.588495575 | 9/11 |
| 1 | 43 | yall | 0.897350993 | 0.445723684 | 0.595604396 | 9/10 |
| 1 | 44 | dear | 0.896103896 | 0.453947368 | 0.602620087 | 12/14 |
| 1 | 45 | first murder | 0.897435897 | 0.460526316 | 0.608695652 | 4/4 |
| 1 | 46 | suck | 0.898734177 | 0.467105263 | 0.614718615 | 10/10 |
| 1 | 47 | life | 0.892307692 | 0.476973684 | 0.621650589 | 11/14 |
| 1 | 48 | os | 0.893292683 | 0.481907895 | 0.626068376 | 4/4 |
| 1 | 49 | ass | 0.894259819 | 0.486842105 | 0.630457934 | 9/9 |
| 1 | 50 | apple & ~aapl & ~hire & ~workshop & ~aapl & ~rt | 0.569086651 | 0.799342105 | 0.664842681 | 442/925 |
| 3 | 13 | intraday | 1 | 0.042412818 | 0.081374322 | 45/45 |
| 3 | 14 | hire | 0.972972973 | 0.067860509 | 0.126872247 | 27/29 |
| 3 | 15 | december | 0.953271028 | 0.096135721 | 0.174657534 | 30/33 |
| 3 | 16 | diein | 0.888030888 | 0.216776626 | 0.348484848 | 128/152 |
| 3 | 17 | aapl | 0.847972973 | 0.473138549 | 0.60738052 | 311/375 |
| 3 | 18 | apple & ceo | 0.850746269 | 0.483506126 | 0.616586538 | 12/12 |
| 3 | 19 | future | 0.848673947 | 0.512723845 | 0.639247944 | 31/38 |
| 3 | 20 | radioshack | 0.850308642 | 0.519321395 | 0.644821533 | 9/9 |
| 3 | 21 | motley | 0.852134146 | 0.526861451 | 0.651135702 | 8/8 |
| 3 | 22 | wozniak | 0.853695324 | 0.533459001 | 0.656612529 | 9/9 |
| 3 | 23 | antitrust | 0.855007474 | 0.539114043 | 0.661271676 | 10/10 |
| 3 | 24 | apple | 0.826582278 | 0.615457116 | 0.70556456 | 337/443 |
| 5 | 1 | love | 1 | 0.091743119 | 0.168067227 | 20/20 |
| 5 | 2 | http://t.co/d7qlmti4uf | 1 | 0.128440367 | 0.227642276 | 8/8 |
| 5 | 3 | team | 1 | 0.165137615 | 0.283464567 | 8/8 |
| 5 | 4 | nice | 0.976190476 | 0.188073394 | 0.315384615 | 5/6 |
| 5 | 5 | awesome | 0.875 | 0.224770642 | 0.357664234 | 8/14 |
| 5 | 6 | earn | 0.883333333 | 0.243119266 | 0.381294964 | 4/4 |
| 5 | 7 | investor | 0.890625 | 0.26146789 | 0.404255319 | 4/4 |
| 5 | 8 | service | 0.863013699 | 0.288990826 | 0.432989691 | 6/10 |
| 5 | 9 | stick | 0.868401266 | 0.30283945 | 0.449078451 | 3/3 |
| 5 | 10 | love | 0.835294118 | 0.325688073 | 0.468646865 | 4/7 |
| 5 | 11 | good | 0.840909091 | 0.339449541 | 0.483660131 | 3/4 |
| 5 | 12 | amp | 0.700854701 | 0.376146789 | 0.489552239 | 16/67 |

***Table 1:*** *Rules generated by the Text Rule Builder for sentiment classification*

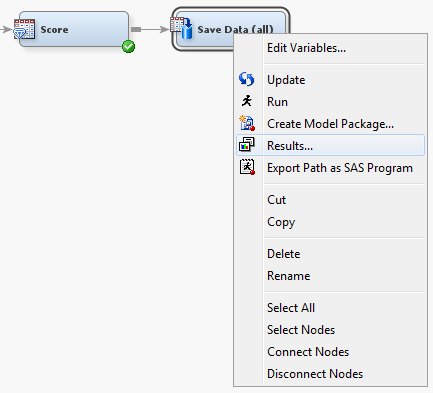
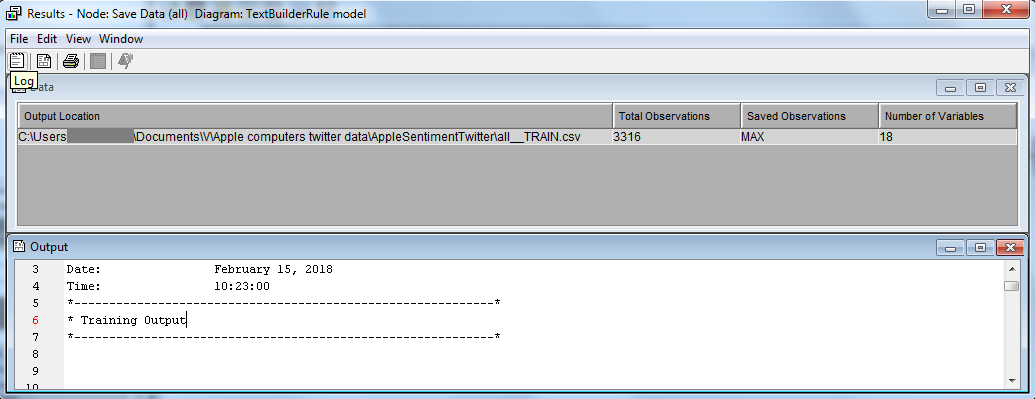
From these rules generated by the Text-Rule-Builder node, we can see that:

* negative tweets should contain {shit, charger, fuck, hate, fix, stop, phone, fuck, hell, cord, suck, apple & ~aapl & ~hire & ~aapl & ~workshop & update, work, battery, fail, upgrade, wtf, yall, dear, life, ass, .. }
* neutral tweets should have {intraday, hire, December, diein, aapl, apple & ceo, future, radioshack, motley, wozniak, antitrust, apple}
* positive tweets should have {love, team, nice, awesome, lifetime, survive, service, pc, love, good, amp}

Each term goes with its corresponding measures such as precision, recall, F1 score, True Positive/total.

Now, let’s see the result of what our model predicts:

*Step 10:* Right-click on Save Data node, and then click on Results, you will have some information about this result file: location, name, number of variables, total observations.

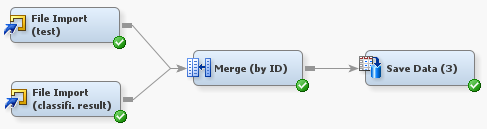
 

***Fig.13:*** *clicking to see Results of the model*

There are several ways to get the classification result for the testing set.

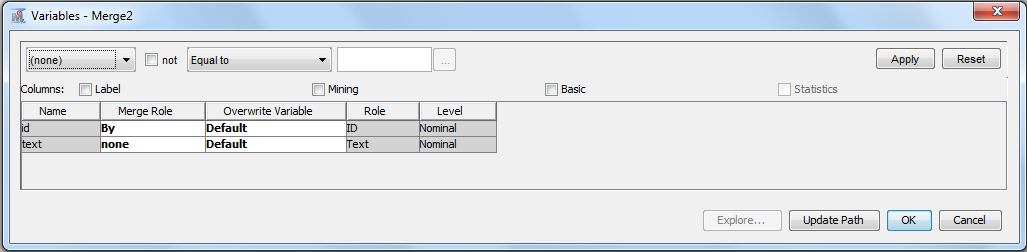
* *Step 11a* (1st way): we can add this diagram into our existing diagram to select the testing observations with their sentiment classification/prediction results.

You add 2 File Import nodes, one for the test file and the other for the file generated by the Save Data node of the previous diagram, named all\_TRAIN.csv. Next, add a Merge node and specify these two files are merged by ID, by right-clicking on the Merge node and choosing Edit variables, setting the id row’s parameters like the figure.



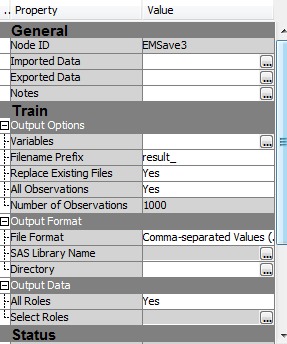
***Fig.13:*** *a diagram to merge the test file & the result file*

*to get the predicted sentiments for ids of the test set*



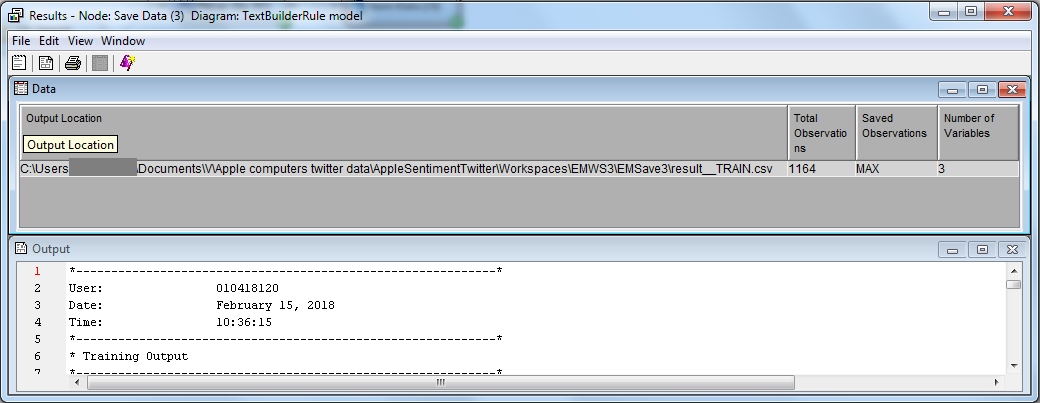
***Fig.14:*** *setting to merge the test file & the result file by ID*

You should have the Save Data node’s parameters like this:



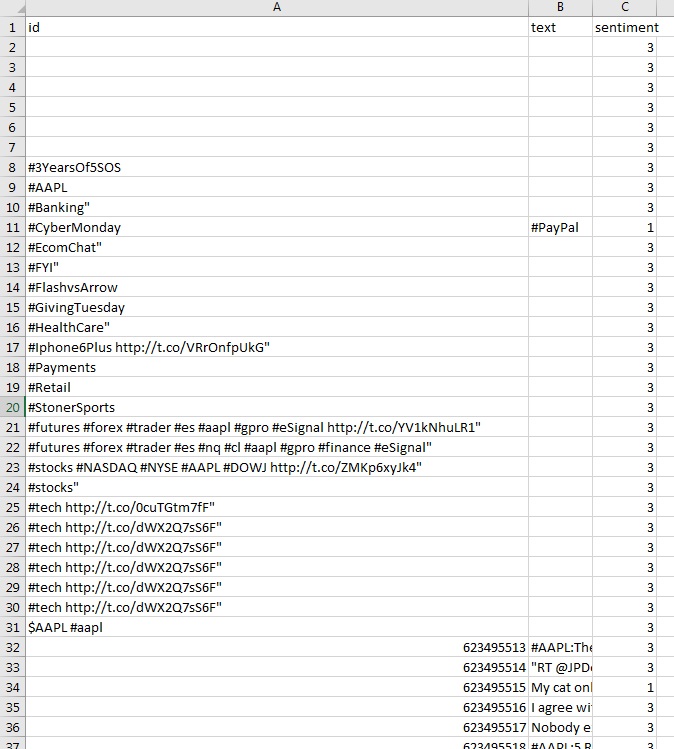
***Fig.15:*** *setting to save the merged file*

Again, clicking on Results of the Save Data (3) node will let you know where the data are saved.



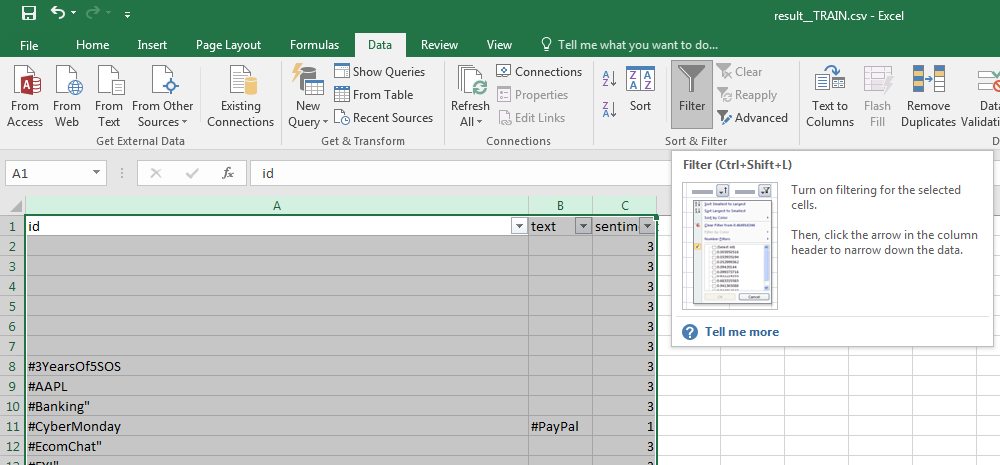
***Fig.16:*** *right-clicking on Save Data>Results to see some properties of the result file*

Why are there 1164 observations for a test set of 1000 observations? SAS doesn’t work perfectly with Text Parsing. Let’s look at the file:

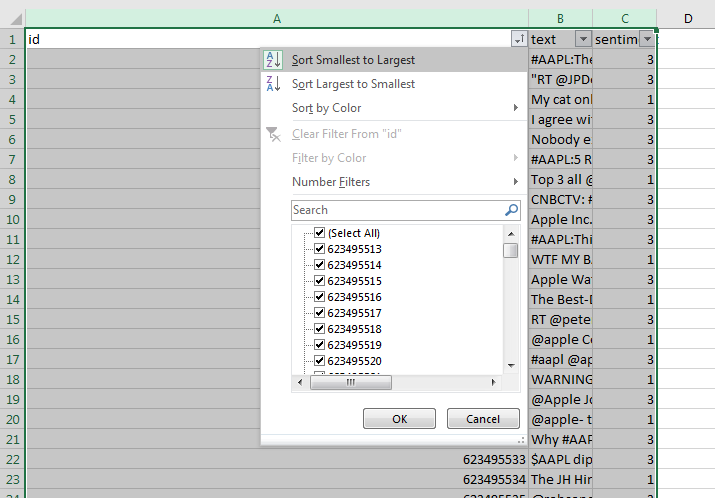


***Fig.17:*** *the file of sentiment prediction by the model*

You can see that there are some hashtags or links remained in the file. You can get rid of these by opening the file, selecting the three columns, and using Filter on the Data tab of the excel. Click on the id column to sort it from the smallest to the largest or the other way. You should select the 1000 rows of this file (rows with id starting with 6234), ignore the text column and save it as another .csv file (you can choose any name you like) and submit it to kaggle.com (the submission file should have only id & sentiment columns).

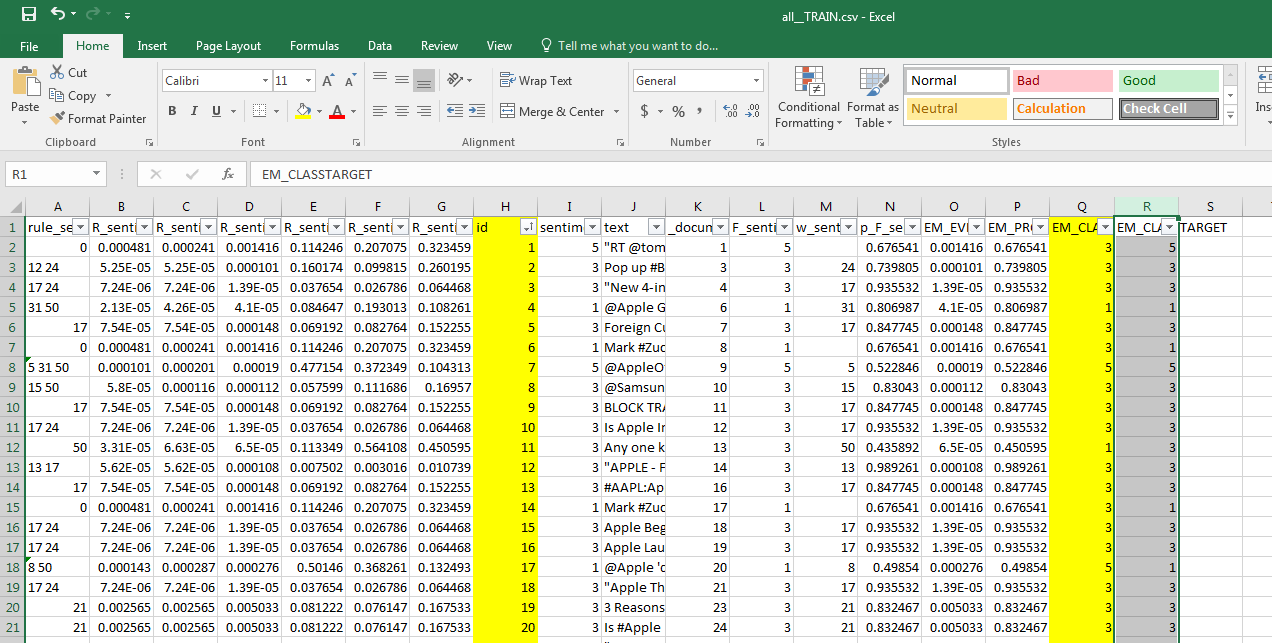


***Fig.18.a:*** *filter the merged data to get the wanted rows & columns*



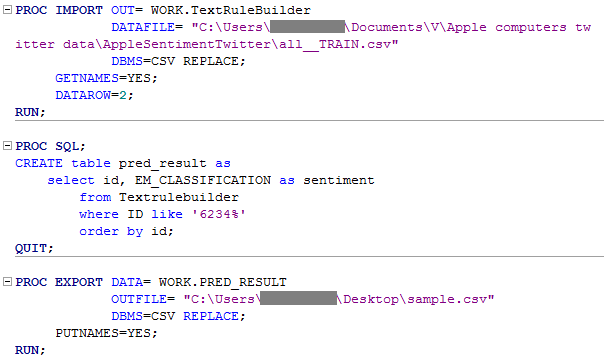
***Fig.18.b:*** *filter the merged data to get the wanted rows & columns: sort by ID*

* *Step 11b* (2nd way): select column id and EM\_CLASSIFICATION from the all\_train.csv file, then select rows with id starting with 6234 (you can do the same way above).



***Fig.19:*** *another way to get the sample result: filter the classification result of the model*

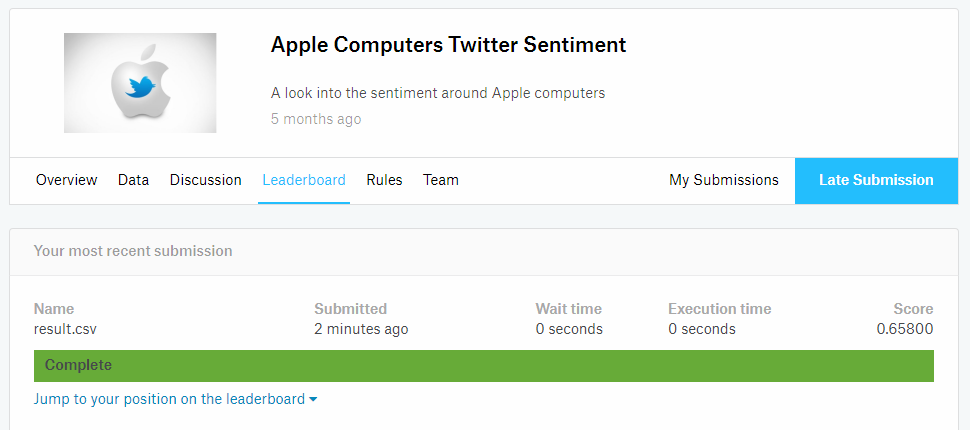
* *Step 11c* (3rd way): *If you know some SAS programming*: import the result file (by Wizard or by PROC IMPORT), then use PROC SQL to select what you need, and export the table to sample.csv file (by Wizard or by PROC EXPORT).



***Fig.20:*** *SAS code to get the results (id & sentiment) for the test set instead of using the Merge diagram*

This snippet of codes returns exactly what we want.

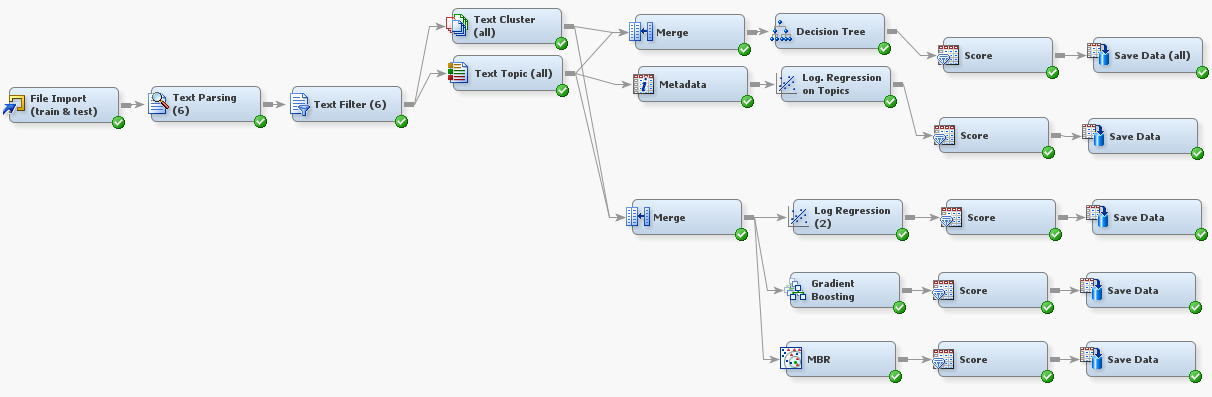
**Submission result from Kaggle:**



***Fig.21:*** *submission result from kaggle for the Text Rule Builder model*

**C. OTHER MODELS on TOPIC & TEXT CLUSTERS:**

After the first model – Text Rule Builder - you may become familiar with kinds of nodes for the diagram, now create another diagram named ‘Other models’. Make the diagram as below.



***Fig.22:*** *another diagram for other models*

After File Import and Text Parsing nodes, feed Text Filter node to Text Cluster & Text Topic nodes. We will have 4 models based on Text Cluster and Text Topics: Decision Tree, Logistic Regression, Gradient Boosting, and MBR (Memory-Based Reasoning). We will also have another model, which is Logistic regression, just based on Text Topics. The nodes: Merge is from Sample tab, Metadata & Save Data from Utility tab, Score from Access tab, model nodes (Decision Tree, Logistic Regression, Gradient Boosting, and MBR) from Model tab. For Logistic Regression node, the node used from Model tab is Regression, in its property window set Regression Type to Logistic Regression and Link function is Logit. For other nodes, leave them with the SAS default setting. For Save Data nodes, in their corresponding property windows, set prefix differently so that you can distinguish one result file with another. For example, for model with MBR, I set the prefix as mbr\_. In order to get the information for submission to kaggle.com, do it in one of three ways described previously in the Text Rule Builder model.



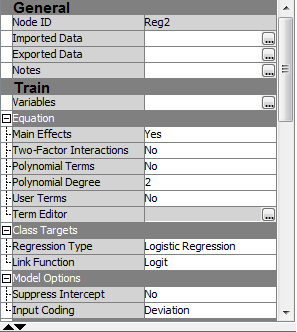
***Fig.23:******Model*** *tab which includes node icon for various models.*



***Fig.24:******Access*** *tab which includes node icon for accessing models*



***Fig.25:******Utility*** *tab which includes node icon for model utilities*



***Fig.26:*** *Property window of the (Logistic) Regression node*

The results from kaggle responding to the models MBR, Gradient Boosting, Logistic regression, Logistic Regression on Text Topics, Decision Tree are below:



***Fig.27:*** *submission result from kaggle for the models: MBR, Gradient Boosting, Logistic Regression on Text Topics & Text Clusters, Logistic Regression on Text Topics only, Decision Tree*

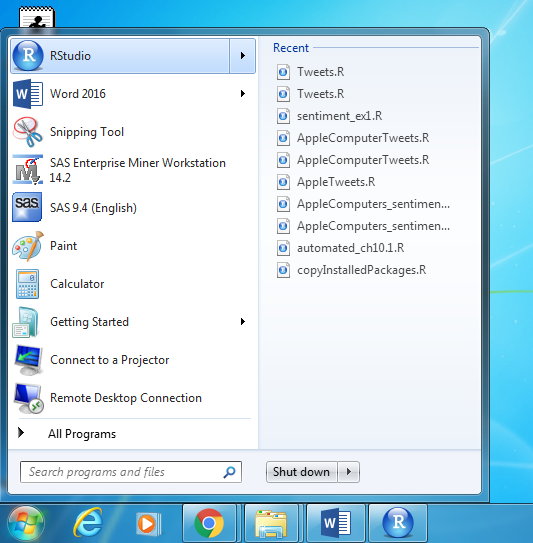
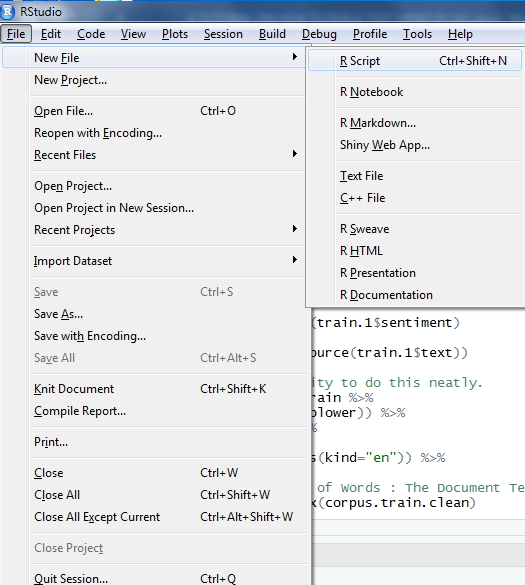
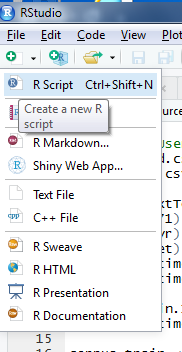
Kaggle provides 2 scores (private score, 50%, & public score, 50%), one based on the test set itself and the other based on other data. So far, Memory Based Reasoning model got the highest private score (**68.4%**) & Gradient Boosting got the highest public score (**67.4%**). If you compare two models with logistic regression, you will find that the model on text topics and text clusters render a better result as it has more information for the sentiment prediction than the other. You can try to improve these scores with other models once you get familiar with SAS Enterprise Miner. These are just some approaches among many.

**R Programming tutorial for**

**Apple Computer Tweets Sentiment Analysis**

This tutorial provides a guide on how to conduct a sentiment analysis using R programming. The analysis is conducted as following: clean data (Apple computer tweets), get the document-term matrix, then apply some state-of-the-art machine learning algorithms on the document term matrix. The following algorithms are used: multinomial logistic regression, elastic net, & random forest. This R programing part deals with models built on the Document Term Matrix.

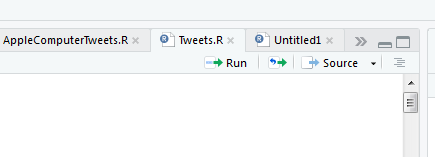
*Technical Setting:* this analysis was done in R Studio 1.1.383, R version is R-3.4.2

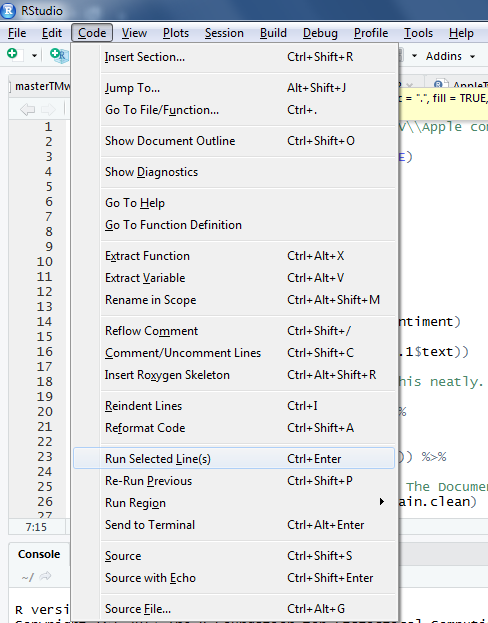
  

***Fig. 1:*** *how to create a R script file*

From Windows icon, open RStudio and then click on File > New File > R Script or R Script icon below the File tab to open a new RScript file. You can also open an Editor in R to create a script.

In order to run some lines of codes, select them and hit Ctrl + Enter or click on Run icon of R Studio, or click on menu Code > Run Selected Line(s)

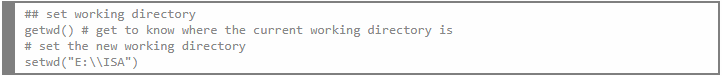




***Fig. 2.a & 2.b:*** *how to run a R script file:* ***(a)*** *click the Run icon;*

***(b)*** *click* ***Run Selected Line(s)*** *from menu*

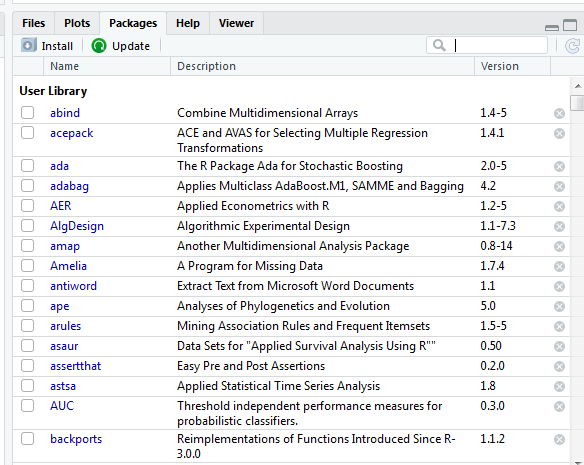
*Set working directory:*



***Fig. 3:*** *setting working directory*

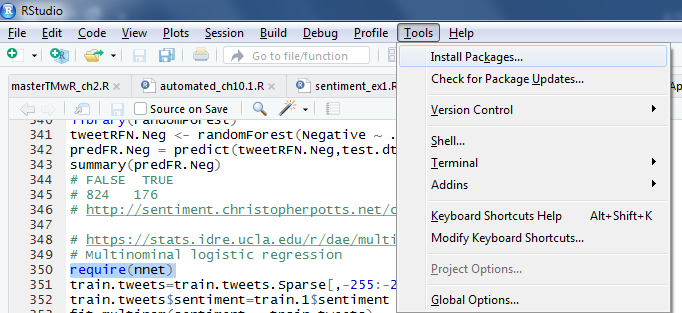
*Load necessary libraries:*

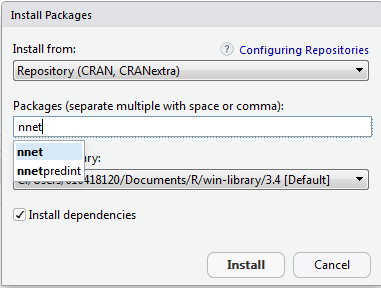
First you need to check if your needed libraries are already installed yet. You can do that by looking at the window located at the bottom right of the R Studio. You can type a specific package name into the search box of that window to see if it is there.



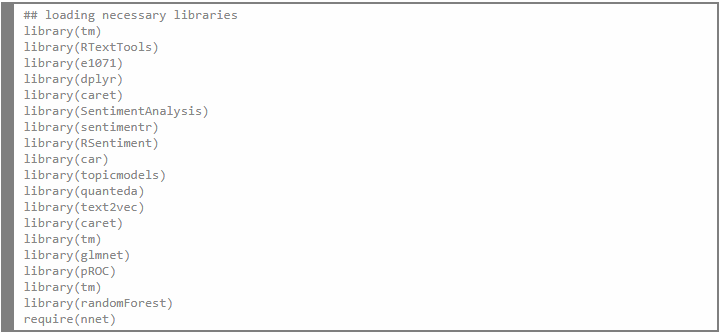
***Fig. 4:*** *check if a library is installed on R yet*

If some package is not installed yet, you can install it by clicking on the menu Tools > Install Packages. A small window will appear, choose the configuring repositories (mostly from CRAN) & type in the package’s name, then click Install and it will be done.





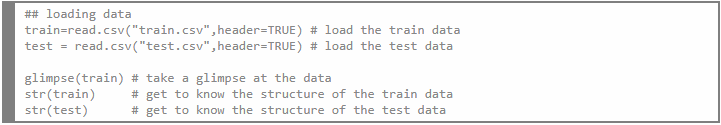
***Fig. 5:*** *how to install a package into R*



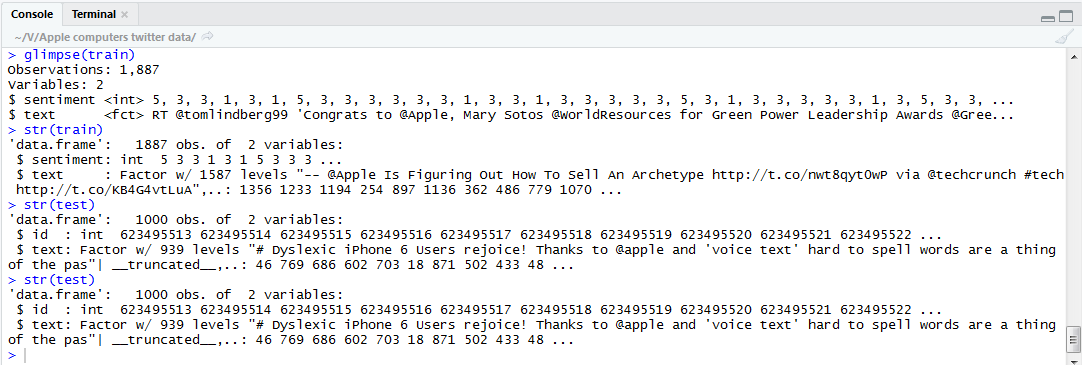
***Fig. 6:*** *codes how to load R libraries*

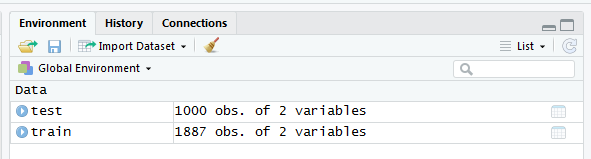
‘Require’ or ‘library’ commands are used to load the necessary packages.

*Load data (training & testing data sets):*



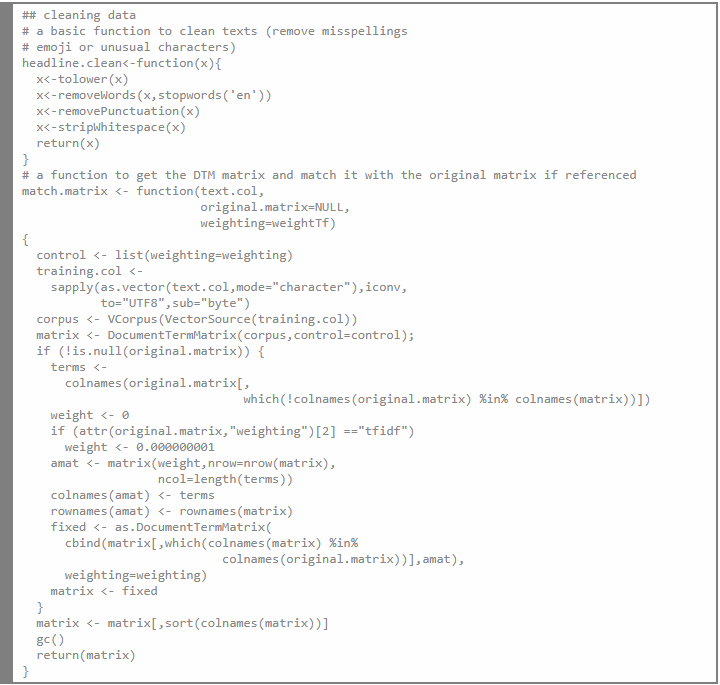
***Fig. 6:*** *codes how to load data*





***Fig. 7:*** *R console & R environment windows show information about data*

*Clean data:*





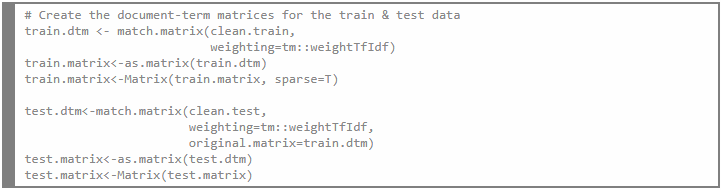
***Fig. 8.a & b:*** *codes of* ***(a)*** *the function* ***headline*** *(cleaning text) &* ***match\_matrix*** *(creating DocumentTermMatrix & forcing two matrices to have the same number of columns);* ***(b)*** *clean texts of the train & test data*

The *headline* function is to clean the text data: turn all words to lower-case, remove stopwords according to English dictionary, remove punctuations, and strip whitespaces.

The *match\_matrix* function is to create a DocumentTermMatrix from text, and to match the DocumentTermMatrix of the test set to that of the train set so that they will have the same number of columns.

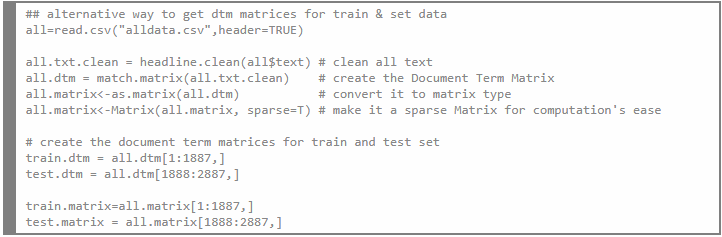
*Create the document-term matrices:*

The train set and test set are cleaned, then the corresponding document-term matrices are computed and forced to have the same dimension.



***Fig. 9a:*** *codes how to create the Document Term Matrices from the train & test sets*

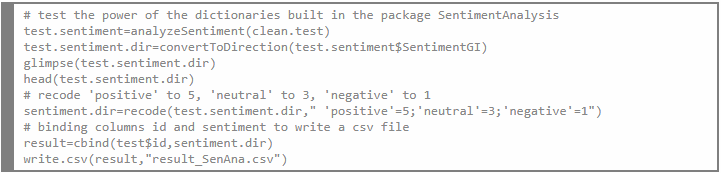
There is another easier way to create the Document Term Matrices for the train & test sets. Remember the alldata.csv file we mentioned in page 3 of SAS tutorial section. This is the file we gather all data from the train set and the test set. We will just take the text column, clean it, make the document term matrix from it, convert it to R matrix data type, make it a Sparse Matrix for the computation’s ease, then separate what is from the train set to that from the test set.



***Fig. 9b:*** *codes of another way to create the Document Term Matrices from the train & test sets*

*Apply a model & predict:*

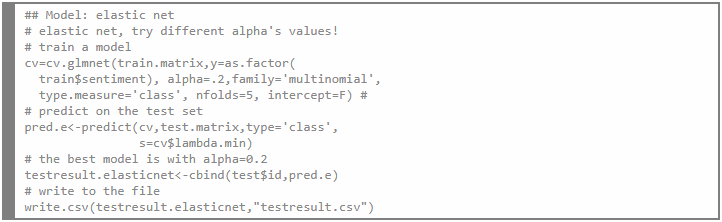
1. Test the power of 3 dictionaries build on R package *SentimentAnalysis*:



***Fig. 10:*** *codes how to use the* ***SentimentAnalysis*** *package’s dictionaries to predict sentiments*

The *SentimentAnalysis* package has several built-in dictionaries. This step is just a trial to see how powerful these dictionaries on predicting the sentiments of the test data.

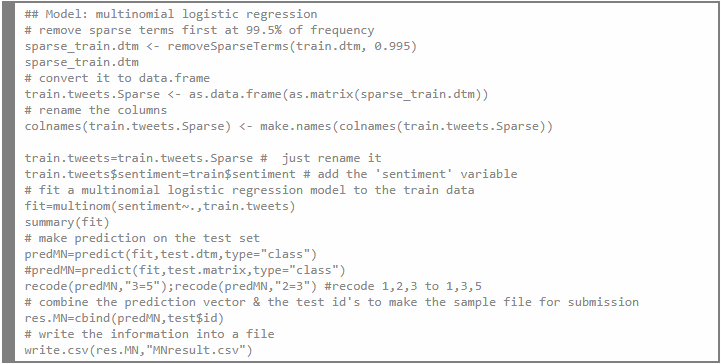
**2. Elastic Net:**



***Fig. 11:*** *codes how to build the* ***elastic net*** *model*

Based on the Document Term Matrix of the train set, elastic net model is trained with multinomial family and 5 folds of cross validation. Different alpha values are tried to get the best model. The optimal alpha is 0.2. Then the fitting object is used to predict the sentiments of the test set. The sentiment & id of the test set are combined into a file for kaggle.com submission.

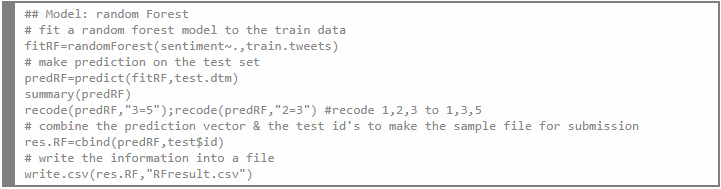
**3. Multinomial Logistic Regression:**



***Fig. 12:*** *codes how to build the* ***multinomial logistic regression*** *model*

The Sparse terms are removed from the train set at 99.5% of frequency, then the Document Term Matrix of the train set is converted to R data.frame type. As the sentiment column has three values, the multinomial logistic regression is applied on the Document Term Matrix. Then the fitting object is used to predict the sentiments of the test set. The results are recoded to match with the required sentiment codes (1,3,5 as negative, neutral, positive). The sentiment & id of the test set are combined into a file for kaggle.com submission.

**4. Random Forest:**



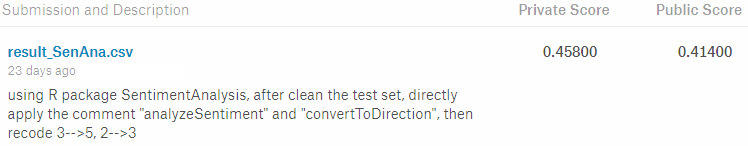
***Fig. 13:*** *codes how to build the* ***random forest*** *model*

The Random Forest model is applied on the Document Term Matrix, the response variable is *sentiment*. Using this fitting model, prediction is made on the test set to acquire the predicted sentiment. The results are recoded to match with the required sentiment codes (1,3,5 as negative, neutral, positive). The sentiment & id of the test set are combined into a file for kaggle.com submission.

*Submit the results to kaggle.com:*

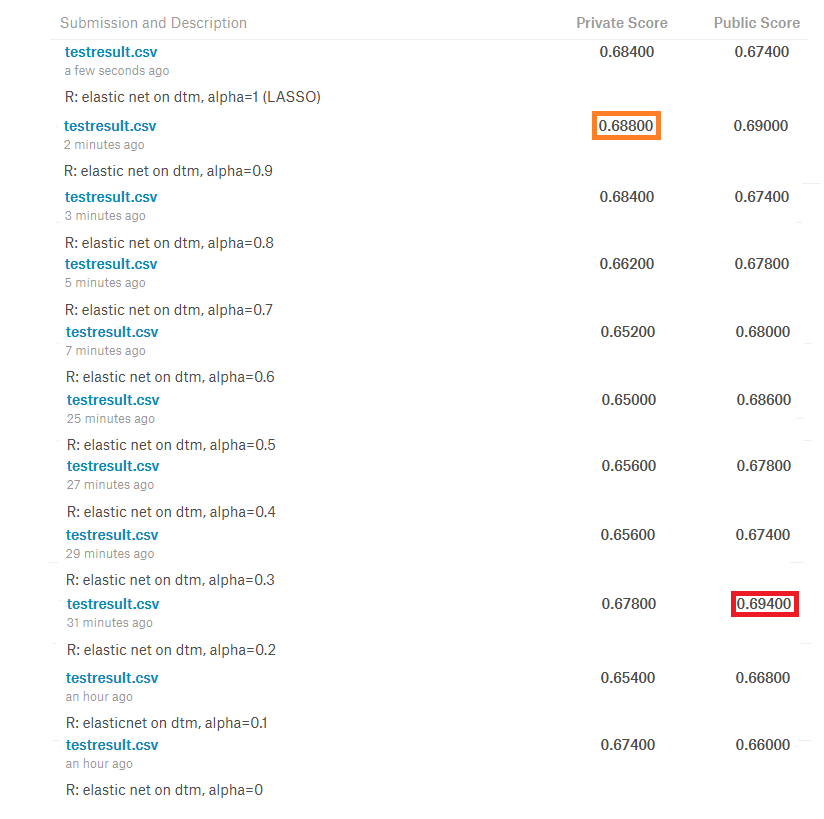
1. **Testing the power of 3 dictionaries built on R package SentimentAnalysis on the test data:**

file [result\_SenAna.csv](https://www.kaggle.com/submissions/6731911/6731911.zip) got scores quite low: private score of 45.8% & public score got 41.4%.



***Fig. 14:*** *result from kaggle.com for this trial*

2. **Elastic Net on the Document-Term Matrix:**



***Fig. 15:*** *result from kaggle.com for the elastic net model*

Elastic net is trained on the training set with different alpha, from alpha=0 (Ridge) to alpha=1 (LASSO). The highest public score **0.694** goes with **alpha=0.2**, The highest private score **0.688** goes with **alpha= 0.9**

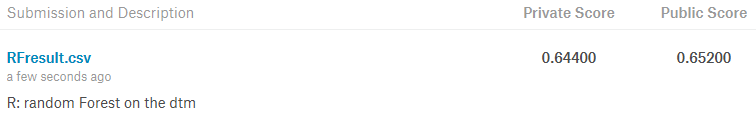
3. **Multinomial Logistic Regression on the Document-Term matrix:**

The train set’s document term matrix is removed rare words by 99.5%. After that, fitting a multinomial regression on this document term matrix with the response is the sentiment variable of the train set. Using this model to predict the sentiment of the test set (the document-term matrix of the test set), I got:



***Fig. 16:*** *result from kaggle.com for the multinomial logistic regression model*

4. **Random Forest on the document-term matrix**:



***Fig. 17:*** *result from kaggle.com for the random forest model*

So far, Elastic net (alpha=0.2 or 0.9) renders a best result on the test set.