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** (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:

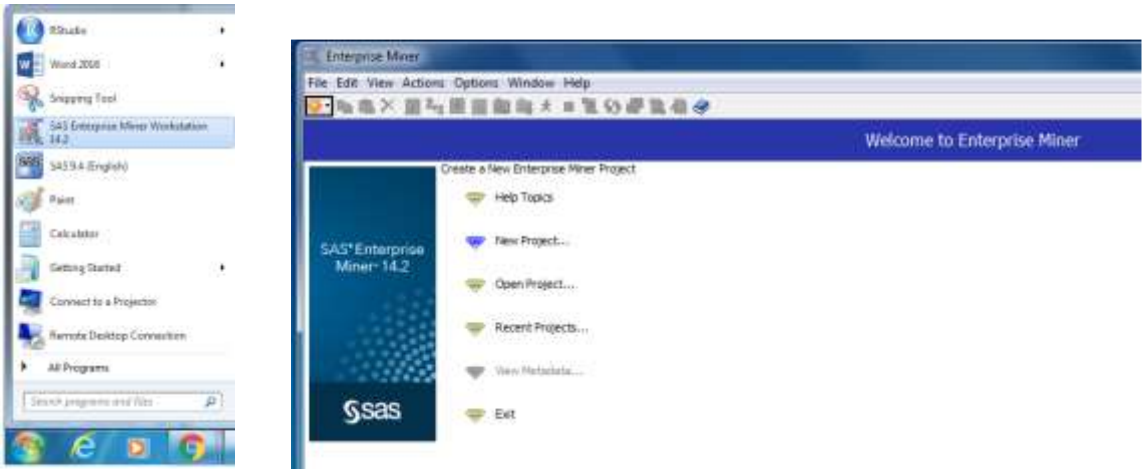
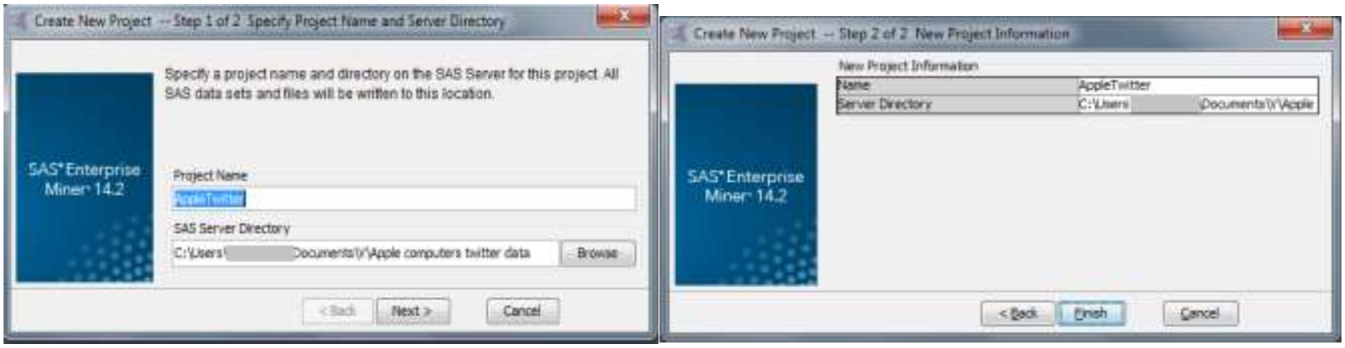

RStudio
Word 2016
Snipping Tool
SAS Enterprise Miner Workstation 14.2
SAS 9.4 (English)
Paint
Calculator
Getting Started
Connect to a Projector
Remote Desktop Connection
All Programs
Search programs and files
Enterprise Miner
File Edit View Actions Options Window Help
Welcome to Enterprise Miner
Create a New Enterprise Miner Project
Help Topics
New Project...
Open Project...
Recent Projects...
View Metadata...
Exit
SAS® Enterprise Miner™ 14.2
sas

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

<u>*Step 1:*</u> 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.


Create New Project -- Step 1 of 2 Specify Project Name and Server Directory

Specify a project name and directory on the SAS Server for this project. All SAS data sets and files will be written to this location.

Project Name: AppleTwitter

SAS Server Directory: C:\Users\[redacted]Documents\\"\Apple computers twitter data

Create New Project -- Step 2 of 2 New Project Information

New Project Information

Name	AppleTwitter
Server Directory	C:\Users[redacted]Documents\\"\Apple

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

<u>*Step 2:*</u> 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.

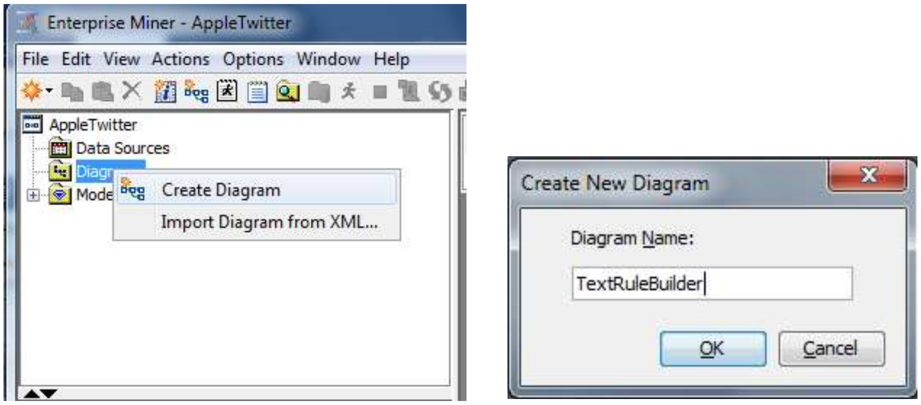


Diagram Name: TextRuleBuilder

Enterprise Miner - AppleTwitter
File Edit View Actions Options Window Help
AppleTwitter
Data Sources
Diagr
Mode
Create Diagram
Import Diagram from XML...
Create New Diagram
OK
Cancel

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

Create the diagram:

The final diagram should be like this:

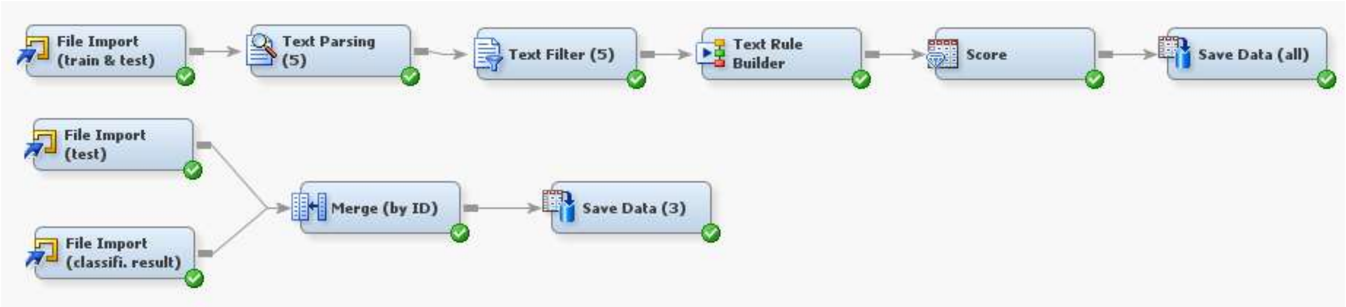
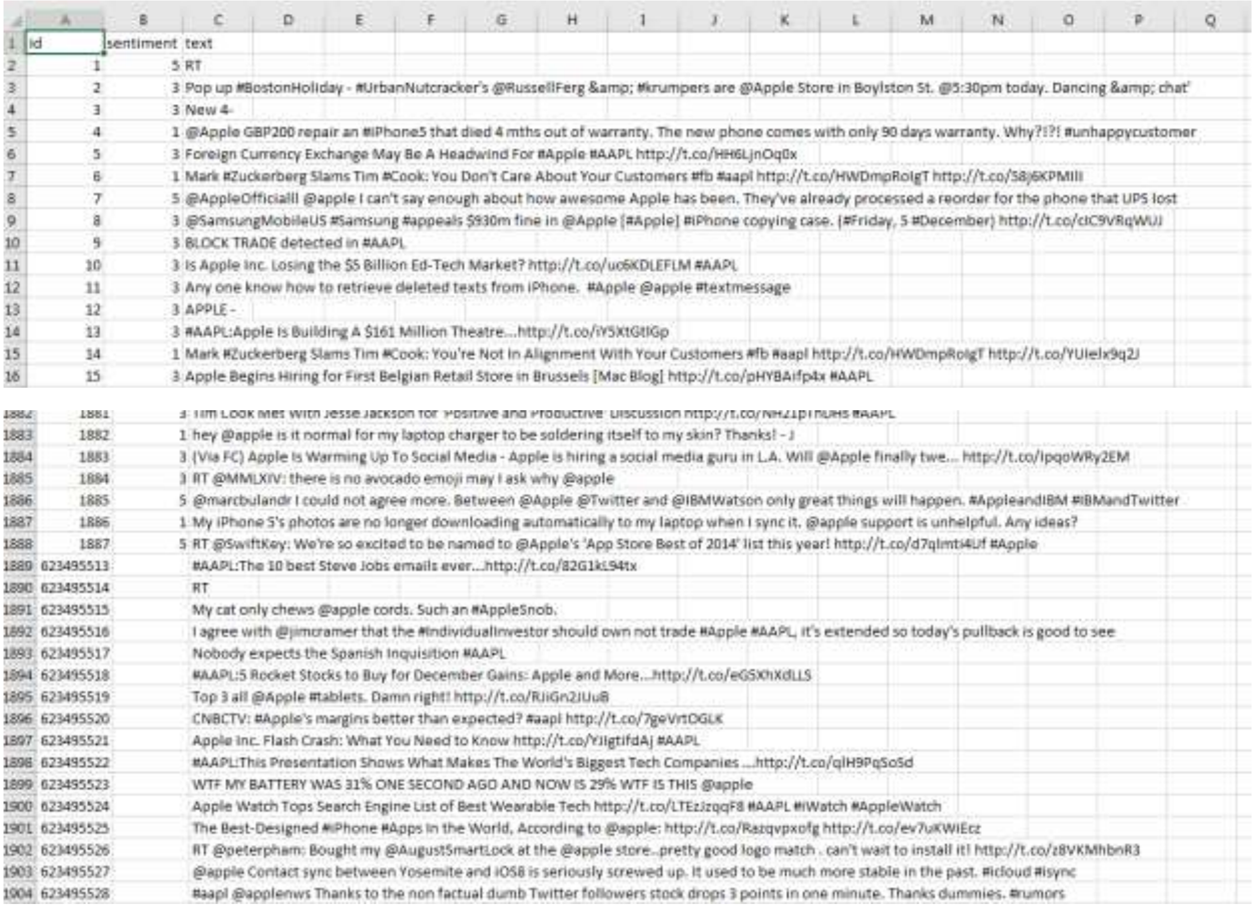


***Fig. 3:** diagram of the Text Rule Builder model*

<u>*Step 3: Data preparation:*</u> 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.



	A	B	C
1	id	sentiment	text
2	1	5	RT
3	2	3	Pop up #BostonHoliday - #UrbanNutcracker's @RussellFerg &amp; #krumpers are @Apple Store in Boylston St. @5:30pm today. Dancing &amp; chat'
4	3	3	New 4-
5	4	1	@Apple GBP200 repair an #iPhone5 that died 4 mths out of warranty. The new phone comes with only 90 days warranty. Why?!?! #unhappycustomer
6	5	3	Foreign Currency Exchange May Be A Headwind For #Apple #AAPL http://t.co/HH6LjnOq0x
7	6	1	Mark #Zuckerberg Slams Tim #Cook: You Don't Care About Your Customers #fb #aapl http://t.co/HWDmpRolgT http://t.co/58j6KPMIll
8	7	5	@AppleOfficialll @apple I can't say enough about how awesome Apple has been. They've already processed a reorder for the phone that UPS lost
9	8	3	@SamsungMobileUS #Samsung #appeals $930m fine in @Apple (#Apple) #iPhone copying case. (#Friday, 5 #December) http://t.co/cIC9VRqWUJ
10	9	3	BLOCK TRADE detected in #AAPL
11	10	3	Is Apple Inc. Losing the $5 Billion Ed-Tech Market? http://t.co/uc6KDLEFLM #AAPL
12	11	3	Any one know how to retrieve deleted texts from iPhone. #Apple @apple #textmessage
13	12	3	APPLE -
14	13	3	#AAPL:Apple Is Building A $161 Million Theatre...http://t.co/iY5XtGtIGp
15	14	1	Mark #Zuckerberg Slams Tim #Cook: You're Not In Alignment With Your Customers #fb #aapl http://t.co/HWDmpRolgT http://t.co/YUIelx9q2J
16	15	3	Apple Begins Hiring for First Belgian Retail Store in Brussels [Mac Blog] http://t.co/pHYBAifp4x #AAPL
1882	1881	3	Tim Cook Met With Jesse Jackson for 'Positive and Productive' Discussion http://t.co/NH21pTnUHs #AAPL
1883	1882	1	hey @apple is it normal for my laptop charger to be soldering itself to my skin? Thanks! - J
1884	1883	3	(Via FC) Apple Is Warming Up To Social Media - Apple is hiring a social media guru in L.A. Will @Apple finally twe... http://t.co/IpqoWRy2EM
1885	1884	3	RT @MMLXIV: there is no avocado emoji may I ask why @apple
1886	1885	5	@marcbulandr I could not agree more. Between @Apple @Twitter and @IBMWatson only great things will happen. #AppleandIBM #IBMandTwitter
1887	1886	1	My iPhone 5's photos are no longer downloading automatically to my laptop when I sync it. @apple support is unhelpful. Any ideas?
1888	1887	5	RT @SwiftKey: We're so excited to be named to @Apple's 'App Store Best of 2014' list this year! http://t.co/d7qlmti4Uf #Apple
1889	623495513		#AAPL:The 10 best Steve Jobs emails ever...http://t.co/82G1kL94tx
1890	623495514		RT
1891	623495515		My cat only chews @apple cords. Such an #AppleSnob.
1892	623495516		I agree with @jimcramer that the #IndividualInvestor should own not trade #Apple #AAPL, it's extended so today's pullback is good to see
1893	623495517		Nobody expects the Spanish Inquisition #AAPL
1894	623495518		#AAPL:5 Rocket Stocks to Buy for December Gains: Apple and More...http://t.co/eG5XhXdLLS
1895	623495519		Top 3 all @Apple #tablets. Damn right! http://t.co/RJiGn2JUuB
1896	623495520		CNBCTV: #Apple's margins better than expected? #aapl http://t.co/7geVrtOGLK
1897	623495521		Apple Inc. Flash Crash: What You Need to Know http://t.co/YJIgtifdAj #AAPL
1898	623495522		#AAPL:This Presentation Shows What Makes The World's Biggest Tech Companies ...http://t.co/qIH9PqSo5d
1899	623495523		WTF MY BATTERY WAS 31% ONE SECOND AGO AND NOW IS 29% WTF IS THIS @apple
1900	623495524		Apple Watch Tops Search Engine List of Best Wearable Tech http://t.co/LTEzJzqqF8 #AAPL #iWatch #AppleWatch
1901	623495525		The Best-Designed #iPhone #Apps In the World, According to @apple: http://t.co/Razqvpxofg http://t.co/ev7uKWIEcz
1902	623495526		RT @peterpham: Bought my @AugustSmartLock at the @apple store..pretty good logo match . can't wait to install it! http://t.co/z8VKMhbnR3
1903	623495527		@apple Contact sync between Yosemite and iOS8 is seriously screwed up. It used to be much more stable in the past. #icloud #isync
1904	623495528		#aapl @applenws Thanks to the non factual dumb Twitter followers stock drops 3 points in one minute. Thanks dummies. #rumors

Fig. 4: *data preparation step*

<u>*Step 4:*</u> 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.

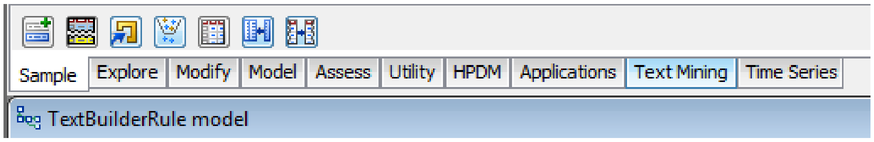
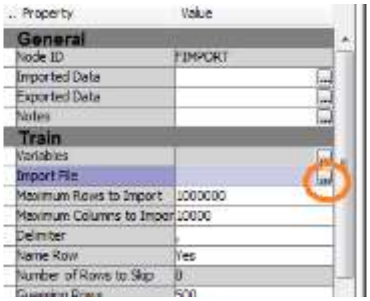
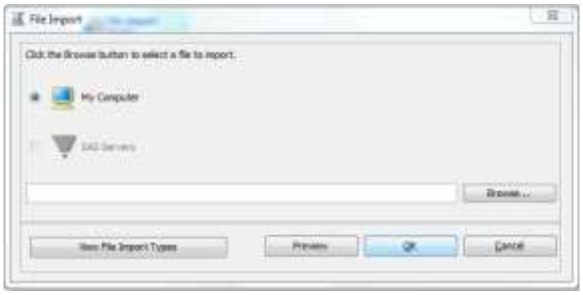


Fig.5: Sample *tab which includes node icons for creating a diagram*



.. Property	Value
General	
Node ID	FIMPORT
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Import File	
Maximum Rows to Import	1000000
Maximum Columns to Impor	10000
Delimiter	,
Name Row	Yes
Number of Rows to Skip	0
Guessing Rows	500



File Import

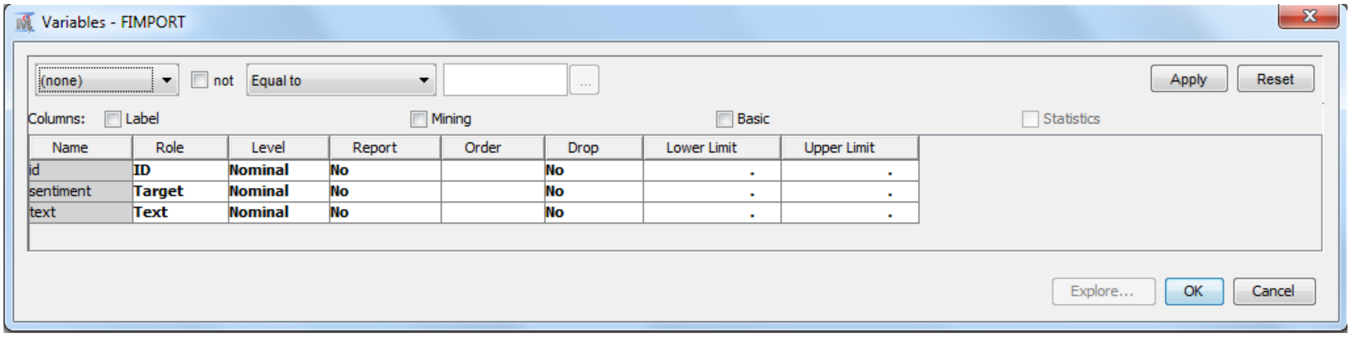
Click the Browse button to select a file to import.

● My Computer

○ SAS Servers

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.



Variables - FIMPORT

(none) ☐ not Equal to

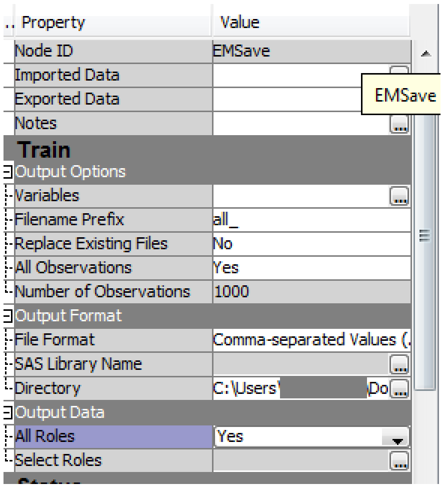
Columns: ☐ Label ☐ Mining ☐ Basic ☐ Statistics

Name	Role	Level	Report	Order	Drop	Lower Limit	Upper Limit
id	**ID**	**Nominal**	**No**		**No**	.	.
sentiment	**Target**	**Nominal**	**No**		**No**	.	.
text	**Text**	**Nominal**	**No**		**No**	.	.

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.

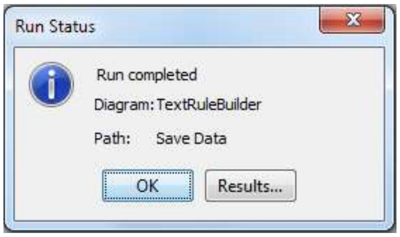


.. Property	Value
Node ID	EMSave
Imported Data	
Exported Data	
Notes	
Train	
Output Options	
Variables	
Filename Prefix	all_
Replace Existing Files	No
All Observations	Yes
Number of Observations	1000
Output Format	
File Format	Comma-separated Values (.
SAS Library Name	
Directory	C:\Users\[redacted]Do
Output Data	
All Roles	Yes
Select Roles	

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’.



Run Status

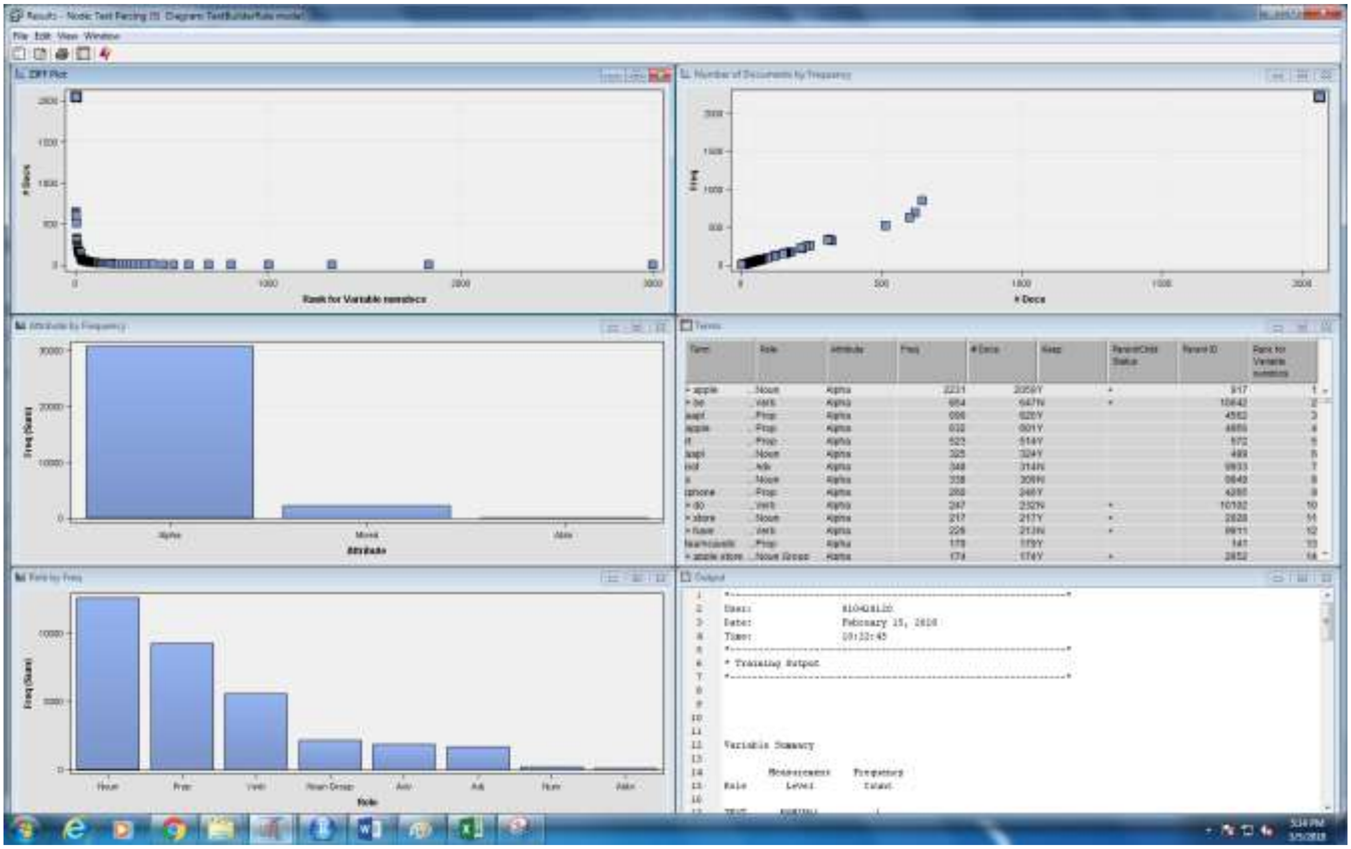
Run completed

Diagram: TextRuleBuilder

Path: Save Data

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

Results of Text Parsing node:



Term	Role	Attribute	Freq	# Docs	Keep	Parent/Child Status	Parent ID	Rank for Variable numdocs
+ apple	Noun	Alpha	2231	2059	Y	+	917	1
+ be	Verb	Alpha	654	647	N	+	10642	2
aapl	Prop	Alpha	656	620	Y		4582	3
apple	Prop	Alpha	632	601	Y		4655	4
rt	Prop	Alpha	523	514	Y		572	5
aapl	Noun	Alpha	325	324	Y		489	6
not	Adv	Alpha	348	314	N		9933	7
s	Noun	Alpha	338	309	N		0640	8
iphone	Prop	Alpha	265	246	Y		4285	9
+ do	Verb	Alpha	247	232	N	+	10702	10
+ store	Noun	Alpha	217	217	Y	+	2828	11
+ have	Verb	Alpha	226	213	N	+	9911	12
[illegible]	Prop	Alpha	179	179	Y		141	13
+ apple store	Noun Group	Alpha	174	174	Y	+	2852	14

```
*------------------------------------------------------------*
User:                 810428120
Date:                 February 15, 2018
Time:                 10:22:45
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*



Variable Summary

         Measurement    Frequency
Role        Level         Count

TEXT       NOMINAL          1
```

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).

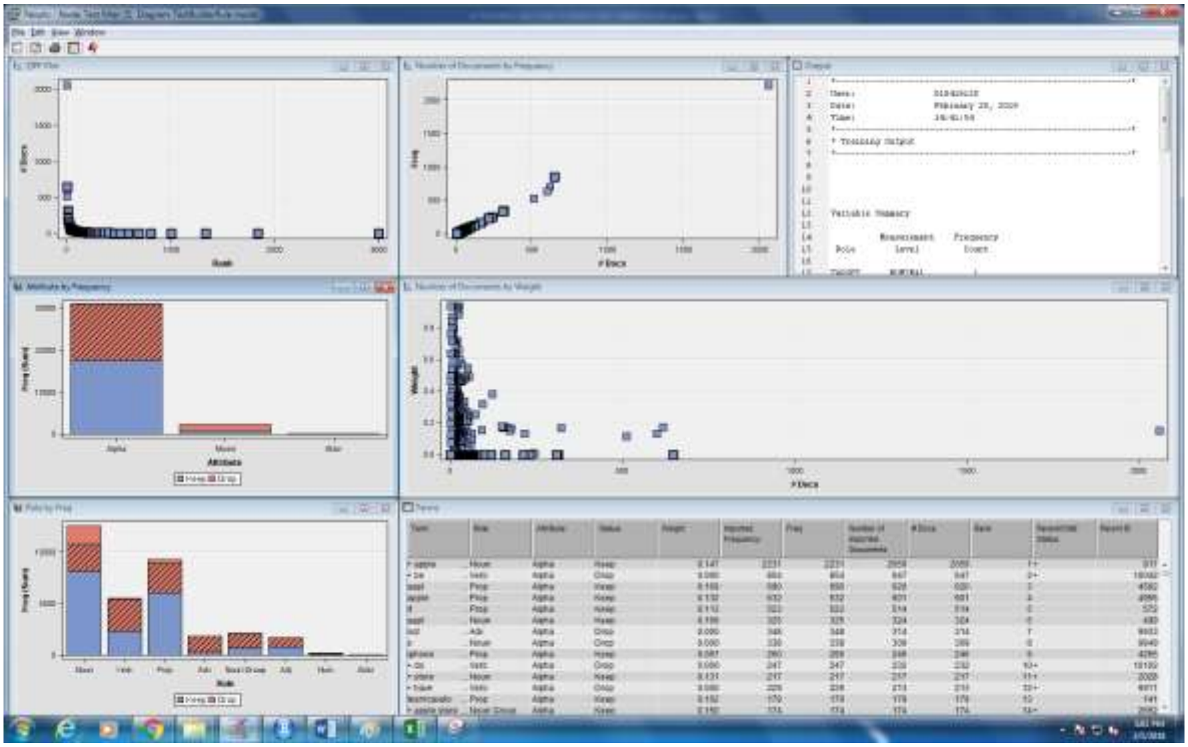


Fig.11: *Results of Text Filter node*

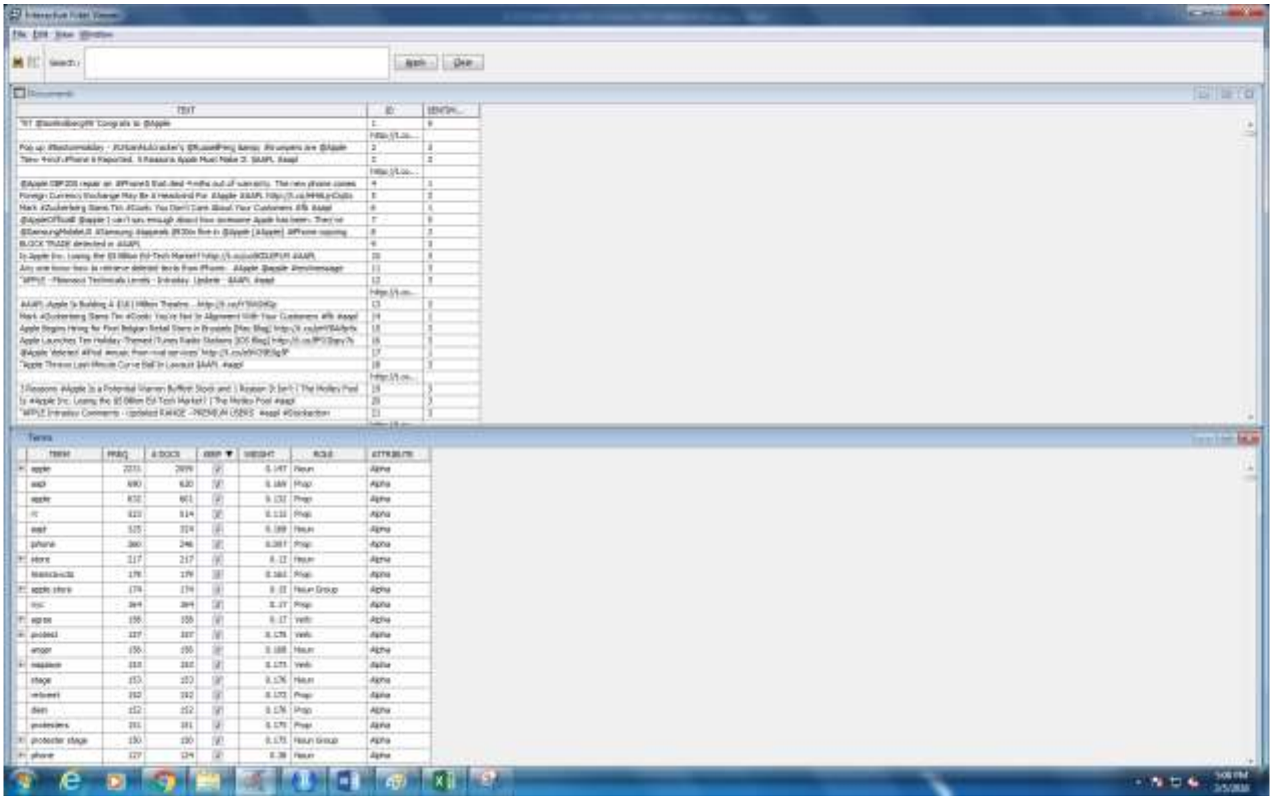


Fig. 12: *clicking the Interactive Filter Viewer of the Property windows of the Text Filter node to see which words are kept or dropped*

<u>What does the Text Rule Builder generate?</u>

<u>Step 9:</u> 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:

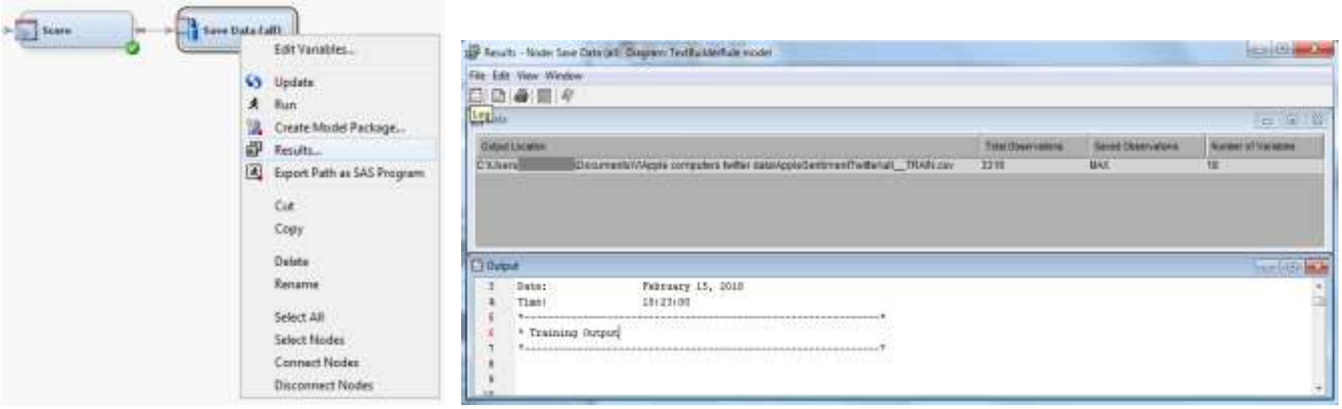
- negative tweets should contain {shit, charger, fuck, hate, fix, stop, phone, fuck, hell, cord, suck, apple & ~aapl & ~hire & ~aapl & ~workshop & update, work, battery, fail, upgrade, wtf, yall, dear, life, ass, .. }

- 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.

<u>Now, let's see the result of what our model predicts</u>:

<u>*Step 10:*</u> 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.



Output Location	Total Observations	Saved Observations	Number of Variables
C:\Users\[redacted]\Documents\\Apple computers twitter data\AppleSentimentTweets\all__TRAIN.csv	2316	MAX	18

```
Date:                    February 15, 2018
Time:                    18:23:08
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*
```

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

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

- <u>*Step 11a* (1st way):</u> 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.

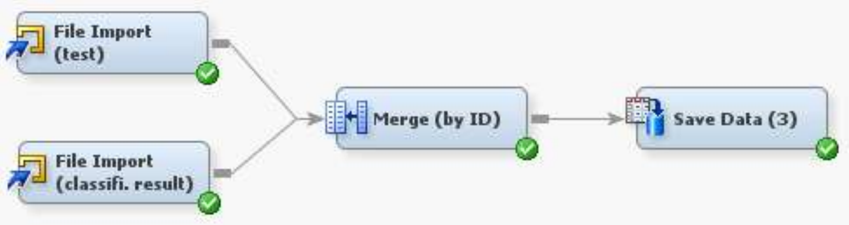
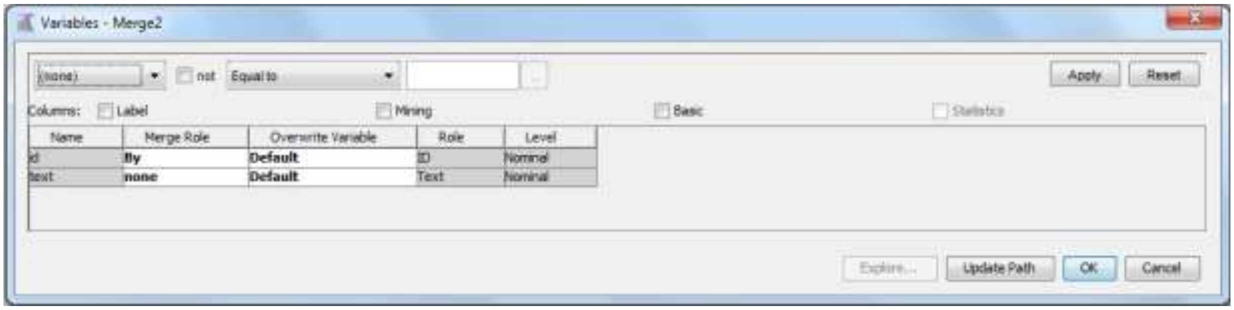


Fig.13: *a diagram to merge the test file & the result file to get the predicted sentiments for ids of the test set*



Variables - Merge2

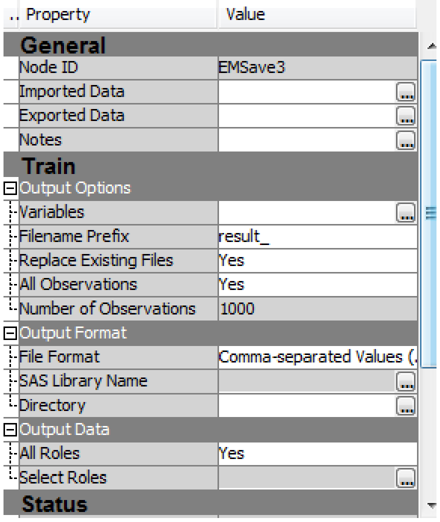
(none) □ not Equal to

Columns: □ Label □ Mining □ Basic □ Statistics

Name	Merge Role	Overwrite Variable	Role	Level
id	**By**	**Default**	ID	Nominal
text	**none**	**Default**	Text	Nominal

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

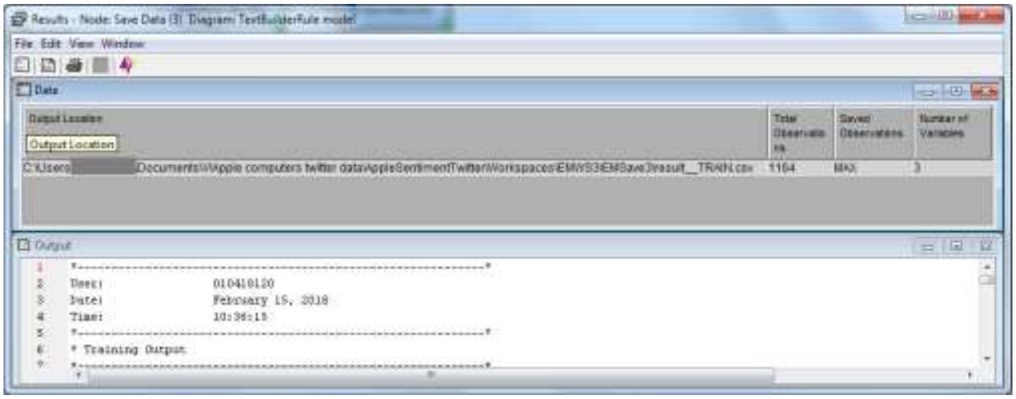
You should have the Save Data node's parameters like this:



Property	Value
General	
Node ID	EMSave3
Imported Data	...
Exported Data	...
Notes	...
Train	
⊟Output Options	
Variables	...
Filename Prefix	result_
Replace Existing Files	Yes
All Observations	Yes
Number of Observations	1000
⊟Output Format	
File Format	Comma-separated Values (.
SAS Library Name	...
Directory	...
⊟Output Data	
All Roles	Yes
Select Roles	...
Status	

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.



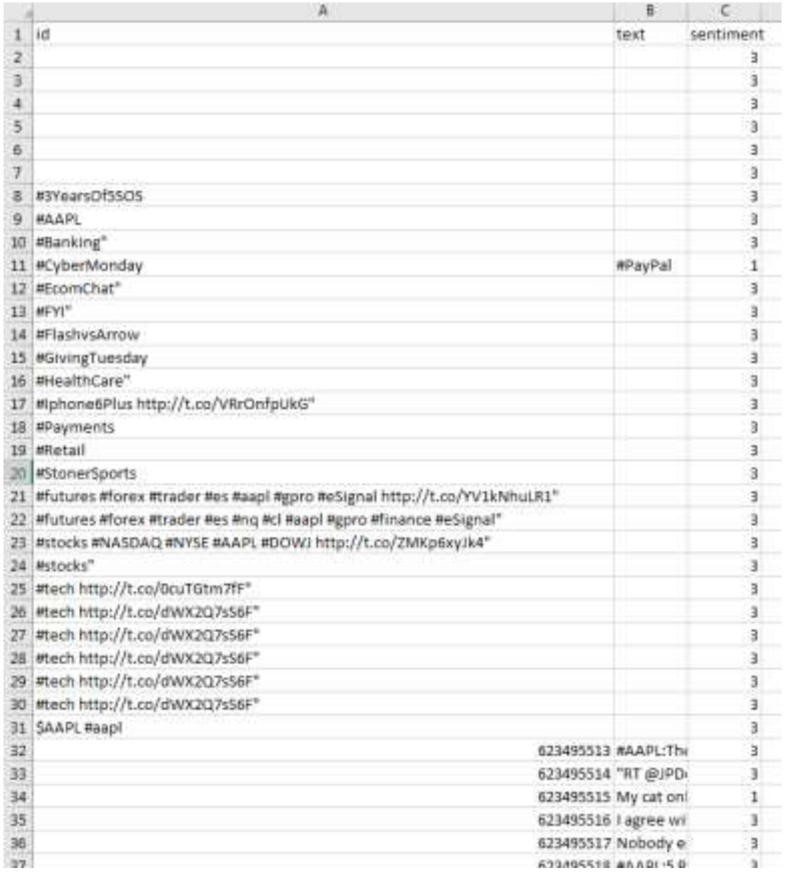
Results - Node: Save Data (3) Diagram: TextBuilderRule model

Output Location	Total Observations	Saved Observations	Number of Variables
C:\Users\[redacted]\Documents\\Apple computers twitter data\AppleSentimentTwitter\Workspaces\EMWS3\EMSave3\result__TRAIN.csv	1164	MAX	3

```
*------------------------------------------------------------*
User:                  010410120
Date:                  February 15, 2018
Time:                  10:36:15
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*
```

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

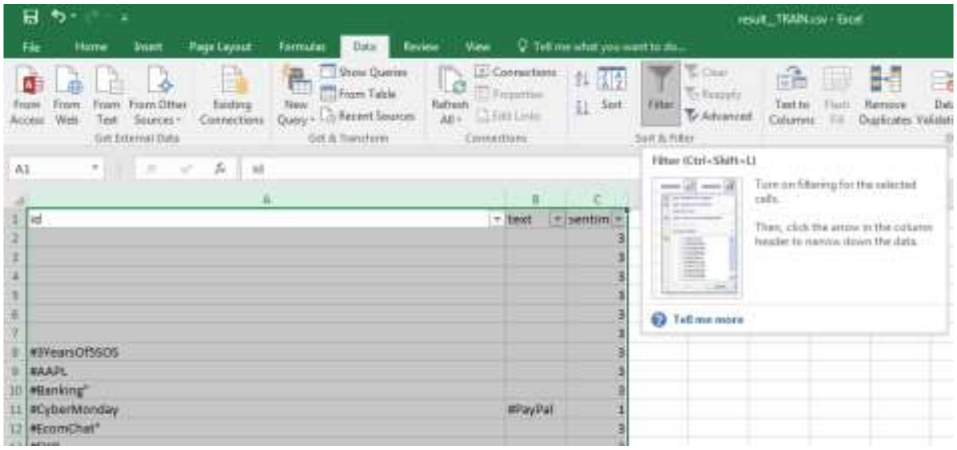
Why are there 1164 observations for a test set of 1000 observations? SAS doesn't work perfectly with Text Parsing. Let's look at the file:



	A	B	C
1	id	text	sentiment
2			3
3			3
4			3
5			3
6			3
7			3
8	#3YearsOf5SOS		3
9	#AAPL		3
10	#Banking"		3
11	#CyberMonday	#PayPal	1
12	#EcomChat"		3
13	#FYI"		3
14	#FlashvsArrow		3
15	#GivingTuesday		3
16	#HealthCare"		3
17	#Iphone6Plus http://t.co/VRrOnfpUkG"		3
18	#Payments		3
19	#Retail		3
20	#StonerSports		3
21	#futures #forex #trader #es #aapl #gpro #eSignal http://t.co/YV1kNhuLR1"		3
22	#futures #forex #trader #es #nq #cl #aapl #gpro #finance #eSignal"		3
23	#stocks #NASDAQ #NYSE #AAPL #DOWJ http://t.co/ZMKp6xyJk4"		3
24	#stocks"		3
25	#tech http://t.co/0cuTGtm7fF"		3
26	#tech http://t.co/dWX2Q7sS6F"		3
27	#tech http://t.co/dWX2Q7sS6F"		3
28	#tech http://t.co/dWX2Q7sS6F"		3
29	#tech http://t.co/dWX2Q7sS6F"		3
30	#tech http://t.co/dWX2Q7sS6F"		3
31	$AAPL #aapl		3
32	623495513	#AAPL:Th	3
33	623495514	"RT @JPD	3
34	623495515	My cat onl	1
35	623495516	I agree wi	3
36	623495517	Nobody e	3
37	623495518	#AAPL:5 R	3

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).



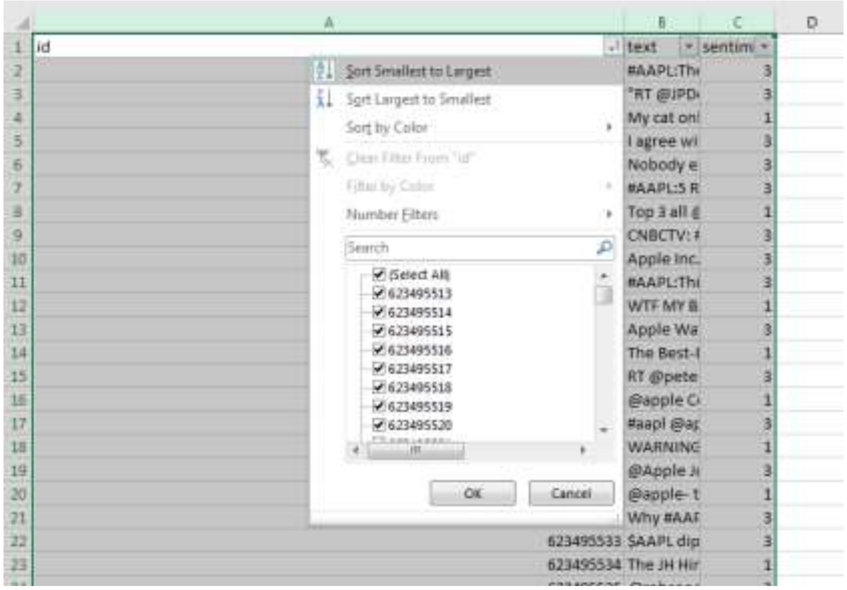
Filter (Ctrl+Shift+L)

Turn on filtering for the selected cells.

Then, click the arrow in the column header to narrow down the data.

	A	B	C
1	id	text	sentim
2			3
3			3
4			3
5			3
6			3
7			3
8	#3YearsOf5SOS		3
9	#AAPL		3
10	#Banking"		3
11	#CyberMonday	#PayPal	1
12	#EcomChat"		3

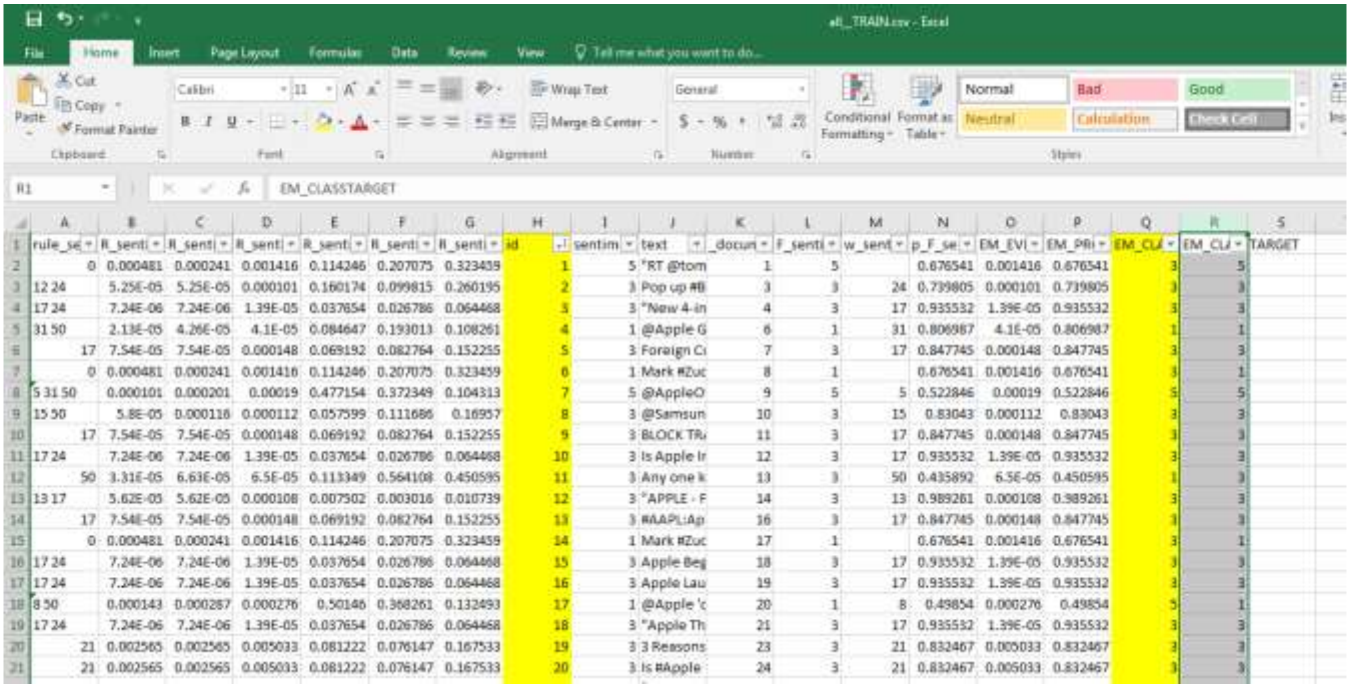
Fig.18.a: *filter the merged data to get the wanted rows & columns*



	id	text	sentim
2		#AAPL:Th	3
3		"RT @JPD	3
4		My cat on	1
5		I agree wi	3
6		Nobody e	3
7		#AAPL:5 R	3
8		Top 3 all	1
9		CNBCTV: #	3
10		Apple Inc.	3
11		#AAPL:Thi	3
12		WTF MY B	1
13		Apple Wa	3
14		The Best-I	1
15		RT @pete	3
16		@apple C	1
17		#aapl @ap	3
18		WARNING	1
19		@Apple Ji	3
20		@apple- t	1
21		Why #AAP	3
22	623495533	$AAPL dip	3
23	623495534	The JH Hir	1

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

- <u>*Step 11b* (2nd way):</u> 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).

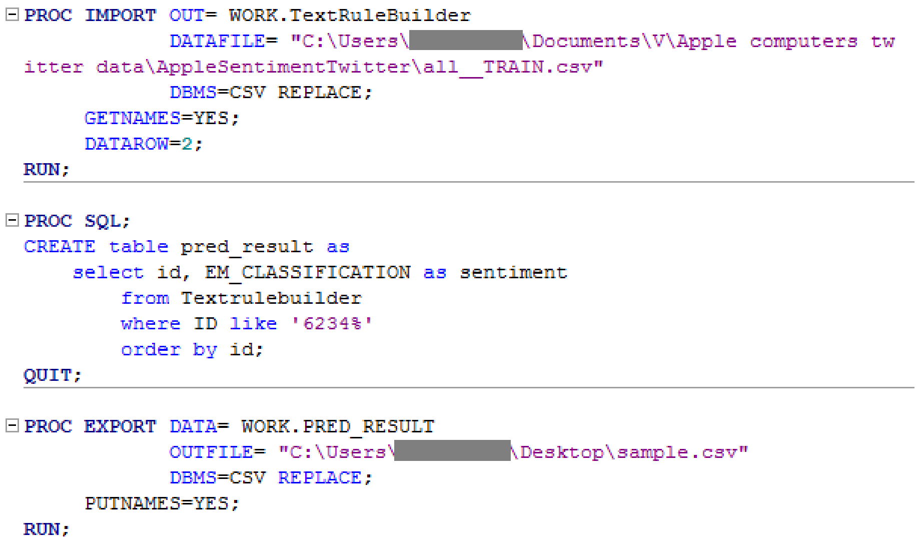


EM_CLASSTARGET

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	rule_se	R_senti	R_senti	R_senti	R_senti	R_senti	R_senti	id	sentim	text	_docun	F_senti	w_sent	p_F_se	EM_EVI	EM_PRI	EM_CLA	EM_CLA	TARGET
2	0	0.000481	0.000241	0.001416	0.114246	0.207075	0.323459	1	5	"RT @tom	1	5		0.676541	0.001416	0.676541	3	5	
3	12 24	5.25E-05	5.25E-05	0.000101	0.160174	0.099815	0.260195	2	3	Pop up #B	3	3	24	0.739805	0.000101	0.739805	3	3	
4	17 24	7.24E-06	7.24E-06	1.39E-05	0.037654	0.026786	0.064468	3	3	"New 4-in	4	3	17	0.935532	1.39E-05	0.935532	3	3	
5	31 50	2.13E-05	4.26E-05	4.1E-05	0.084647	0.193013	0.108261	4	1	@Apple G	6	1	31	0.806987	4.1E-05	0.806987	1	1	
6	17	7.54E-05	7.54E-05	0.000148	0.069192	0.082764	0.152255	5	3	Foreign C	7	3	17	0.847745	0.000148	0.847745	3	3	
7	0	0.000481	0.000241	0.001416	0.114246	0.207075	0.323459	6	1	Mark #Zuc	8	1		0.676541	0.001416	0.676541	3	1	
8	5 31 50	0.000101	0.000201	0.00019	0.477154	0.372349	0.104313	7	5	@AppleO	9	5	5	0.522846	0.00019	0.522846	5	5	
9	15 50	5.8E-05	0.000116	0.000112	0.057599	0.111686	0.16957	8	3	@Samsun	10	3	15	0.83043	0.000112	0.83043	3	3	
10	17	7.54E-05	7.54E-05	0.000148	0.069192	0.082764	0.152255	9	3	BLOCK TR	11	3	17	0.847745	0.000148	0.847745	3	3	
11	17 24	7.24E-06	7.24E-06	1.39E-05	0.037654	0.026786	0.064468	10	3	Is Apple Ir	12	3	17	0.935532	1.39E-05	0.935532	3	3	
12	50	3.31E-05	6.63E-05	6.5E-05	0.113349	0.564108	0.450595	11	3	Any one k	13	3	50	0.435892	6.5E-05	0.450595	1	3	
13	13 17	5.62E-05	5.62E-05	0.000108	0.007502	0.003016	0.010739	12	3	"APPLE - F	14	3	13	0.989261	0.000108	0.989261	3	3	
14	17	7.54E-05	7.54E-05	0.000148	0.069192	0.082764	0.152255	13	3	#AAPL:Ap	16	3	17	0.847745	0.000148	0.847745	3	3	
15	0	0.000481	0.000241	0.001416	0.114246	0.207075	0.323459	14	1	Mark #Zuc	17	1		0.676541	0.001416	0.676541	3	1	
16	17 24	7.24E-06	7.24E-06	1.39E-05	0.037654	0.026786	0.064468	15	3	Apple Beg	18	3	17	0.935532	1.39E-05	0.935532	3	3	
17	17 24	7.24E-06	7.24E-06	1.39E-05	0.037654	0.026786	0.064468	16	3	Apple Lau	19	3	17	0.935532	1.39E-05	0.935532	3	3	
18	8 50	0.000143	0.000287	0.000276	0.50146	0.368261	0.132493	17	1	@Apple 'c	20	1	8	0.49854	0.000276	0.49854	5	1	
19	17 24	7.24E-06	7.24E-06	1.39E-05	0.037654	0.026786	0.064468	18	3	"Apple Th	21	3	17	0.935532	1.39E-05	0.935532	3	3	
20	21	0.002565	0.002565	0.005033	0.081222	0.076147	0.167533	19	3	3 Reasons	23	3	21	0.832467	0.005033	0.832467	3	3	
21	21	0.002565	0.002565	0.005033	0.081222	0.076147	0.167533	20	3	Is #Apple	24	3	21	0.832467	0.005033	0.832467	3	3	

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

- <u>*Step 11c* (3rd way):</u> *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).

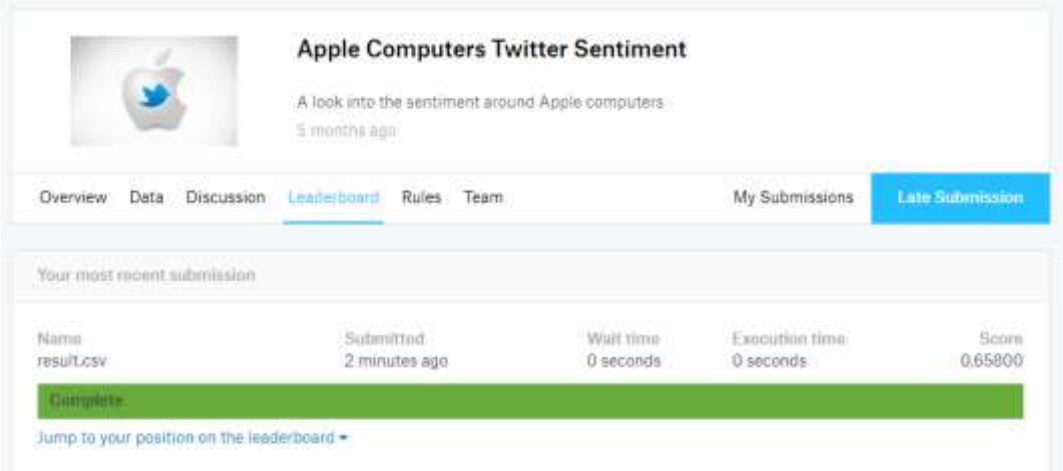


```
PROC IMPORT OUT= WORK.TextRuleBuilder
            DATAFILE= "C:\Users\[redacted]\Documents\V\Apple computers tw
itter data\AppleSentimentTwitter\all__TRAIN.csv"
            DBMS=CSV REPLACE;
     GETNAMES=YES;
     DATAROW=2;
RUN;

PROC SQL;
CREATE table pred_result as
    select id, EM_CLASSIFICATION as sentiment
        from Textrulebuilder
        where ID like '6234%'
        order by id;
QUIT;

PROC EXPORT DATA= WORK.PRED_RESULT
            OUTFILE= "C:\Users\[redacted]\Desktop\sample.csv"
            DBMS=CSV REPLACE;
     PUTNAMES=YES;
RUN;
```

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:



Apple Computers Twitter Sentiment

A look into the sentiment around Apple computers

5 months ago

Your most recent submission

Name
result.csv

Submitted
2 minutes ago

Wait time
0 seconds

Execution time
0 seconds

Score
0.65800

Complete

Jump to your position on the leaderboard ▾

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.

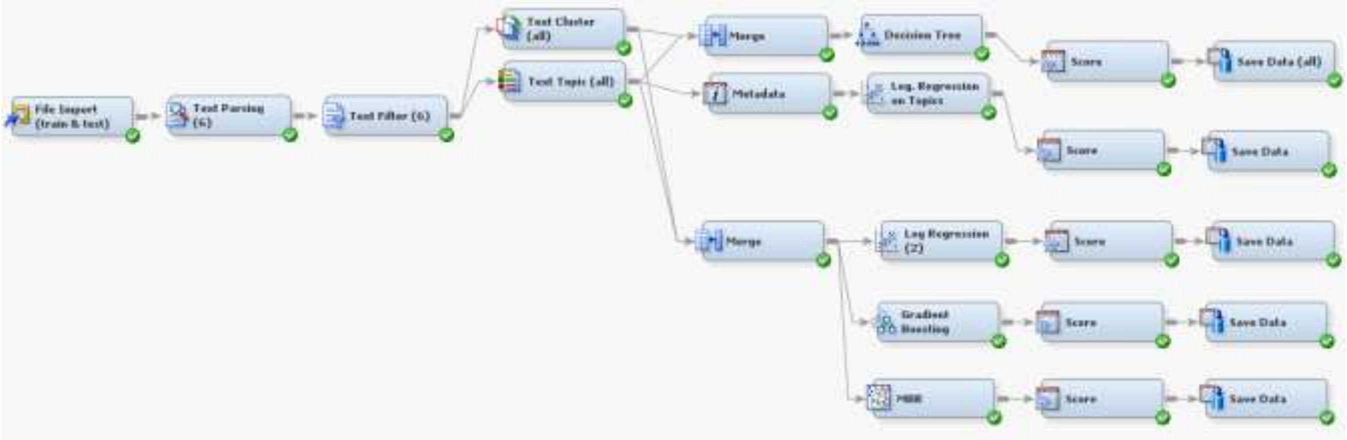


Fig.22: *another diagram for other models*

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.

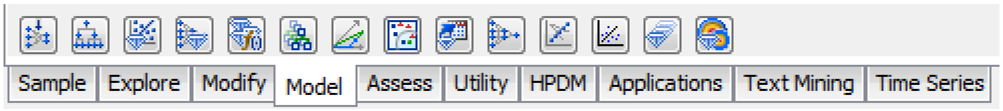


Fig.23: Model *tab which includes node icon for various models.*

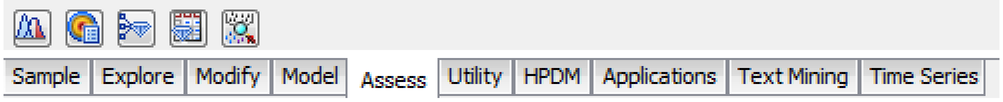


Fig.24: Access *tab which includes node icon for accessing models*

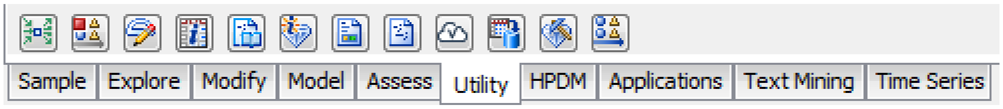
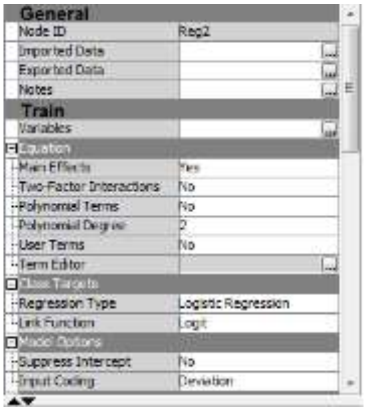


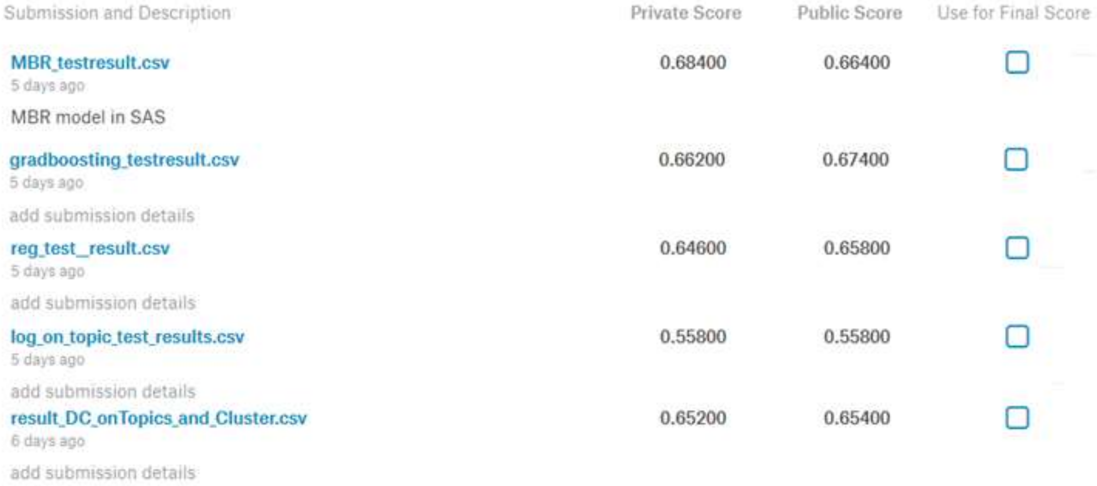
Fig.25: Utility *tab which includes node icon for model utilities*



General	
Node ID	Reg2
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Equation	
Main Effects	Yes
Two-Factor Interactions	No
Polynomial Terms	No
Polynomial Degree	2
User Terms	No
Term Editor	
Class Targets	
Regression Type	Logistic Regression
Link Function	Logit
Model Options	
Suppress Intercept	No
Input Coding	Deviation

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:



Submission and Description	Private Score	Public Score	Use for Final Score
MBR_testresult.csv 5 days ago MBR model in SAS	0.68400	0.66400	□
gradboosting_testresult.csv 5 days ago add submission details	0.66200	0.67400	□
reg_test_result.csv 5 days ago add submission details	0.64600	0.65800	□
log_on_topic_test_results.csv 5 days ago add submission details	0.55800	0.55800	□
result_DC_onTopics_and_Cluster.csv 6 days ago add submission details	0.65200	0.65400	□

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

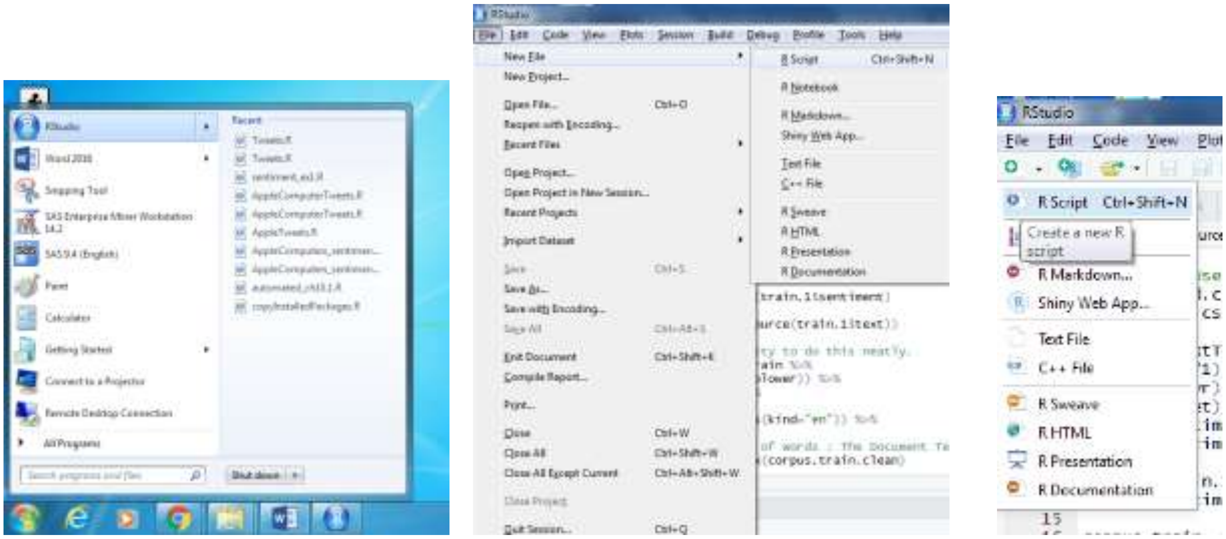
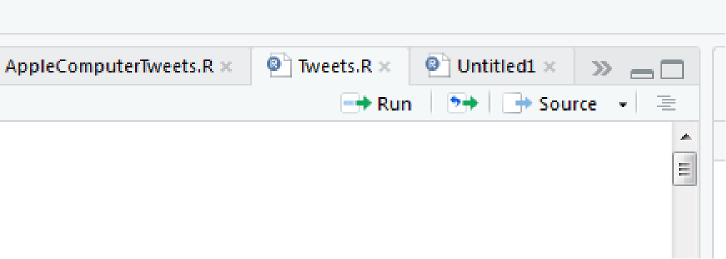


Fig. 1: how to create a R script file

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)



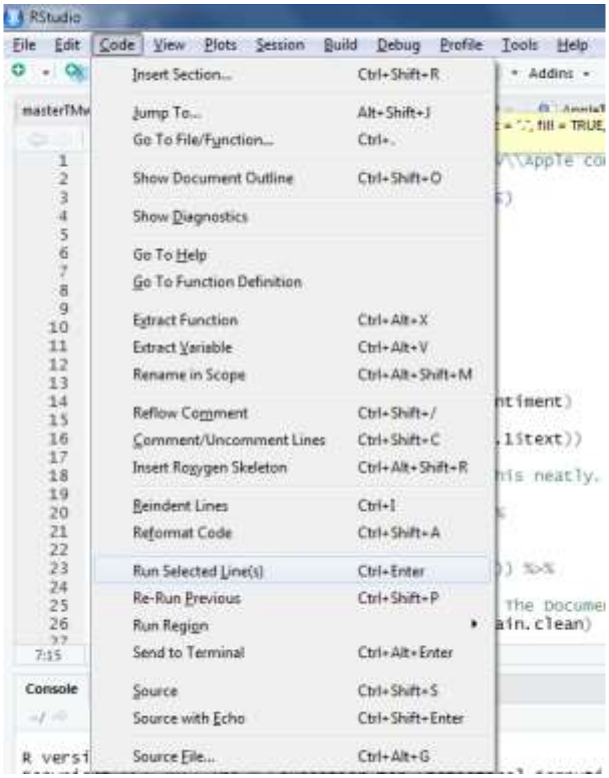
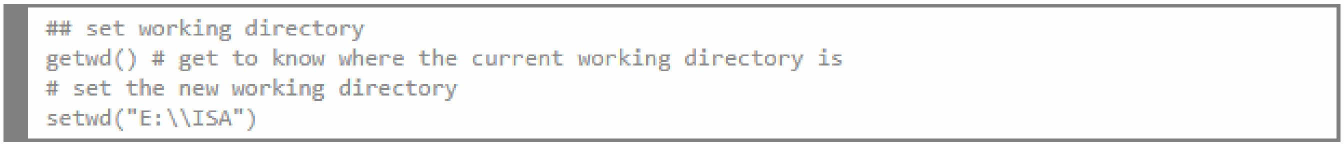

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Addins
Insert Section... Ctrl+Shift+R
Jump To... Alt+Shift+J
Go To File/Function... Ctrl+.
Show Document Outline Ctrl+Shift+O
Show Diagnostics
Go To Help
Go To Function Definition
Extract Function Ctrl+Alt+X
Extract Variable Ctrl+Alt+V
Rename in Scope Ctrl+Alt+Shift+M
Reflow Comment Ctrl+Shift+/
Comment/Uncomment Lines Ctrl+Shift+C
Insert Roxygen Skeleton Ctrl+Alt+Shift+R
Reindent Lines Ctrl+I
Reformat Code Ctrl+Shift+A
Run Selected Line(s) Ctrl+Enter
Re-Run Previous Ctrl+Shift+P
Run Region
Send to Terminal Ctrl+Alt+Enter
Source Ctrl+Shift+S
Source with Echo Ctrl+Shift+Enter
Source File... Ctrl+Alt+G
Console

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:

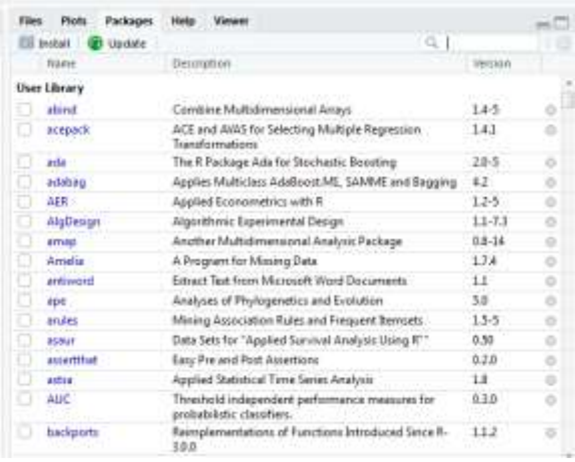


```
## set working directory
getwd() # get to know where the current working directory is
# set the new working directory
setwd("E:\\ISA")
```

Fig. 3: *setting working directory*

Load necessary libraries:

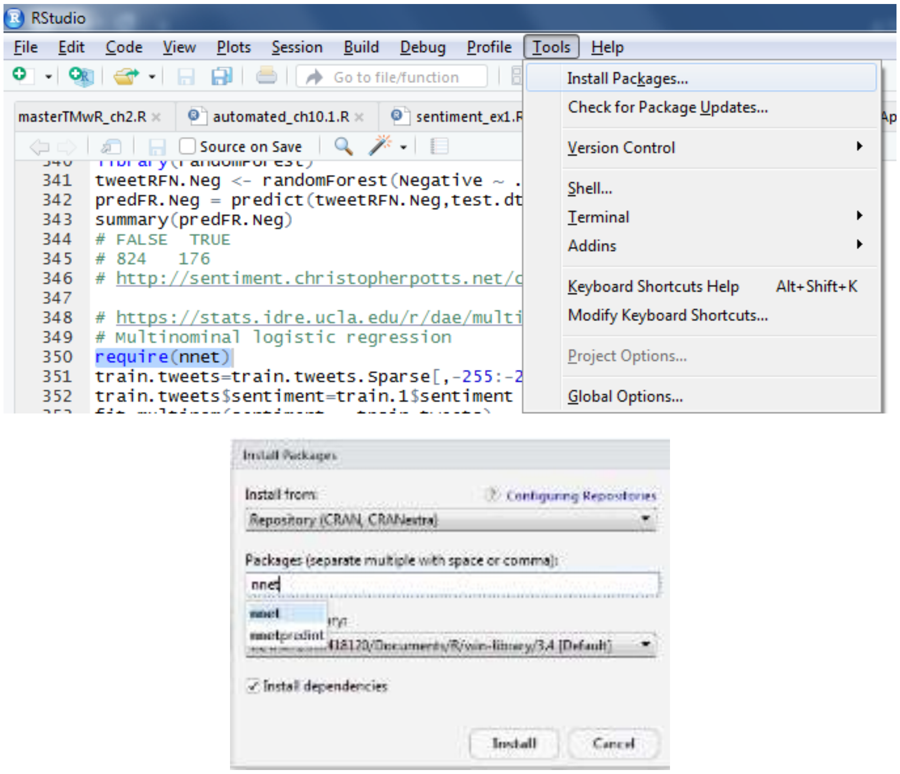
First you need to check if your needed libraries are already installed yet. You can do that by looking at the window located at the bottom right of the R Studio. You can type a specific package name into the search box of that window to see if it is there.



Name	Description	Version
User Library		
abind	Combine Multidimensional Arrays	1.4-5
acepack	ACE and AVAS for Selecting Multiple Regression Transformations	1.4.1
ada	The R Package Ada for Stochastic Boosting	2.0-5
adabag	Applies Multiclass AdaBoost.M1, SAMME and Bagging	4.2
AER	Applied Econometrics with R	1.2-5
AlgDesign	Algorithmic Experimental Design	1.1-7.3
amap	Another Multidimensional Analysis Package	0.8-14
Amelia	A Program for Missing Data	1.7.4
antiword	Extract Text from Microsoft Word Documents	1.1
ape	Analyses of Phylogenetics and Evolution	5.0
arules	Mining Association Rules and Frequent Itemsets	1.5-5
asaur	Data Sets for "Applied Survival Analysis Using R""	0.50
assertthat	Easy Pre and Post Assertions	0.2.0
astsa	Applied Statistical Time Series Analysis	1.8
AUC	Threshold independent performance measures for probabilistic classifiers.	0.3.0
backports	Reimplementations of Functions Introduced Since R-3.0.0	1.1.2

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.



```
tweetRFN.Neg <- randomForest(Negative ~ .
predFR.Neg = predict(tweetRFN.Neg,test.dt
summary(predFR.Neg)
# FALSE  TRUE
# 824    176
# http://sentiment.christopherpotts.net/c

# https://stats.idre.ucla.edu/r/dae/multi
# Multinominal logistic regression
require(nnet)
train.tweets=train.tweets.Sparse[,-255:-2
train.tweets$sentiment=train.1$sentiment
```

Install from: Repository (CRAN, CRANextra)

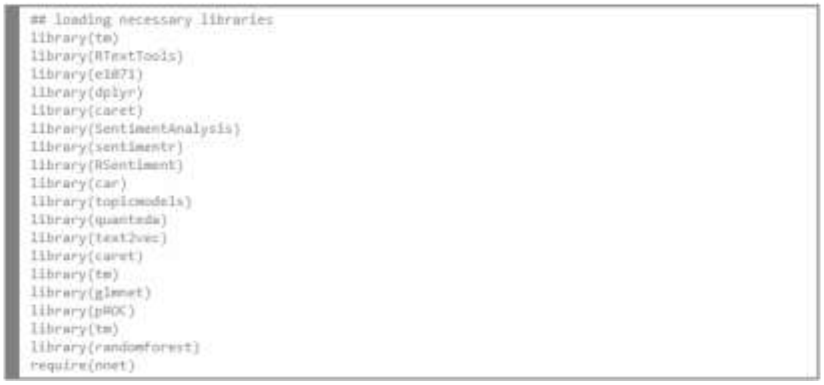
Packages (separate multiple with space or comma): nnet

nnet
nnetpredint

...418120/Documents/R/win-library/3.4 [Default]

☑ Install dependencies

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

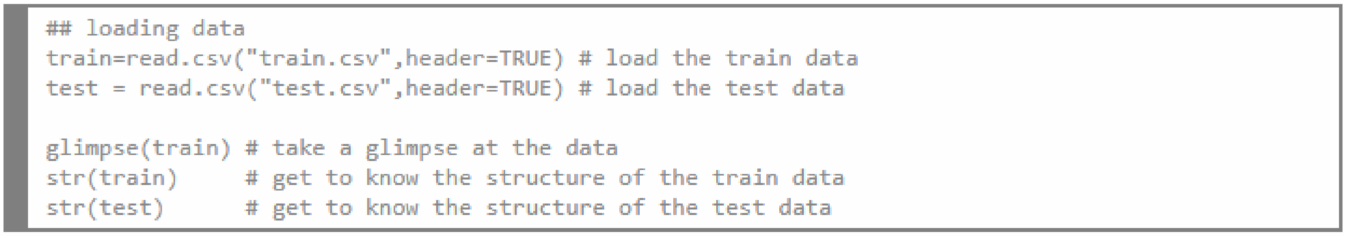


```
## loading necessary libraries
library(tm)
library(RTextTools)
library(e1071)
library(dplyr)
library(caret)
library(SentimentAnalysis)
library(sentimentr)
library(RSentiment)
library(car)
library(topicmodels)
library(quanteda)
library(text2vec)
library(caret)
library(tm)
library(glmnet)
library(pROC)
library(tm)
library(randomforest)
require(nnet)
```

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

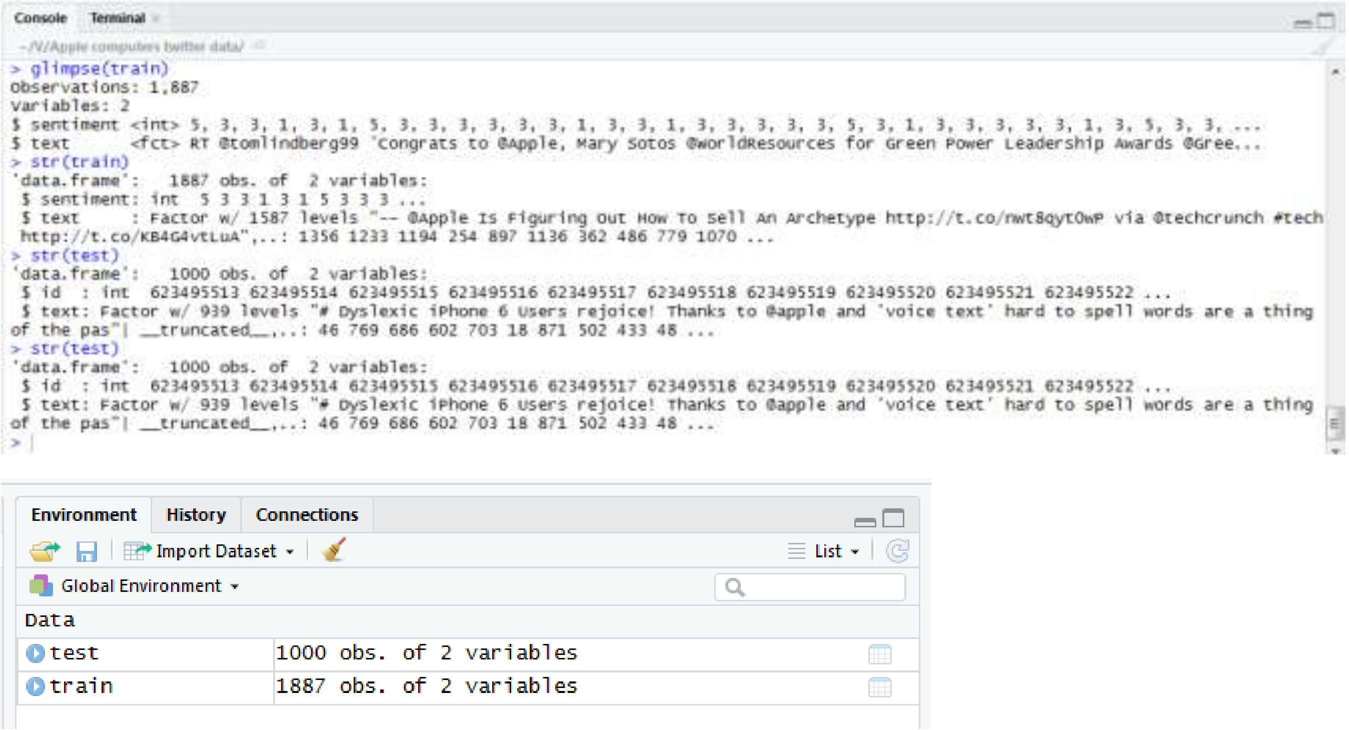
'Require' or 'library' commands are used to load the necessary packages.

__Load data (training & testing data sets):__



```
## loading data
train=read.csv("train.csv",header=TRUE) # load the train data
test = read.csv("test.csv",header=TRUE) # load the test data

glimpse(train) # take a glimpse at the data
str(train)     # get to know the structure of the train data
str(test)      # get to know the structure of the test data
```

Fig. 6: *codes how to load data*



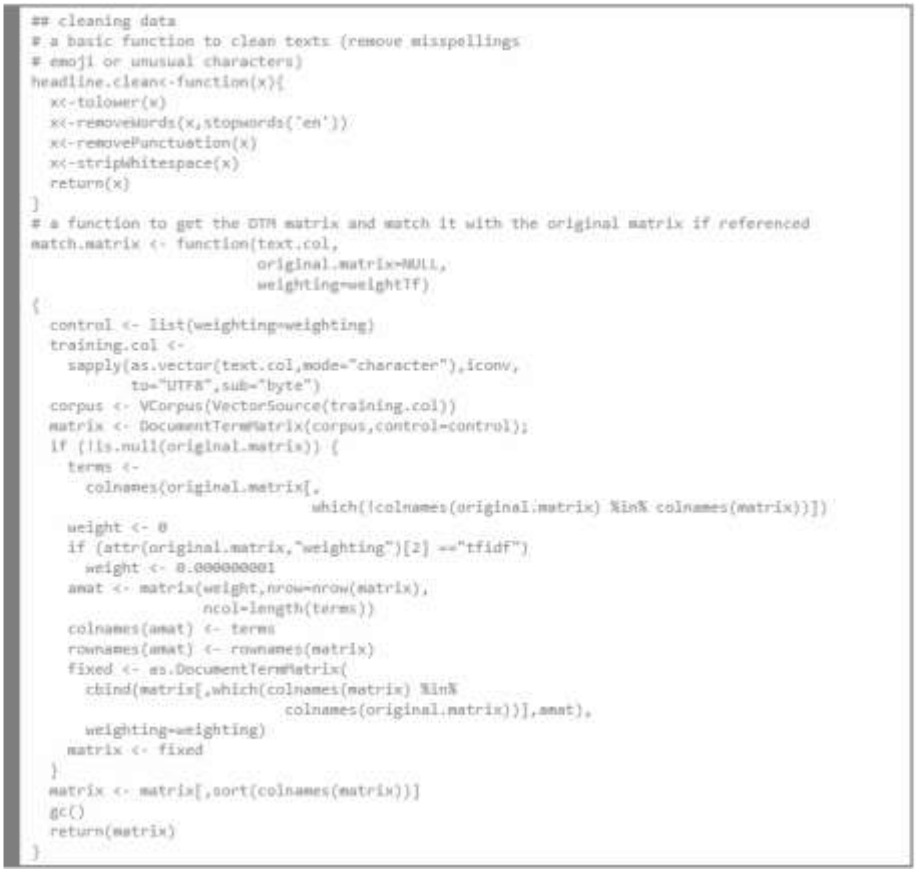
```
> glimpse(train)
Observations: 1,887
Variables: 2
$ sentiment <int> 5, 3, 3, 1, 3, 1, 5, 3, 3, 3, 3, 3, 3, 1, 3, 3, 1, 3, 3, 3, 3, 3, 5, 3, 1, 3, 3, 3, 3, 3, 1, 3, 5, 3, 3, ...
$ text      <fct> RT @tomlindberg99 'Congrats to @Apple, Mary Sotos @WorldResources for Green Power Leadership Awards @Gree...
> str(train)
'data.frame':	1887 obs. of  2 variables:
 $ sentiment: int  5 3 3 1 3 1 5 3 3 3 ...
 $ text     : Factor w/ 1587 levels "-- @Apple Is Figuring Out How To Sell An Archetype http://t.co/nwt8qytOwP via @techcrunch #tech
 http://t.co/KB4G4vtLuA",..: 1356 1233 1194 254 897 1136 362 486 779 1070 ...
> str(test)
'data.frame':	1000 obs. of  2 variables:
 $ id  : int  623495513 623495514 623495515 623495516 623495517 623495518 623495519 623495520 623495521 623495522 ...
 $ text: Factor w/ 939 levels "# Dyslexic iPhone 6 Users rejoice! Thanks to @apple and 'voice text' hard to spell words are a thing
of the pas"| __truncated__,..: 46 769 686 602 703 18 871 502 433 48 ...
> str(test)
'data.frame':	1000 obs. of  2 variables:
 $ id  : int  623495513 623495514 623495515 623495516 623495517 623495518 623495519 623495520 623495521 623495522 ...
 $ text: Factor w/ 939 levels "# Dyslexic iPhone 6 Users rejoice! Thanks to @apple and 'voice text' hard to spell words are a thing
of the pas"| __truncated__,..: 46 769 686 602 703 18 871 502 433 48 ...
> 
```

Data

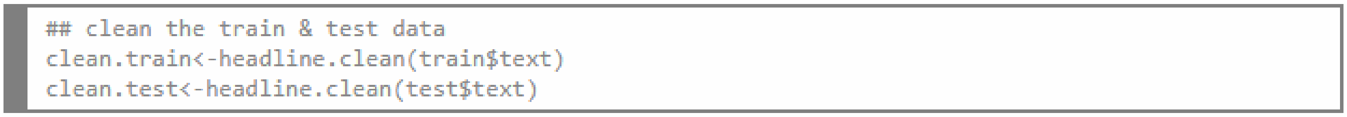
test	1000 obs. of 2 variables
train	1887 obs. of 2 variables

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

Clean data:



```
## cleaning data
# a basic function to clean texts (remove misspellings
# emoji or unusual characters)
headline.clean<-function(x){
  x<-tolower(x)
  x<-removeWords(x,stopwords('en'))
  x<-removePunctuation(x)
  x<-stripWhitespace(x)
  return(x)
}
# a function to get the DTM matrix and match it with the original matrix if referenced
match.matrix <- function(text.col,
                         original.matrix=NULL,
                         weighting=weightTf)
{
  control <- list(weighting=weighting)
  training.col <-
    sapply(as.vector(text.col,mode="character"),iconv,
           to="UTF8",sub="byte")
  corpus <- VCorpus(VectorSource(training.col))
  matrix <- DocumentTermMatrix(corpus,control=control);
  if (!is.null(original.matrix)) {
    terms <-
      colnames(original.matrix[,
                             which(!colnames(original.matrix) %in% colnames(matrix))])
    weight <- 0
    if (attr(original.matrix,"weighting")[2] =="tfidf")
      weight <- 0.000000001
    amat <- matrix(weight,nrow=nrow(matrix),
                   ncol=length(terms))
    colnames(amat) <- terms
    rownames(amat) <- rownames(matrix)
    fixed <- as.DocumentTermMatrix(
      cbind(matrix[,which(colnames(matrix) %in%
                          colnames(original.matrix))],amat),
      weighting=weighting)
    matrix <- fixed
  }
  matrix <- matrix[,sort(colnames(matrix))]
  gc()
  return(matrix)
}
```



```
## clean the train & test data
clean.train<-headline.clean(train$text)
clean.test<-headline.clean(test$text)
```

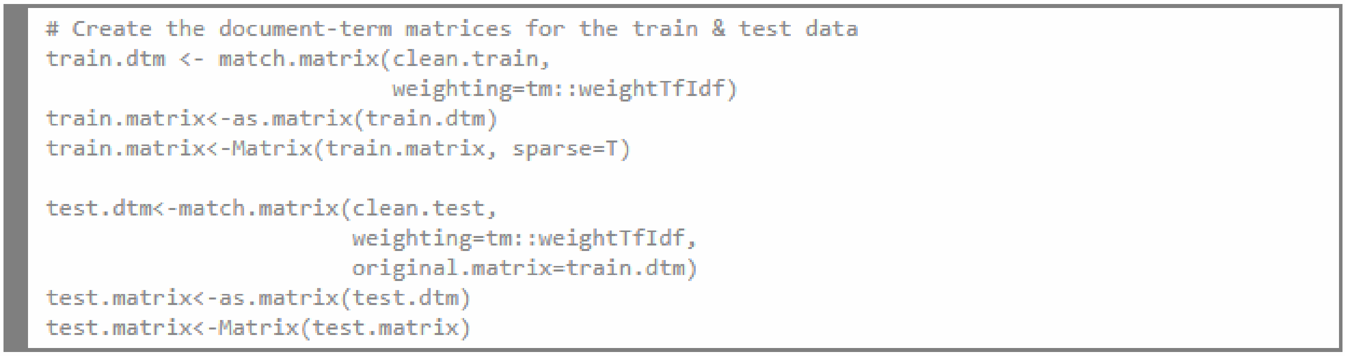
***Fig. 8.a & b:** codes of **(a)** the function **headline** (cleaning text) & **match_matrix** (creating DocumentTermMatrix & forcing two matrices to have the same number of columns); **(b)** clean texts of the train & test data*

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:

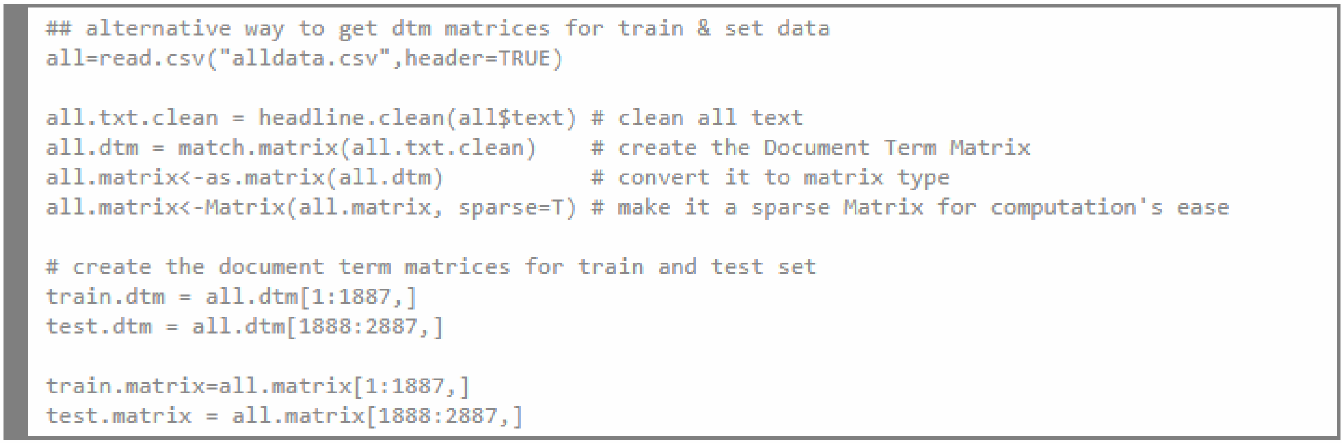
The train set and test set are cleaned, then the corresponding document-term matrices are computed and forced to have the same dimension.



```
# Create the document-term matrices for the train & test data
train.dtm <- match.matrix(clean.train,
                          weighting=tm::weightTfIdf)
train.matrix<-as.matrix(train.dtm)
train.matrix<-Matrix(train.matrix, sparse=T)

test.dtm<-match.matrix(clean.test,
                       weighting=tm::weightTfIdf,
                       original.matrix=train.dtm)
test.matrix<-as.matrix(test.dtm)
test.matrix<-Matrix(test.matrix)
```

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.

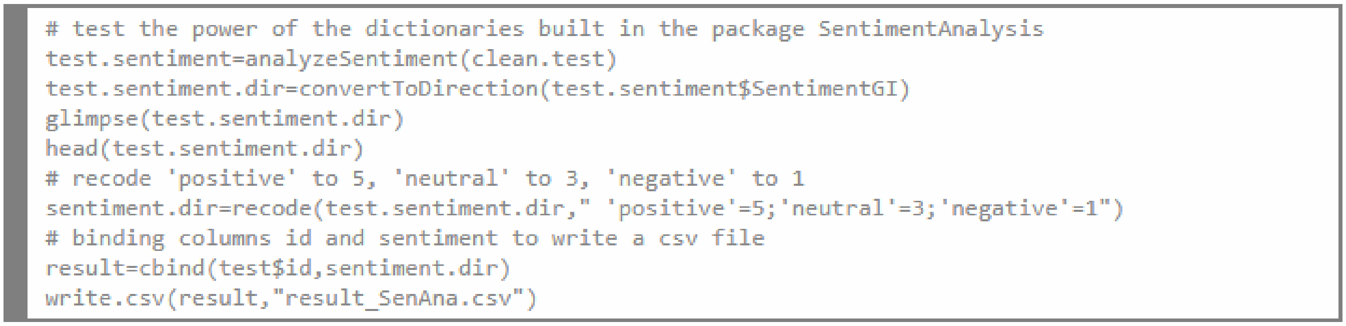


```
## alternative way to get dtm matrices for train & set data
all=read.csv("alldata.csv",header=TRUE)

all.txt.clean = headline.clean(all$text) # clean all text
all.dtm = match.matrix(all.txt.clean)    # create the Document Term Matrix
all.matrix<-as.matrix(all.dtm)           # convert it to matrix type
all.matrix<-Matrix(all.matrix, sparse=T) # make it a sparse Matrix for computation's ease

# create the document term matrices for train and test set
train.dtm = all.dtm[1:1887,]
test.dtm = all.dtm[1888:2887,]

train.matrix=all.matrix[1:1887,]
test.matrix = all.matrix[1888:2887,]
```

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*:

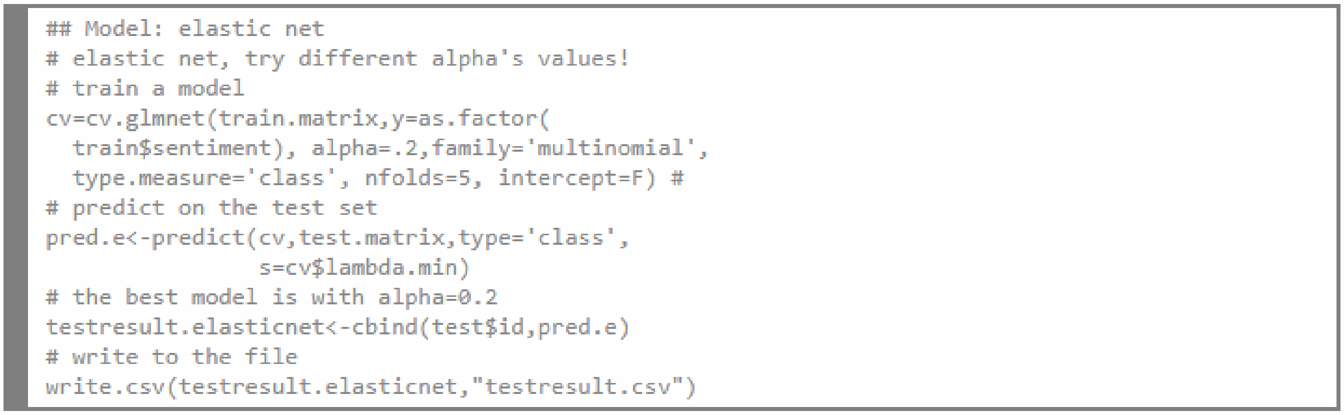


```
# test the power of the dictionaries built in the package SentimentAnalysis
test.sentiment=analyzeSentiment(clean.test)
test.sentiment.dir=convertToDirection(test.sentiment$SentimentGI)
glimpse(test.sentiment.dir)
head(test.sentiment.dir)
# recode 'positive' to 5, 'neutral' to 3, 'negative' to 1
sentiment.dir=recode(test.sentiment.dir," 'positive'=5;'neutral'=3;'negative'=1")
# binding columns id and sentiment to write a csv file
result=cbind(test$id,sentiment.dir)
write.csv(result,"result_SenAna.csv")
```

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:

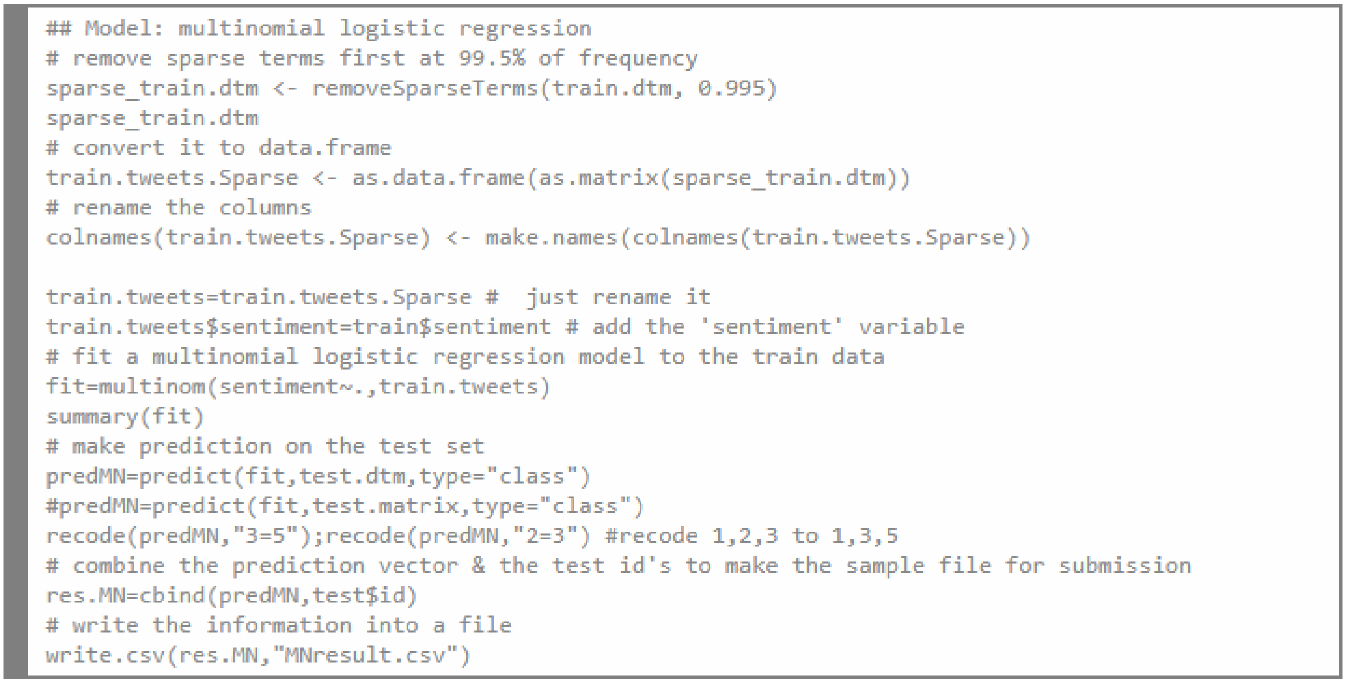


```
## Model: elastic net
# elastic net, try different alpha's values!
# train a model
cv=cv.glmnet(train.matrix,y=as.factor(
  train$sentiment), alpha=.2,family='multinomial',
  type.measure='class', nfolds=5, intercept=F) #
# predict on the test set
pred.e<-predict(cv,test.matrix,type='class',
                s=cv$lambda.min)
# the best model is with alpha=0.2
testresult.elasticnet<-cbind(test$id,pred.e)
# write to the file
write.csv(testresult.elasticnet,"testresult.csv")
```

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:

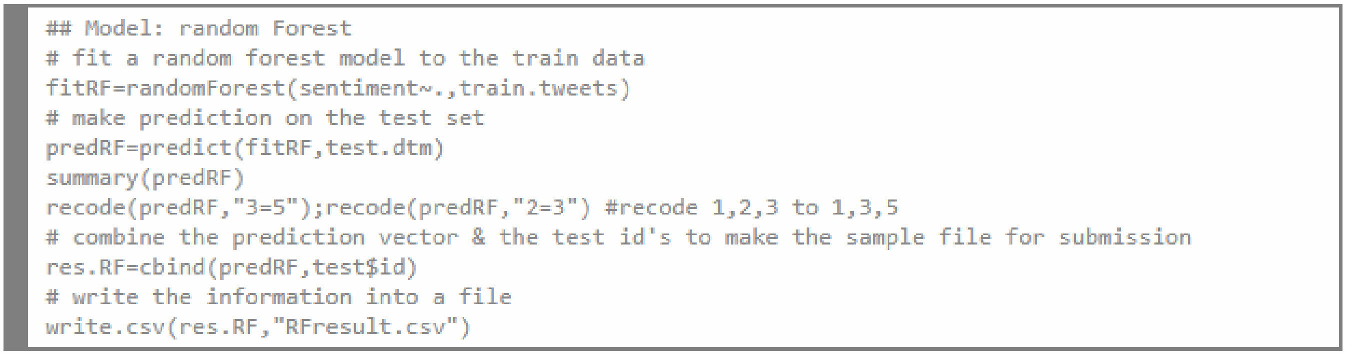


```
## Model: multinomial logistic regression
# remove sparse terms first at 99.5% of frequency
sparse_train.dtm <- removeSparseTerms(train.dtm, 0.995)
sparse_train.dtm
# convert it to data.frame
train.tweets.Sparse <- as.data.frame(as.matrix(sparse_train.dtm))
# rename the columns
colnames(train.tweets.Sparse) <- make.names(colnames(train.tweets.Sparse))

train.tweets=train.tweets.Sparse #  just rename it
train.tweets$sentiment=train$sentiment # add the 'sentiment' variable
# fit a multinomial logistic regression model to the train data
fit=multinom(sentiment~.,train.tweets)
summary(fit)
# make prediction on the test set
predMN=predict(fit,test.dtm,type="class")
#predMN=predict(fit,test.matrix,type="class")
recode(predMN,"3=5");recode(predMN,"2=3") #recode 1,2,3 to 1,3,5
# combine the prediction vector & the test id's to make the sample file for submission
res.MN=cbind(predMN,test$id)
# write the information into a file
write.csv(res.MN,"MNresult.csv")
```

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:



```
## Model: random Forest
# fit a random forest model to the train data
fitRF=randomForest(sentiment~.,train.tweets)
# make prediction on the test set
predRF=predict(fitRF,test.dtm)
summary(predRF)
recode(predRF,"3=5");recode(predRF,"2=3") #recode 1,2,3 to 1,3,5
# combine the prediction vector & the test id's to make the sample file for submission
res.RF=cbind(predRF,test$id)
# write the information into a file
write.csv(res.RF,"RFresult.csv")
```

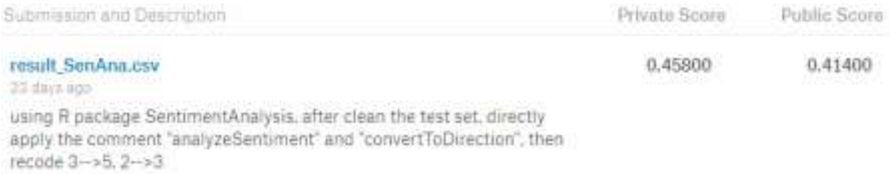
__Fig. 13:__ codes how to build the __random forest__ model

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.

<u>*Submit the results to kaggle.com:*</u>

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

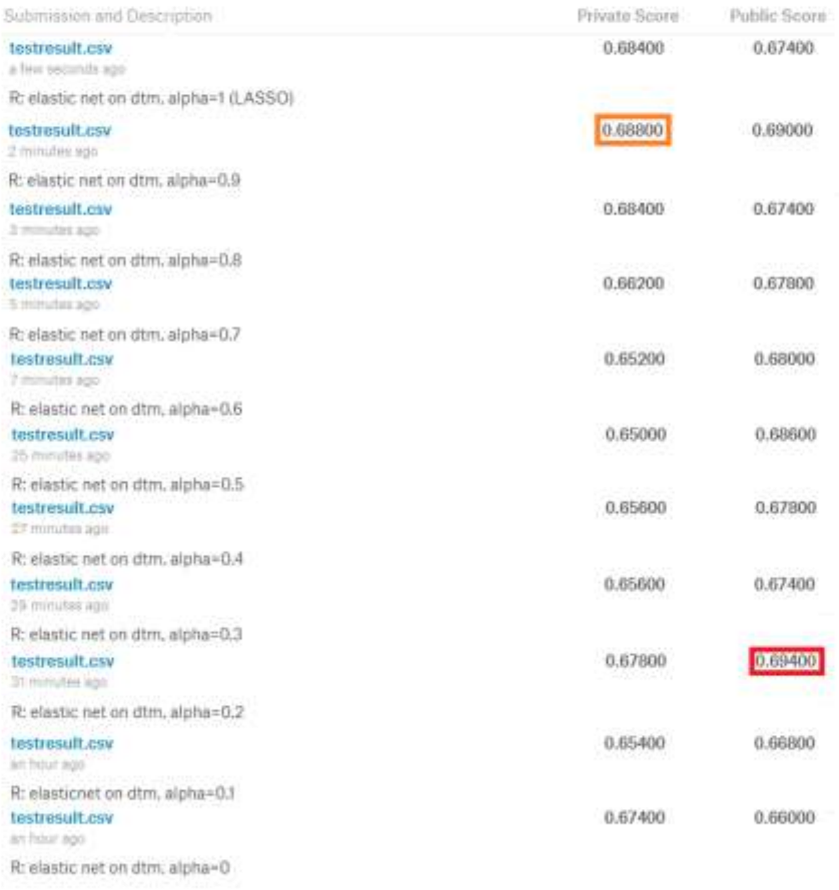
file result_SenAna.csv got scores quite low: private score of 45.8% & public score got 41.4%.



Submission and Description	Private Score	Public Score
result_SenAna.csv 23 days ago using R package SentimentAnalysis, after clean the test set, directly apply the comment "analyzeSentiment" and "convertToDirection", then recode 3-->5, 2-->3	0.45800	0.41400

__Fig. 14:__ result from kaggle.com for this trial

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



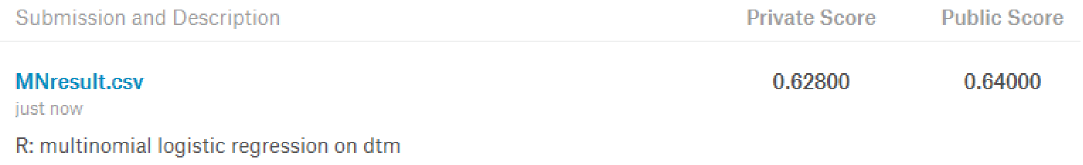
Submission and Description	Private Score	Public Score
testresult.csv a few seconds ago R: elastic net on dtm, alpha=1 (LASSO)	0.68400	0.67400
testresult.csv 2 minutes ago R: elastic net on dtm, alpha=0.9	0.68800	0.69000
testresult.csv 3 minutes ago R: elastic net on dtm, alpha=0.8	0.68400	0.67400
testresult.csv 5 minutes ago R: elastic net on dtm, alpha=0.7	0.66200	0.67800
testresult.csv 7 minutes ago R: elastic net on dtm, alpha=0.6	0.65200	0.68000
testresult.csv 25 minutes ago R: elastic net on dtm, alpha=0.5	0.65000	0.68600
testresult.csv 27 minutes ago R: elastic net on dtm, alpha=0.4	0.65600	0.67800
testresult.csv 29 minutes ago R: elastic net on dtm, alpha=0.3	0.65600	0.67400
testresult.csv 31 minutes ago R: elastic net on dtm, alpha=0.2	0.67800	0.69400
testresult.csv an hour ago R: elasticnet on dtm, alpha=0.1	0.65400	0.66800
testresult.csv an hour ago R: elastic net on dtm, alpha=0	0.67400	0.66000

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:**

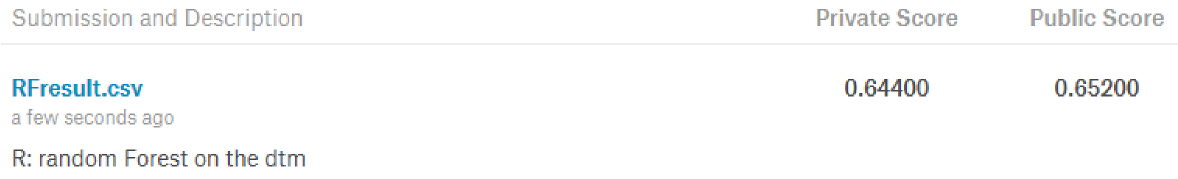
The train set's document term matrix is removed rare words by 99.5%. After that, fitting a multinomial regression on this document term matrix with the response is the sentiment variable of the train set. Using this model to predict the sentiment of the test set (the document-term matrix of the test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



Submission and Description	Private Score	Public Score
MNresult.csv just now R: multinomial logistic regression on dtm	0.62800	0.64000

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

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



Submission and Description	Private Score	Public Score
RFresult.csv a few seconds ago R: random Forest on the dtm	0.64400	0.65200

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.