

text analytics lab

The lab uses R programming to work with text data. Programs are coded to clean text, remove stopwords, and apply Porter Stemming to the remaining words. An Azure ML web service will be deployed to classify Tweets based on Sentiment analysis.

The lab works with a set of 160,000 tweets, which include sentiment labels.

Social media sentiment is an important indicator of public opinion. Determining sentiment can be valuable in a number of applications including brand awareness, product launches, and detecting political trends.

Raw text is inherently messy. Machine understanding and analysis is inhibited by the presence of extraneous symbols and words that clutter the text. The exact nature of the required text cleaning depends on the application. In this case, the lab focuses on text cleaning to facilitate sentiment classification. The presence of certain words determines the sentiment of the tweet. Words and symbols which are extraneous to this purpose are distractions at best, and a likely source of noise in the analysis. The following steps prepare the tweet text for analysis:

- Symbols and unnecessary white space which do not convey sentiment are removed, leaving only alphabetic characters.
- There is no difference in the sentiment conveyed by a word in upper case or lower case, so all case is set to lower.
- Stop words are words that occur with high frequency in text, but do not have any particular meaning. Examples include word like "the", "and", and "this". Since these words are relatively common, yet communicate no particular sentiment, they can bias analytics. Therefore, stopwords which do not convey sentiment are therefore removed from the tweet text.
- A stem is a root word. For example, "go" is the root word of conjugated verbs, "going", "goes", and "gone". The meaning of these words is the same in terms of analysis. A process known as stemming is applied to transform words to their roots, before analysis.

The **tweets.csv** and **stopwords.csv** files are loaded into the Azure Machine Learning Studio.

Load and transform the tweet data

The data is then loaded and column names are set to convenient values in R in a Jupyter Notebook.

	sentiment	tweets
1	4	@elephantbird Hey dear, Happy Friday to You Already had your rice's bowl for lunch ?
2	4	Ughhh layin downnnn Waiting for zeina to cook breakfast
3	0	@greeniebach I reckon he'll play, even if he's not 100%...but i know nothing!! ;) It won't be the same without him.
4	0	@vaLewee I know! Saw it on the news!
5	0	very sad that http://www.fabchannel.com/ has closed down. One of the few web services that I've used for over 5 years
6	0	@Fearnecotton who sings 'I Remember'? i alwaysss hear it on Radio 1 but never catch the artist

- .. The **Sentiment** column contains a sentiment score {0,4} for -/+ sentiment of the tweet.
- .. The **Tweets** column contains the actual text of the tweet.

In order to work with the text in the tweets using the tools in the R **tm** package, they must be converted to a **corpus object**. A **tm vector corpus** is a vector of corpus objects. In this case, the text of each tweet is a single corpus.

Clean and Lower Case Symbols with R

The R Text Mining package **tm** is used to perform some basic filtering and cleaning of the tweet text

- .. Remove numbers.
- .. Remove punctuation.
- .. Remove excess white space.
- .. Convert to lower case.

The code uses the **tm_map** function to apply the specified transformation to the text.

A **term-document matrix (TDM)** of the tweets is a sparse matrix structure with the words (terms) in the rows and documents which may or may not contain that word in the columns. The count of word occurrence of the document is contained in the cells.

The frequency of words is simply computed by summing the values in the rows. Note that the **row_sums** function from the **slam** package is used to deal with the sparse matrix structure. A **dataframe** is then constructed with the words and their frequencies in descending order.

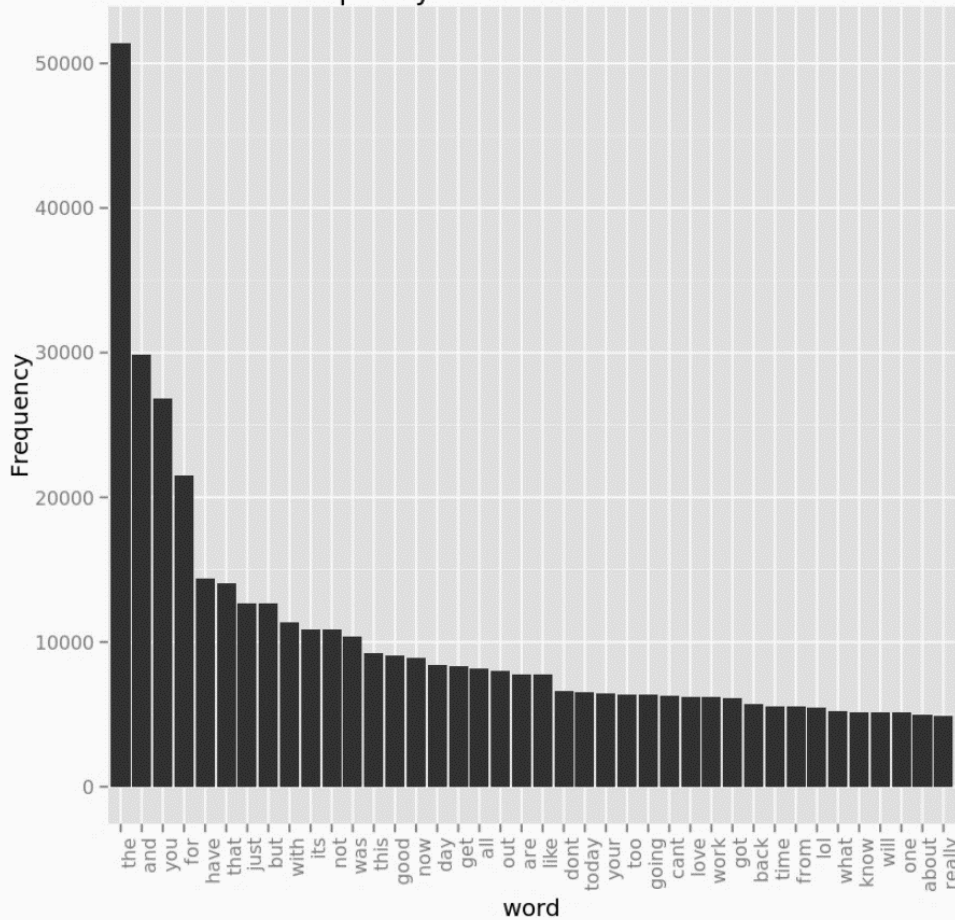
	word	freq	Cum
the	the	51366	0.03266961
and	and	29900	0.05168649
you	you	26847	0.06876162
for	for	21508	0.08244106
have	have	14357	0.09157234
that	that	14059	0.1005141
just	just	12674	0.108575
but	but	12651	0.1166212
with	with	11401	0.1238724
its	its	10912	0.1308126

not	not	10835	0.1377039
was	was	10348	0.1442854
this	this	9234	0.1501583
good	good	9076	0.1559308
now	now	8935	0.1616136
day	day	8416	0.1669663
get	get	8296	0.1722427
all	all	8148	0.177425
out	out	8013	0.1825214
are	are	7777	0.1874677

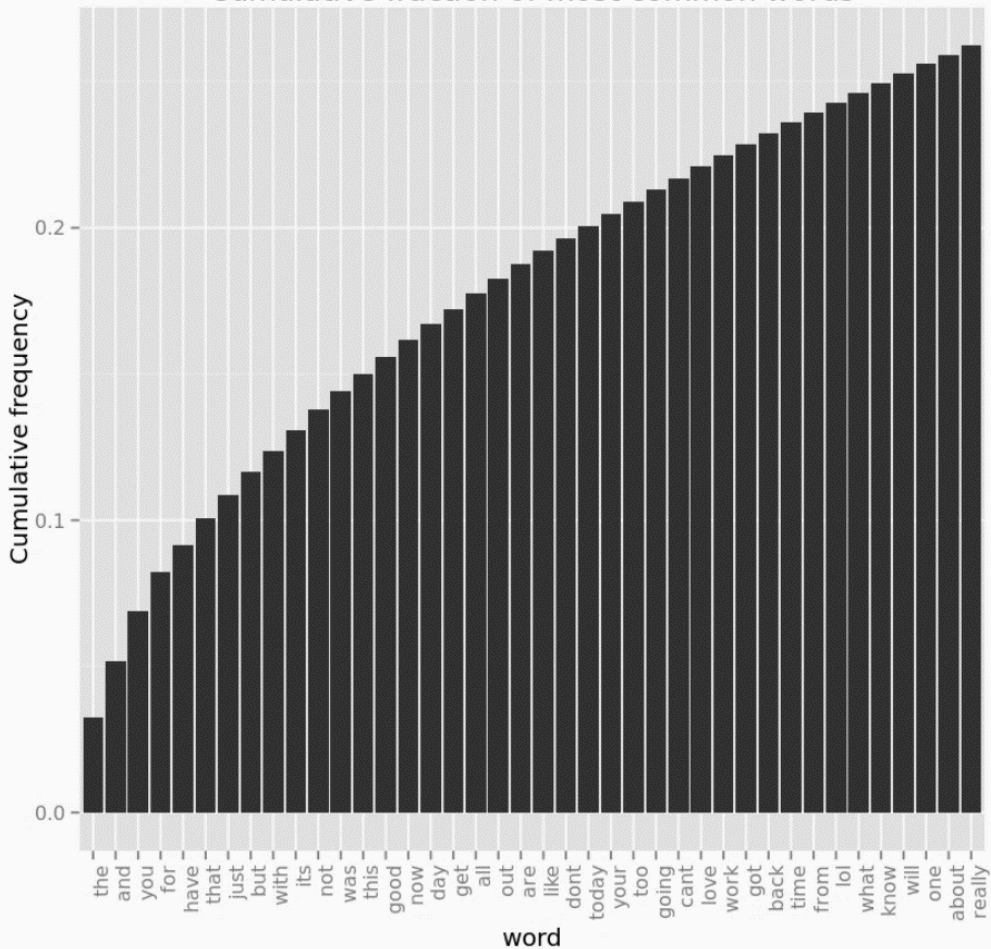
Notice that the most frequent words are in the head of this data frame. Of these 20 most frequent words, only one, 'good', is likely to convey much information on sentiment.

Many of the most frequent words are stop words, such as "the", "and", and "you", which are not likely to be helpful in determining sentiment. Also, the frequency of the words drops off fairly quickly to less than 500 out of the 160,000 tweets.

Frequency of most common words



Cumulative fraction of most common words



Another tool for examining the frequency of words in a corpus of documents is the **cumulative distribution frequency (CDF)** plot:

The conclusions one can draw from the second chart are largely the same as the first. The most frequent words are stop words and the frequency of words drops off rather quickly. Also notice, that the frequency of the words becomes uniform fairly quickly.

Finally, the normalized text in the processed tweets is illustrated as follows:

	sentiment	text
tweets.content1	4	elephantbird hey dear happy friday to you already had your rices bowl for lunch
tweets.content2	4	ughhh layin downnnn waiting for zeina to cook breakfast
tweets.content3	0	greeniebach i reckon hell play even if hes not but i know nothing it wont be the same without him
tweets.content4	0	valewee i know saw it on the news
tweets.content5	0	very sad that httpwwwfabchannelcom has closed down one of the few web services that ive used for over years
tweets.content6	0	fearnecotton who sings i remember i alwaysss hear it on radio but never catch the artist

All text is lower case and there are no numbers, punctuation or special characters.

Examine the head of the resulting word frequency data frame to determine the following:

- What is the percentage of all words for these first 20 words?
- Of these 20 words, how many are likely to contribute sentiment information?
- Are these 20 words different from the words seen for the raw text?

Remove Stopwords

The extraneous characters and whitespace have been removed from the tweet text. The results show that the most frequent words do not communicate much sentiment information. These frequent words, which are largely extraneous, are known as stop words and should be removed from the text before further analysis:

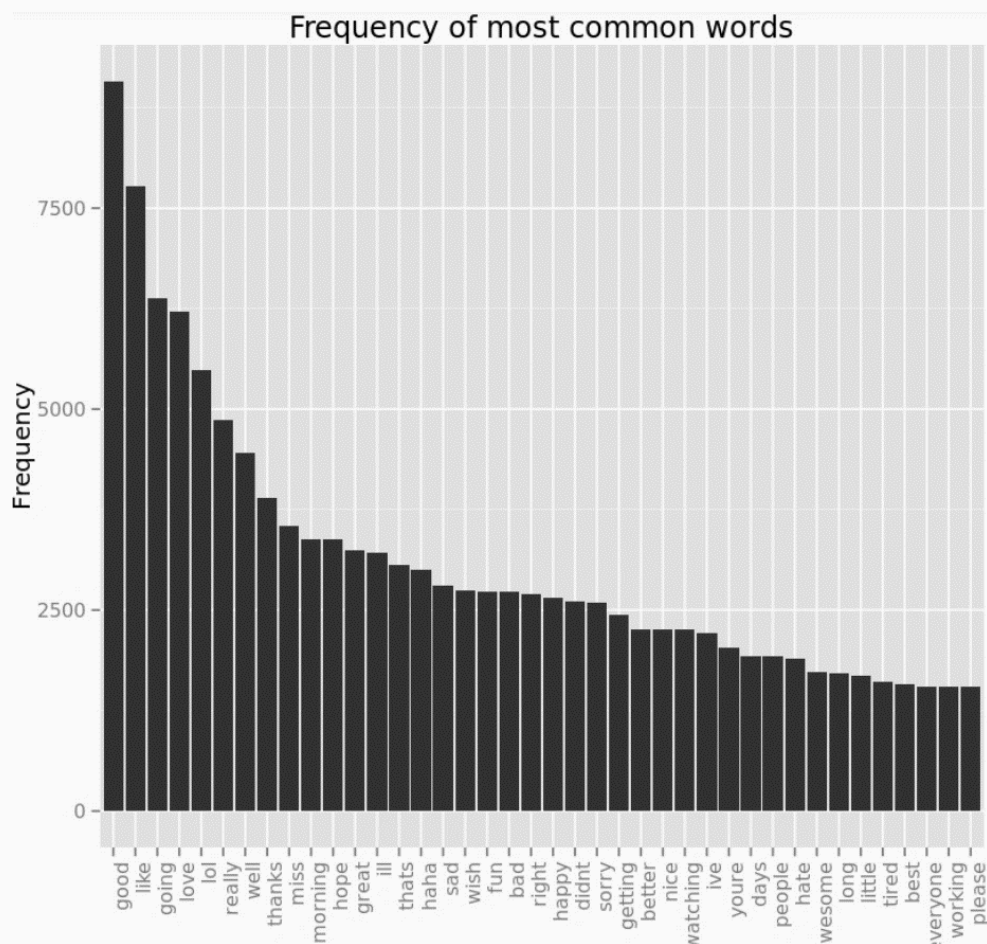
"a" "about" "above" "actual" "after" "again" "against" "all" "alreadi" "also" "alway" "am" "amp" "an" "and" "ani" "anoth"
 "any" "anyth" "are" "aren't" "around" "as" "at" "aww" "babi" "back" "be" "becaus" "because" "bed" "been" "befor" "before"
 "being" "below" "between" "birthday" "bit" "book" "both" "boy" "but" "by" "call" "can" "can't" "cannot" "cant" "car" "check"
 "com" "come" "could" "couldn't" "day" "did" "didn" "didn't" "dinner" "do" "doe" "does" "doesn" "doesn't" "doing" "don" "don't"
 "done" "dont" "down" "during" "each" "eat" "end" "even" "ever" "everyon" "exam" "famili" "feel" "few" "final" "find" "first"
 "follow" "for" "for." "found" "friday" "from" "further" "game" "get" "girl" "give" "gone" "gonna" "got" "gotta"

- These words are generally common in English language text.
- None of these words seem likely to indicate any particular sentiment.
- Some of these words, like 'aww', are specialized to this application of analyzing tweets.

The next code applies the **removeWords** operation to the tweet text:

The distribution of word frequency is not quite different. Note that many of the most frequent words are now likely to convey some sentiment, such as "good", "like", and "love". Evidently, removing stop words has had the desired effect.

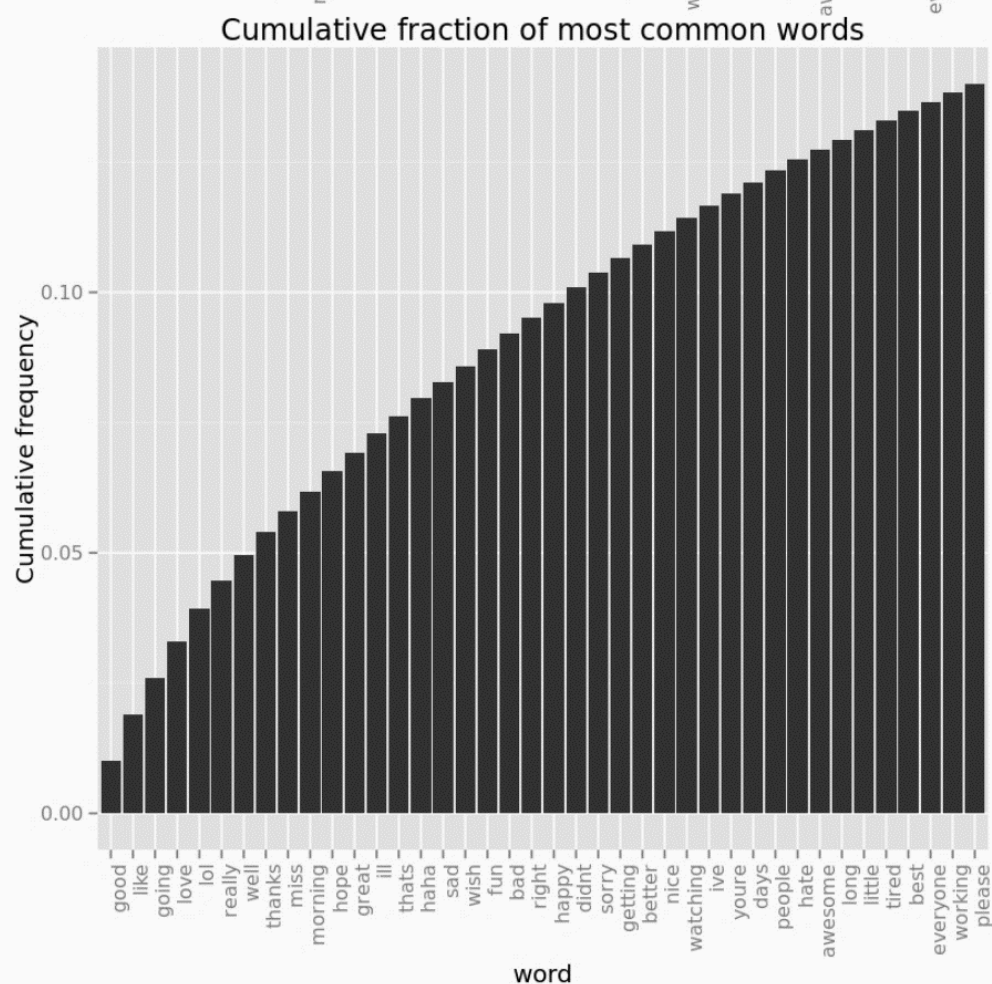
The **CDF** of the tweets with the stopwords removed is illustrated on the following page:



As before, this chart shows a number of frequent words which are likely to convey sentiment. However, note that these 40 most frequent words only make up about 15% or the total.

The head of the resulting word frequency data frame to determine is as follows:

- What is the percentage of all words for these first 20 words?
- Of these 20 words, how many are likely to contribute sentiment information?
- Are these 20 words different from the words seen for the normalized text?



	word	freq	Cum
good	good	9076	0.01019084
like	like	7767	0.01891188
going	going	6384	0.02608005
love	love	6217	0.03306071
lol	lol	5489	0.03922394

really	really	4859	0.04467979
well	well	4451	0.04967752
thanks	thanks	3899	0.05405545
miss	miss	3548	0.05803926
morning	morning	3386	0.06184118

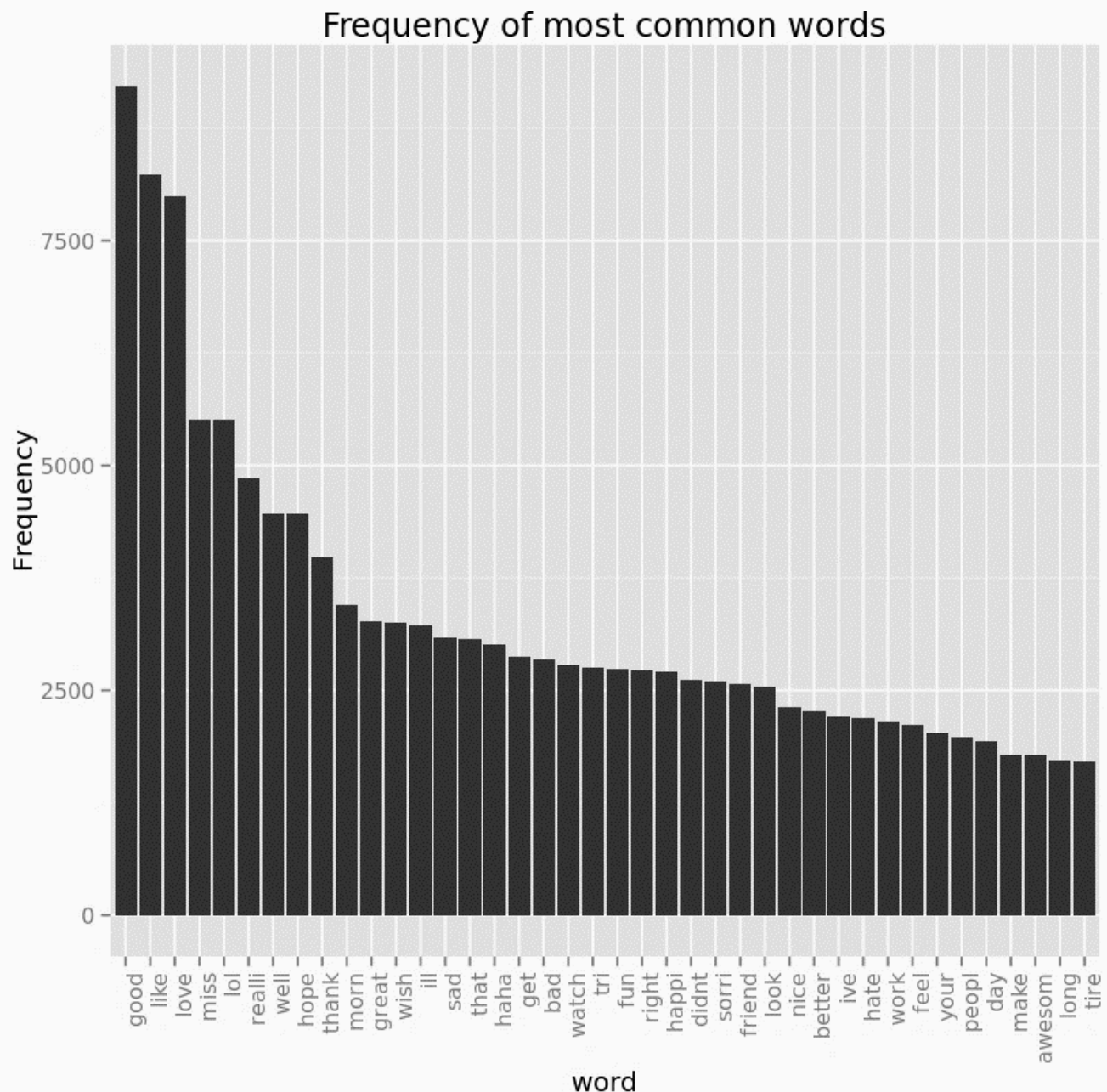
hope	hope	3383	0.06563972
great	great	3243	0.06928107
ill	ill	3207	0.072882
thats	thats	3066	0.07632461
haha	haha	3005	0.07969872

sad	sad	2800	0.08284266
wish	wish	2748	0.0859282
fun	fun	2734	0.08899803
bad	bad	2730	0.09206336
right	right	2699	0.09509389

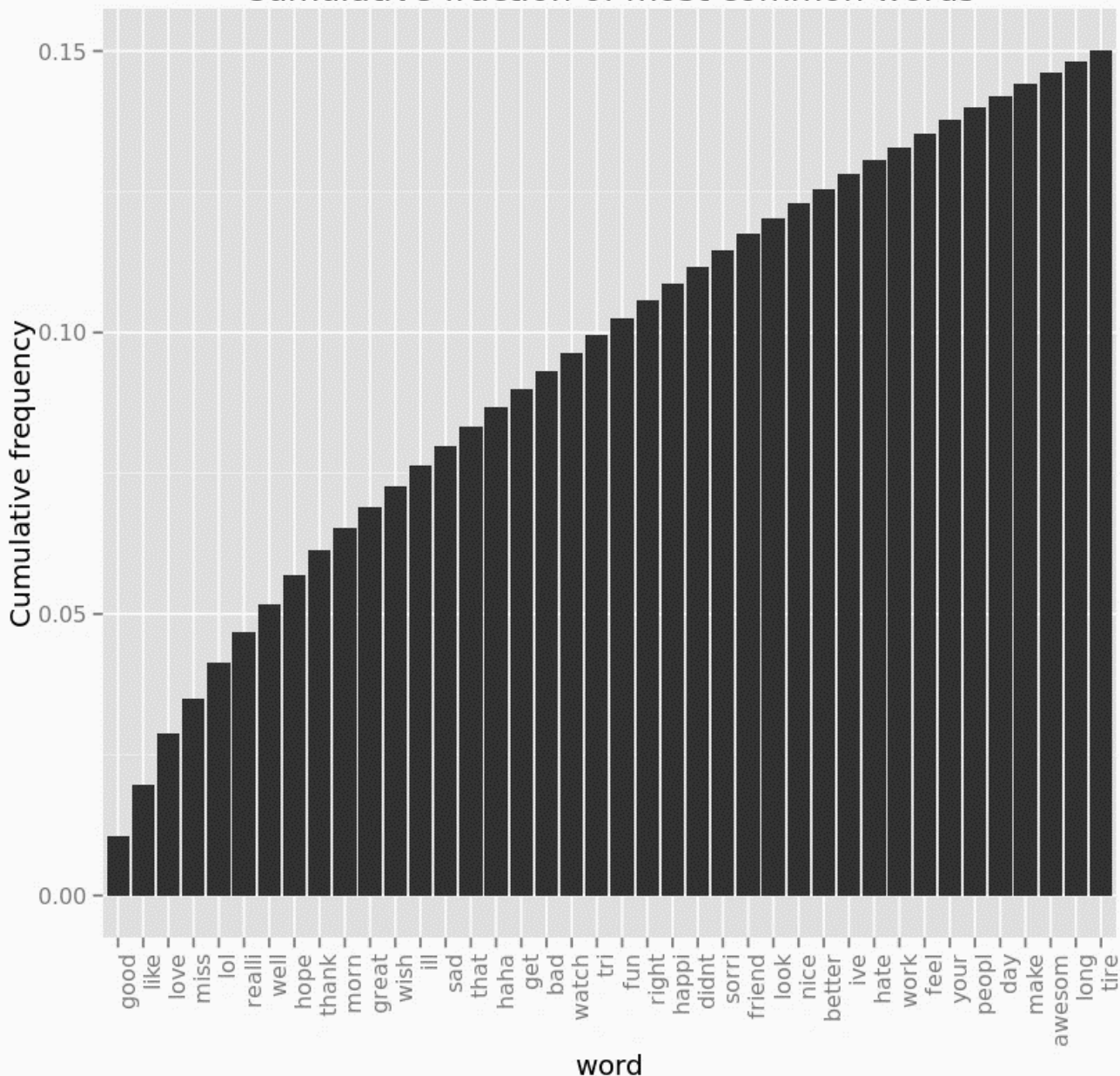
Stem the Words

The tweet text has been cleaned and stop words removed. There is one last data preparation step required, stemming the words. Stemming is a process of reducing words to their stems or roots. For example, conjugated verbs such as "goes", "going", and "gone" are stemmed to the word "go". Both Python and R offer a choice of stemmers. Depending on this choice, the results can be more or less suitable for the application. In this case, the popular Porter stemmer is applied.

The Porter stemmer is in the R **SnowBallC** library and the word frequency bar chart and **CDF** follow:



Cumulative fraction of most common words



	sentiment	text
tweets.content1	4	elephantbird dear happi already rice bowl
tweets.content2	4	ughhh layin downnnn wait zeina cook breakfast
tweets.content3	0	greeniebach reckon hell hes noth wont without
tweets.content4	0	valewe news
tweets.content5	0	sad httpwwwfabchannelcom close web servic ive use year
tweets.content6	0	fearnecotton sing rememb alwaysss radio catch artist

The two charts using the stemmed words to the charts created with just stop word filtering have notable differences. For example, some words like “good”, and “like” have moved higher in the order of most frequent words, while some other words like “going” have moved down.

	word	freq	Cum
good	good	9214	0.01042899
like	like	8241	0.01975667
love	love	7991	0.02880139
miss	miss	5512	0.03504022
lol	lol	5508	0.04127452
realli	realli	4859	0.04677425
well	well	4469	0.05183254
hope	hope	4464	0.05688518
thank	thank	3982	0.06139226
morn	morn	3451	0.06529832
great	great	3275	0.06900517
wish	wish	3258	0.07269278
ill	ill	3227	0.0763453
sad	sad	3079	0.07983031
that	that	3066	0.0833006
haha	haha	3005	0.08670185
get	get	2871	0.08995143
bad	bad	2848	0.09317498
watch	watch	2787	0.09632948
tri	tri	2757	0.09945003

Transformations Code (R)

```
dataset <- mam1.mapInputPort(1)

library(tm) ## Text mining library

## Set the column names
colnames(dataset) <- c("sentiment", "tweets")

## Extract text data and coerce the vector to a tm corpus
tweet.text <- Corpus(VectorSource(dataset['tweets']))

## Apply transformations to the corpus
tweet.text <- tm_map(tweet.text, content_transformer(removeNumbers))
tweet.text <- tm_map(tweet.text, content_transformer(removePunctuation))
tweet.text <- tm_map(tweet.text, content_transformer(stripWhitespace))
tweet.text <- tm_map(tweet.text, content_transformer(tolower))

## Transform the processed corpus back to a vector of
## character strings in a dataframe
tweet_content <- unlist(sapply(tweet.text, '[', "content"))
outframe <- data.frame(tweets = enc2utf8(tweet_content),
                      sentiment = dataset$sentiment / 2 - 1,
                      stringsAsFactors = F,
                      row.names = NULL)

## Output the result
mam1.mapOutputPort("outframe")
```

Stopwords Code (R)

```
dataset <- mam1.mapInputPort(1)
stop.words <- mam1.mapInputPort(2)

library(tm) ## Text mining library

## Extract text data and coerce the vector to a tm corpus
tweet.text <- Corpus(VectorSource(dataset['tweets']))

## Remove the stopwords
stop.words['words'] <- unique(stop.words['words'])
tweet.text <- tm_map(tweet.text, removeWords, stop.words[, 'words'])

## Transform the processed corpus back to a vector of
## character strings in a dataframe
dataset['tweets'] <- data.frame(text =
enc2utf8(unlist(sapply(tweet.text, `[`, "content"))),
                              stringsAsFactors = F)

## Output the result
mam1.mapOutputPort("dataset")
```

Stemming Code (R)

```
dataset <- mam1.mapInputPort(1)

library(tm) ## Text mining library
library(SnowballC) ## For stemming words

## Extract text data and create a tm corpus
tweet.text <- Corpus(VectorSource(dataset['tweets']))

## Stem the words in the tweets
tweet.text <- tm_map(tweet.text, stemDocument)

## Transform the processed corpus back to a vector of
## character strings in a dataframe
dataset['tweets'] <- data.frame(text =
enc2utf8(unlist(sapply(tweet.text, `[`, "content"))),
                              stringsAsFactors = F)

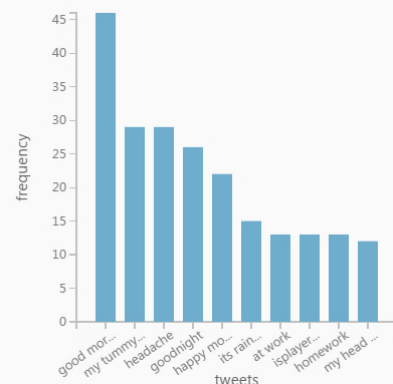
## Output the result
mam1.mapOutputPort("dataset")
```

Statistics

Unique Values	158761
Missing Values	0
Feature Type	String Feature

Visualizations

tweets
Histogram

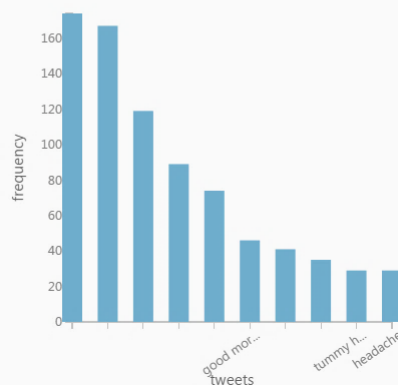


Statistics

Unique Values	157475
Missing Values	0
Feature Type	String Feature

Visualizations

tweets
Histogram

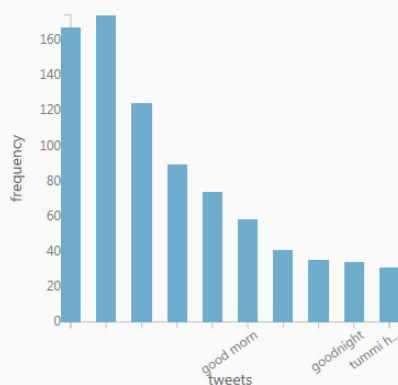


Statistics

Unique Values	157158
Missing Values	0
Feature Type	String Feature

Visualizations

tweets
Histogram



Feature Hashing

Target column(s)

Selected columns:

Column names: tweets

Launch column selector

Hashing bitsize

15

N-grams

2

START TIME 9/11/2016 12:56:25 PM

END TIME 9/11/2016 12:56:25 PM

ELAPSED TIME 0:00:00.000

STATUS CODE Finished

STATUS DETAILS Task output was present in output cache

Two-Class Logistic Regression

Create trainer mode

Parameter Range

Optimization tolerance

☐ Use Range Builder

0.0001, 0.0000001

L1 regularization weight

☐ Use Range Builder

0.0, 0.01, 0.1, 1.0

L2 regularization weight

☐ Use Range Builder

0.01, 0.1, 1.0

Memory size for L-BFGS

☐ Use Range Builder

5, 20, 50

Random number seed

1234

☒ Allow unknown categorical levels

START TIME 9/11/2016 12:56:24 PM

END TIME 9/11/2016 12:56:24 PM

ELAPSED TIME 0:00:00.000

STATUS CODE Finished

STATUS DETAILS Task output was present in output cache

Logistic Regression Classifier

Settings

Setting	Value
Optimization Tolerance	7.219577E-05
L1 Weight	0.9873694
L2 Weight	0.8192218
Memory Size	37
Quiet	True
Use Threads	True
Allow Unknown Levels	True
Random Number Seed	1234

DAT203.3x: Tweet Sentiment > Feature Hashing > Transformed dataset

rows

160000

columns

32770

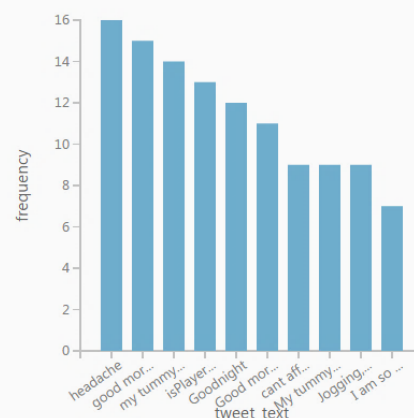
Statistics

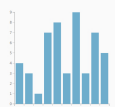
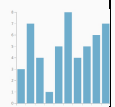
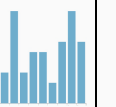
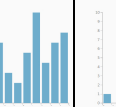
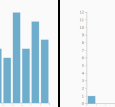
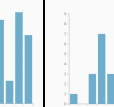
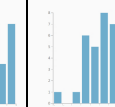
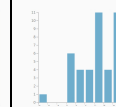
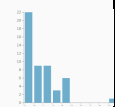
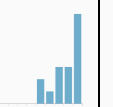

Unique Values	159384
Missing Values	0
Feature Type	String Feature

Visualizations

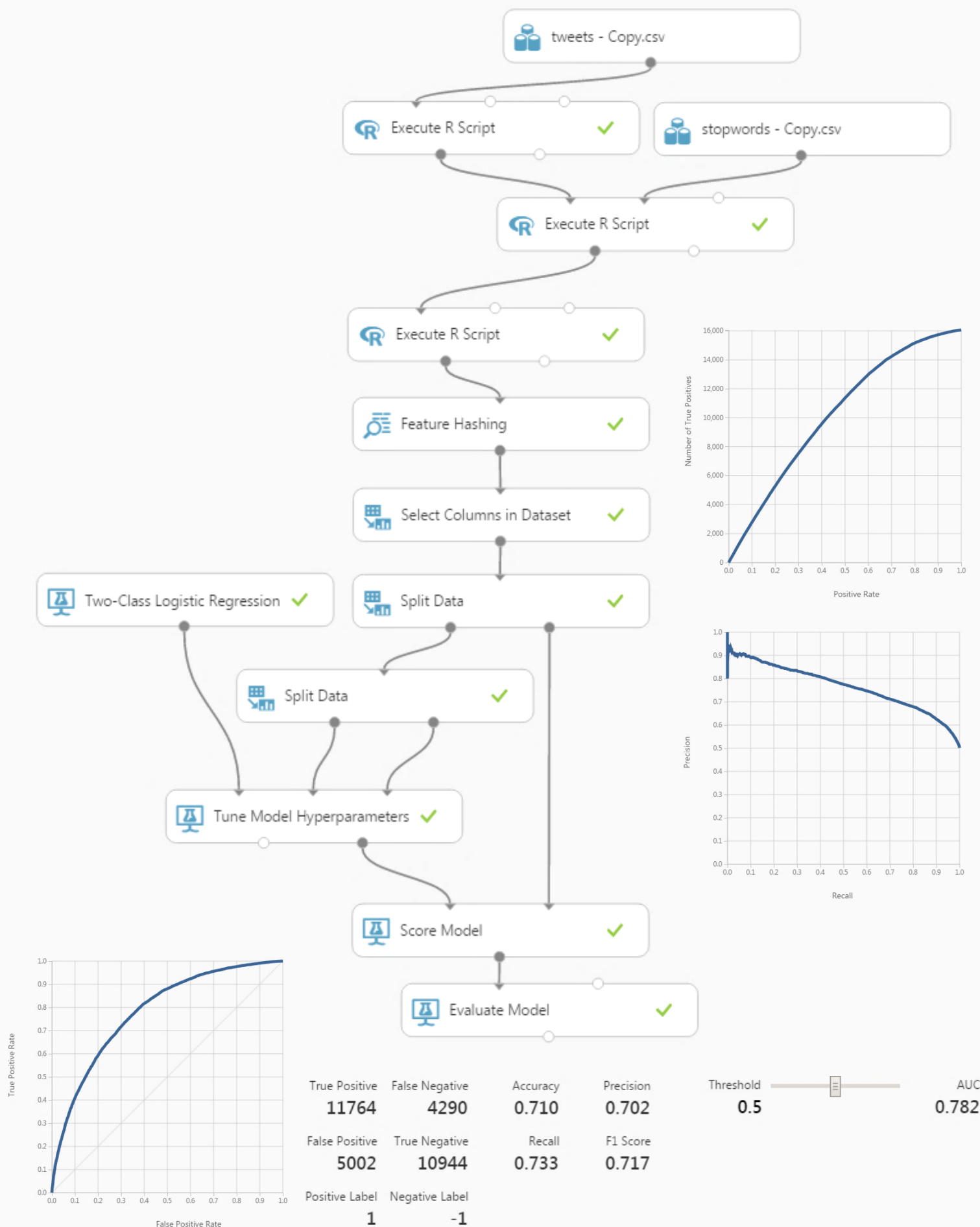
tweet_text

Histogram



Optimization Tolerance	L1Weight	L2Weight	Memory Size	Accuracy	Precision	Recall	F-Score	AUC	Average Log Loss	Training Log Loss
										
0.000072	0.987369	0.819222	37	0.715938	0.706338	0.737015	0.721351	0.786657	0.562243	18.885164
0.000062	0.965212	0.84226	47	0.715703	0.706289	0.736337	0.721	0.786473	0.562561	18.839362
0.000042	0.966165	0.837019	33	0.715417	0.705509	0.737328	0.721068	0.786168	0.562846	18.79819
0.000037	0.949337	0.744665	49	0.714688	0.70522	0.735554	0.720067	0.785144	0.565134	18.468124
0.000071	0.863326	0.680521	17	0.714219	0.705531	0.733152	0.719076	0.784801	0.56727	18.159966
0.000078	0.890538	0.592252	32	0.712708	0.704053	0.731691	0.717606	0.782702	0.570259	17.728712
0.000066	0.983824	0.427967	34	0.7125	0.702834	0.734092	0.718123	0.782088	0.572611	17.389419
0.00006	0.792988	0.656329	7	0.711224	0.702287	0.731064	0.716387	0.781737	0.572943	17.341528
0.000042	0.797965	0.74529	12	0.711016	0.7029	0.728768	0.7156	0.781616	0.57176	17.512114
0.000059	0.630235	0.81784	15	0.709818	0.702742	0.725009	0.713702	0.780277	0.575658	16.949781
0.000075	0.902023	0.278784	16	0.709271	0.701041	0.727463	0.714008	0.779992	0.577874	16.630165
0.000086	0.822914	0.557223	37	0.709219	0.700769	0.727985	0.714117	0.780054	0.576258	16.863311
0.000095	0.599584	0.961183	44	0.708255	0.701673	0.722295	0.711835	0.77894	0.57573	16.939368
0.000048	0.62759	0.886608	43	0.708021	0.701109	0.722921	0.711848	0.778503	0.577022	16.75308
0.000092	0.702919	0.693006	18	0.707552	0.700612	0.722556	0.711415	0.778491	0.578702	16.51068
0.000004	0.855496	0.300866	34	0.706875	0.699369	0.723391	0.711177	0.775861	0.587661	15.218137
0.000008	0.579902	0.899284	43	0.706823	0.700574	0.720102	0.710204	0.777338	0.579268	16.429014
0.000087	0.902168	0.234615	44	0.706667	0.699147	0.723234	0.710987	0.776459	0.587072	15.303101
0.000055	0.810852	0.296853	17	0.706224	0.699453	0.720885	0.710007	0.77638	0.587958	15.175238
0.000059	0.538525	0.913186	49	0.706146	0.700198	0.718693	0.709325	0.776486	0.581103	16.164302
0.000082	0.487715	0.852109	7	0.704766	0.699469	0.715717	0.7075	0.775048	0.584891	15.617813
0.000082	0.517855	0.801053	18	0.704271	0.698448	0.716605	0.70741	0.774875	0.586322	15.411283
0.000042	0.438004	0.766232	12	0.702083	0.696973	0.71269	0.704744	0.771902	0.594117	14.286758
0.000049	0.44319	0.803723	30	0.701771	0.697049	0.711385	0.704144	0.771883	0.59286	14.4681
0.000062	0.76727	0.171883	34	0.699792	0.693935	0.712481	0.703086	0.76969	0.60724	12.393441
0.000057	0.73465	0.144597	29	0.698828	0.692047	0.714047	0.702875	0.768911	0.611077	11.839867
0.000035	0.282475	0.931563	26	0.698776	0.694516	0.707313	0.700856	0.768482	0.600342	13.388639
0.000048	0.69215	0.219875	27	0.698568	0.693735	0.708618	0.701097	0.768936	0.610133	11.976103
0.000015	0.440053	0.578547	49	0.698255	0.694063	0.706635	0.700292	0.767434	0.607729	12.322923
0.000009	0.28278	0.904311	29	0.698255	0.694103	0.70653	0.700261	0.76795	0.601917	13.161364
0.000021	0.787116	0.05818	41	0.698099	0.692175	0.711072	0.701496	0.766425	0.620188	10.525497
0.000029	0.683195	0.220871	22	0.697812	0.692846	0.708253	0.700465	0.76784	0.611684	11.752335
0.000035	0.510715	0.362778	45	0.696927	0.692647	0.705591	0.699059	0.765129	0.619046	10.690283
0.000079	0.475943	0.400795	28	0.69651	0.692493	0.704494	0.698442	0.764749	0.619777	10.584724
0.000088	0.559042	0.24014	19	0.69599	0.691862	0.704286	0.698019	0.764667	0.626174	9.66191
0.000061	0.537679	0.236857	32	0.695286	0.689489	0.708096	0.698669	0.762416	0.63113	8.946835
0.000078	0.219613	0.793519	25	0.694193	0.690689	0.700893	0.695754	0.763573	0.616491	11.058797
0.000044	0.291094	0.528909	35	0.692917	0.689908	0.698335	0.694096	0.760212	0.632603	8.734293
0.00008	0.122658	0.808293	35	0.691901	0.689361	0.69609	0.692709	0.759991	0.627567	9.460836
0.000007	0.128374	0.749161	48	0.691458	0.68901	0.695412	0.692196	0.758809	0.632897	8.691962
0.000019	0.265951	0.419201	5	0.69125	0.687465	0.698805	0.693088	0.757015	0.651799	5.964985
0.000048	0.204696	0.574287	13	0.690964	0.688841	0.694054	0.691438	0.757935	0.642081	7.366973
0.000066	0.381252	0.186516	21	0.688906	0.686071	0.69395	0.689988	0.752958	0.677696	2.228742
0.000032	0.204689	0.471536	49	0.688516	0.685886	0.69301	0.68943	0.754481	0.657154	5.192418
0.000034	0.541512	0.02431	43	0.688151	0.684836	0.694524	0.689646	0.751224	0.688735	0.636169
0.000039	0.207929	0.386836	8	0.6875	0.685232	0.691027	0.688117	0.752809	0.667518	3.697091
0.000079	0.130017	0.371928	6	0.687109	0.687277	0.684084	0.685677	0.752501	0.674533	2.685072
0.000065	0.127815	0.443888	39	0.684818	0.682784	0.687738	0.685252	0.749667	0.680648	1.802955
0.000032	0.069577	0.497344	32	0.684036	0.682404	0.685859	0.684127	0.748955	0.683001	1.463398
0.000057	0.03185	0.21697	38	0.670521	0.669923	0.669364	0.669643	0.731614	0.819961	18.295771

Tweet Sentiment



Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	2006	260	0.071	0.553	0.219	0.885	0.125	0.528	0.984	0.001
(0.800,0.900]	2404	600	0.165	0.609	0.414	0.837	0.275	0.564	0.946	0.009
(0.700,0.800]	2520	903	0.272	0.660	0.560	0.797	0.432	0.609	0.889	0.029
(0.600,0.700]	2313	1241	0.383	0.693	0.653	0.755	0.576	0.655	0.812	0.069
(0.500,0.600]	2502	1982	0.523	0.710	0.716	0.702	0.732	0.718	0.687	0.150
(0.400,0.500]	1672	1823	0.632	0.705	0.740	0.663	0.836	0.776	0.573	0.240
(0.300,0.400]	1048	1874	0.723	0.679	0.738	0.625	0.901	0.820	0.455	0.343
(0.200,0.300]	776	2045	0.812	0.639	0.725	0.587	0.949	0.865	0.327	0.462
(0.100,0.200]	516	2505	0.906	0.577	0.700	0.544	0.981	0.901	0.170	0.614
(0.000,0.100]	297	2713	1.000	0.502	0.668	0.502	1.000	1.000	0.000	0.782

Tweet Sentiment [Predictive Exp.]

