Web Mining Final Project
Emotion Analysis from Tweets.

Tegjyot Singh Sethi
T0seth01@louisville.edu

Ranjith Kumar Nandella
Nandella.ranjithkumar@gmail.com

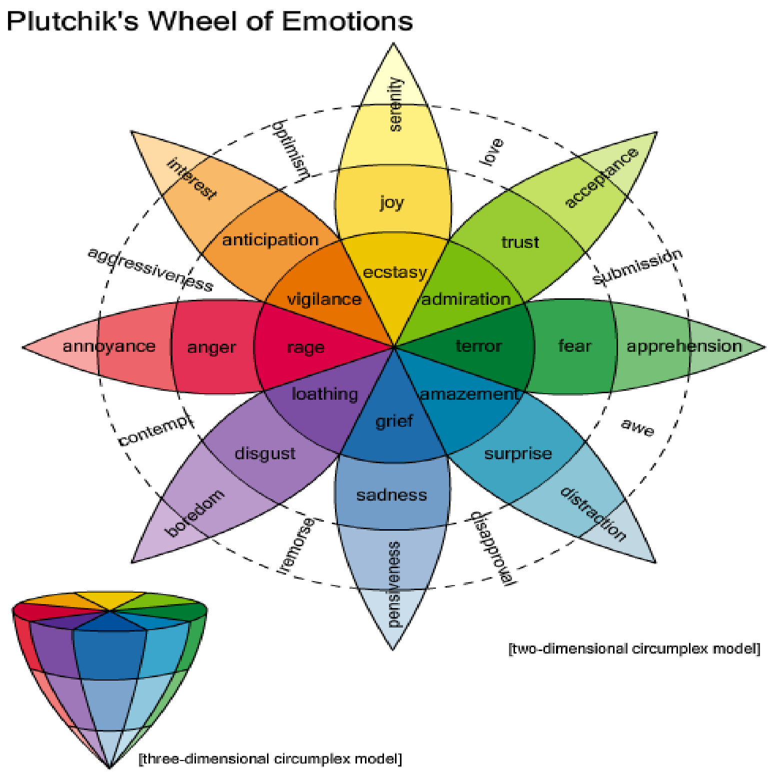
1. Introduction

Tweet Represent a wealth of real time data for analysis and Data Mining. One such analysis of tweets is to predict the emotion of the person who has written a particular tweet. Emotions are usually evident to the reader as either positive or negative. Also, the use of emoticons to reinforce one's emotions makes it evident that tweets are rarely neutral and factual, but instead they do represent a certain state of mind based on who is tweeting. Such information could be highly beneficial for Targeted marketing , Counseling and also for Recommendations based on the mood of a person.

In this project, a proof of concept is implemented which tries to demonstrate the 'Classifiability' of tweets into either Positive mood or Negative mood, based solely on the text in the tweet. Tweets are collected based on Hashtags denoting one of the states of emotions and they are then combined into one corpus with the queried Hashtags removed so that the model is not biased. Since, the tweets collected tend to be messy and not usable as is, preprocessing needs to be done on the tweets so that it is reduced to a form that is amenable to classification.

2. Motivation

Robert Plutchik's psychoevolutionary theory of emotion is one of the most influential classification approaches for general emotional responses. He considered there to be eight primary emotions - anger, fear, sadness, disgust, surprise, anticipation, trust, and joy. Plutchik proposed that these 'basic' emotions are biologically primitive and have evolved in order to increase the reproductive fitness of the animal. Robert Plutchik also created a wheel of emotions. This wheel is used to illustrate different emotions compelling and nuanced. He suggested 8 primary bipolar emotions: joy versus sadness; anger versus fear; trust versus disgust; and surprise versus anticipation. Additionally, his circumplex model makes connections between the idea of an emotion circle and a color wheel. Like colors, primary emotions can be expressed at different intensities and can mix with one another to form different emotions.



In this project we take two of the basic opposing emotions: Happiness and Grief and try to predict the tweets on these two emotions. The aim of the project is to ultimately extend the model to include all the 8 emotions and then derive auxiliary emotions from them.

3. Data Used

In order to perform the task, we need to collect tweets. Since twitter has an hourly download query limit of 150, the API needs to be used over a period of time to collect the tweets to be used for the Model generation task. The tweets were collected over a period of one week. The tweets were collected under two categories of either positive or negative and these are assumed to be specified by certain Hashtags as described below:

> ***Positive Emotions: #joy, #happy, # bliss , #ecstasy, #merry***
> ***Negative Emotions: #sad, #gloomy, # depressed, #mourn, #despair.***

The tweets are labeled as they are collected as either or positive or negative, based on the hashtags it contains. Once the tweets are collected and labeled these five hashtags are removed so that the data is not biased based on the hashtags which were used to collect the tweet.

The resulting tweets are sorted and duplicates are removed. The resulting files now have the following size:

Positive Tweets: 22088;Negative Tweets: 16175

This data is now passed to preprocessing to clean it up and make it ready for classification.

4. Background and Related Concepts

4.1. Vector Space Transformation

TF-IDF Weighting

The Term Frequency-Inverse Document Frequency (TF-IDF) transform is a vector space transform of model which is most widely used. Documents are also treated as a “bag” of words or terms. Each document is represented as a vector. However, the term weights are no longer 0 or 1. The TF_IDF scheme is used for weighting.

TF–IDFis the product of two statistics, term frequency and inverse document frequency. Various ways for determining the exact values of both statistics exist. Commonly used formula is normalized frequency, to prevent a bias towards longer documents, e.g. raw frequency divided by the maximum raw frequency of any term in the document:

$$\mathrm{tf}(t,d) = \frac{\mathrm{f}(t,d)}{\max\{\mathrm{f}(w,d) : w \in d\}}$$

The **inverse document frequency** is a measure of whether the term is common or rare across all documents. It is obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

$$\mathrm{idf}(t,D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

with $|D|$: cardinality of D, or the total number of documents in the corpus.
Then tf–idf is calculated as

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \times \mathrm{idf}(t,D)$$

A high weight in tf–idf is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms.

Before the TF-IDF term weighting scheme can be applied, the text needs to be preprocessed to remove terms that are similar but represented differently(Stemming) and also for removing commonly occurring words, that do not add to the value of the data mining task (Stop Word Removal).

Stemming

Stemming is the process for reducing inflected (or sometimes derived) words to theirstem, base or root form—generally a written word form. A stemmer for English, for example, should identify the string "cats" (and possibly "catlike", "catty" etc.) as based on the root "cat", and "stemmer", "stemming", "stemmed" as based on "stem". A stemming algorithm reduces the words "fishing", "fished", "fish", and "fisher" to the root word, "fish". Snowball is one of the popular stemmers used.

StopWord Removal

Stop words are words which are filtered out prior to, or after, processing of natural language data (text). Any group of words can be chosen as the stop words for a given purpose. For some search machines, these are some of the most common, short function words, such as *the*, *is*, *at*, *which*, and *on*.

4.2. Classification Techniques

Text classification is done based on finding the document given a word. It can be assumed as the probability of a word belonging to a document computed with respect to class prior probabilities.

Naïve Bayes Classifier

It is a classifier based on the Bayes theorem and it assumes the Independence among attribute values. In simple terms, a Naive Bayes classifier assumes that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class variable. For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting.

The pseudo code of the Naive Bayes classifier.

```
function train(   i) {
  Instances++
  if (++N[$Klass]==1) Klasses++
  for(i=1;i<=Attr;i++)
    if (i != Klass)
     if ($i !~ /\?/)
        symbol(i,$i,$Klass)
}
function symbol(col,value,klass) {
  Count[klass,col,value]++;
}
```

When testing, find the likelihood of each hypothetical class and return the one that is most likely.
Time and space complexity:The theoretical time complexity for learning a naive Bayes classifier is O(Np), where N is the number of training examples and p is the number of features. The theoretical space complexity for naive Bayes algorithm is O(pqr), where p is the num-ber of features, q is values for each feature, and r is alternative values for the class.

Decision Trees

The c4.5 and its implementation in Weka J48 is an extrapolation of the principles of Information Gain in data mining. *C4.5* is an algorithm used to generate a decision tree.

In pseudocode, the general algorithm for building decision trees is:

1. Check for base cases
2. For each attribute *a*
    1. Find the normalized information gain from splitting on *a*
3. Let *a_best* be the attribute with the highest normalized information gain
4. Create a decision *node* that splits on *a_best*
5. Recurse on the sublists obtained by splitting on *a_best*, and add those nodes as children of *node*

The time complexity of a decision tree algorithm is shown below:
Assume: m attributes,n training instances, tree depth O (log n), Building a tree takes : O (m n log n).
Total cost with subtree raising and pruning, Total cost: O (m n log n) + O (n (log n)^2.

Support Vector Machines

SVM are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, a SVM training algorithm builds a model that assigns new examples into one category or the other. A SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.In addition to performing linear classification, SVMs can efficiently perform non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

Constructing a SVM model often reduces to solving the optimization problem given below:

$$\text{Minimize: } \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} + C\sum_{i=1}^{r} \xi_i$$

$$\text{Subject to: } y_i(\langle \mathbf{w} \cdot \phi(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i, \quad i = 1, 2, \ldots, r$$
$$\xi_i \geq 0, \quad i = 1, 2, \ldots, r$$

The dual is

$$\text{Maximize: } L_D = \sum_{i=1}^{r} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{r} y_i y_j \alpha_i \alpha_j \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \rangle.$$

$$\text{Subject to: } \sum_{i=1}^{r} y_i \alpha_i = 0$$
$$0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, r.$$

The final decision rule for classification (testing) is

$$\sum_{i=1}^{r} y_i \alpha_i \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) \rangle + b$$

K-Nearest Neighbour

This algorithm is based on the observation that a sample that has features that are similar to the ones of points of one particular class it belongs to that class. These points are known as nearest neighbors. The parameter k specifies the number of neighbors (neighboring points) used to classify one particular sample point. Finally, the assignment of a sample to a particular class is done by having the k neighbors considered to "vote". In this fashion, the class represented by the largest number of points among the neighbors ought to be the class that the sample belongs to.

The KNN Algorithm's pseudo-code
Consider k as the desired number of nearest neighbors and $S: = p_1,...,p_n$ be the set of training samples in the form $p_1 = (x_i,c_i)$, where x_i is the d-dimensional feature vector of the point p_i and c_i is the class that p_i belongs to.

For each p'=(x',c')

- Compute the distance $d(x',x_i)$ between p' and all p_i belonging to S
- Sort all points p_i according to the key $d(x',x_i)$
- Select the first k points from the sorted list, those are the k closest training samples to p'
- Assign a class to p' based on majority vote: $c' = argmax_y \sum_{(x_i,c_i)}$ belonging to S, $I(y = c_i)$

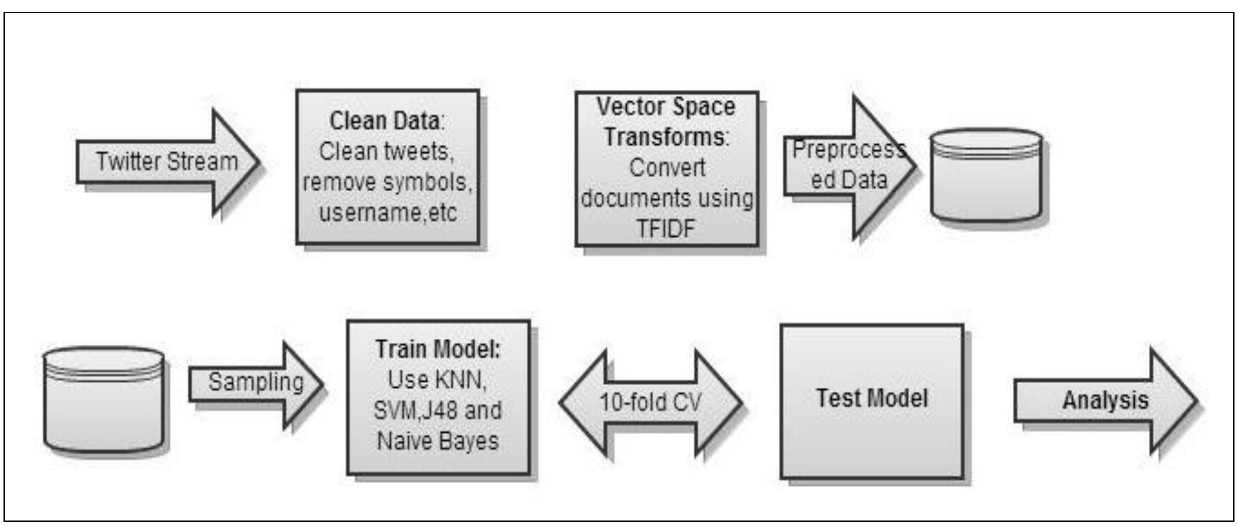
End.

Time and Space Complexity of KNN: Time Complexity of KNN is o(mkd) and space complexity of KNN is o(n*k*s) where n is number of samples, k is the nearest variable, d is the number of dimensions and s is the average space taken by each of the values.

5. Architecture of the System

The diagram below depicts the general architecture of the system.

Objective: Analyzing Tweets to classify them as either having a Positive or a Negative mood.



The Steps Involved in implementing the models are as follows:

Collecting the Data
Data from twitter streams needs to be collected. The Twitter API can be used to collect the tweets, but since the API has an hourly limit, this process should be spanned over a couple of days. The tweets with specific hashtags as described previously is obtained by using the search() function of the Twitter API.

Cleaning the Data
The tweets tend to have a lot of auxiliary information which can be discarded since our approach solely relies on analyzing the text in each of the tweets. The text itself could have a lot of unwanted information such as the username, URL's etc. These symbols could be removed, while other symbols relevant to us, eg. Emoticons, should be converted to a a different format amenable to classification.

Data Preprocessing
The Cleaned tweets are now ready for classification, However to classify based on text, we need to transform the tweets to their Vector space by using TFIDF transform. Additional cleansing can be provided by using Stopword removal and Stemming in this case.

Training the Model
The data after preprocessing is used for Training the appropriate classifier model. The model can either be trained on the entire data or a subset of the data based on whether we want to deliver the project or perform a lot of experiments randomly. The trained model is retrained and tested in several iterations till we get an appropriate measure of its performance by using the 10-fold crossvalidation metrics.

Test the model
The model after delivery can be tested by using new unknown tweets without their class labels.

System Performance Metrics

Since this is a classification task, the Performance metrics of interest are Precision, Accuracy, Recall, F-measure and AUC.These Measures are summarized below:

Measure	**Formula**	**Intuitive Meaning**
Precision	TP / (TP + FP)	The percentage of positive predictions that are correct.
Recall / Sensitivity	TP / (TP + FN)	The percentage of positive labeled instances that were predicted as positive.
Specificity	TN / (TN + FP)	The percentage of negative labeled instances that were predicted as negative.
Accuracy	(TP + TN) / (TP + TN + FP + FN)	The percentage of predictions that are correct.

F-Measure: is the harmonic mean of precision and recall

$$F1 = 2TP/(P + P') = 2TP/(2TP + FP + FN)$$

AUC: Area Under ROC curve.
ROC curve, is a graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold is varied. It is created by plotting the fraction of true positives out of the positives (TPR = true positive rate) vs. the fraction of false positives out of the negatives (FPR = false positive rate), at various threshold settings. When using normalized units, the area under the curve (AUC) is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one (assuming 'positive' ranks higher than 'negative')

Any clustering task in the process of analyzing the data can be measured in terms of two basic measures:

- InterCluster Similarity : should be low for good clusters.
- Intra Cluster Similarity : should be high for good clusters.
-

6. Preprocessing the data

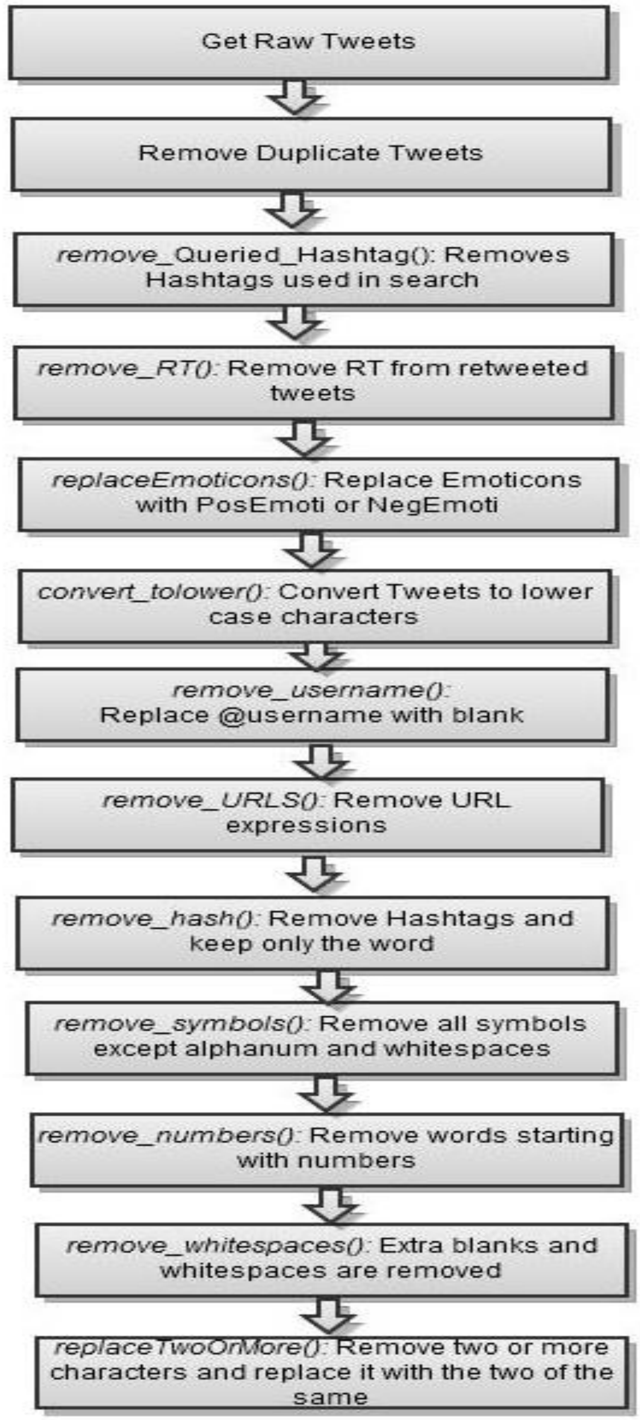
The tweets obtained from the above steps are collected using the API and are very messy they need to be cleaned and brought to a format that is suitable for classification or any other Data Mining technique.

The preprocessing in this part can be divided into two parts:

- Data Cleaning
- Vector Space Conversion

6.1. Data Cleaning

The tweets are cleaned initially so that unwanted and stray symbols are removed. The various steps involved in the data cleaning phase are depicted below:



The various steps in the above above figure are elaborated below:

Get_Tweets(): We get tweets by querying the Twitter API at constant intervals over a period of one week.

Remove_duplicate_tweets: The tweets obtained have a lot of repitions as the there may be no tweet on the same term for a period of time. Thus only unique tweets are kept and the rest are removed.

Remov_Queried _HashTags: The Hashtags used to search the Twitter API are removed from the tweets because they might cause a bias in the classification process as all the tweets obtained usin g the keyword will contain them

remove_RT(): Removing RT which is the Short form of Re tweet. It appears when a tweet is retweeted. It mostly appears at the start of the tweet.

replaceEmoticons(): Emoticons are symbols or signs used to describe the emotions. In the tweets these are replaced with PosEmoti or NegEmoti based upon the symbols defined. These descriptive text denote the kind of emoticon thus providing a more generalized frame for evaluation.

Descriptive Text	**Symbols**
PosEmoti	:), :-),:o), :],:3,:c),:D, C:, ;), :},:8
NegEmoti	:'(,;(,',D:, :{, :<, :-D, ', v.v, DX,D=,D;,D8,:C,:c , :-(, :(,
Heart	'<3'
BrokenHeart	'</3'

convert_tolower(): the tweets are converted to lowercase characters so that they are regarded the same based on the text alone.

remove_username(): Some tweets come along with a username which means they are referred to a person or thing, these are removed because they are used for further classifier. Username is usually specified as *@username,* these can be removed by specifying the regular expression for it.

remove_URLS(): URLs are mentioned to provide a link to a particular source or information. That does not have to describe the mood of the person or kind of tweet it is. So it removed in the next step.

remove_hash(): When a person wants to tell something in a single word he uses hash tag to highlight the importance of the word by inserting the symbol in front of the word. The word is necessary but not the hash tag so it is removed. Eg: #mad is replaced with mad.

remove_symbols_new(): Most of the symbols like {,.?/\<>*\(\)!_-} does not define the meaning of the statement so are removed during the preprocessing

remove_numbers(): Words which start with numbers are irrelevant and does not give much meaning to the sentence for example 2am,2morrow etc

remove_whitespaces(): The blanks are whitespaces in the tweets are removed to make word tokenization easy.

replaceTwoOrMore(): In tweets many a times characters are repeated althought it is not required so these repeated words are removed. Eg : happyyyyyyyyyyyyy is replaced with happy.

Sample Tweet:
Raw:

How to Avoid the and #Discouragement of Long Term #JobLoss. http://t.co/1RuLoLPg62 #Depression #Networking #HiddenJobMarket
How to deal with #pessimism and even in the midst of hardship, with @carter_phipps: http://t.co/RrRfpmCwhA @hunterr_hancock @hannahkshumate #coldshoulder #ignore #sadness #depression #bacon #lubricant #yellowpages #brush #randomhashtags
I advised my teenage cousin to checkout the #GWU podcast from @RealJudgeJules. His reply, "I'm an indie rock kinda guy".
I can't find my Star Wars T-Shirt... @sonofsammie ! #despondency

After Preprocessing:

how to avoid the and discouragement of long term jobloss depression networking hiddenjobmarket
how to deal with pessimism and even in the midst of hardship with
coldshoulder ignore sadness depression bacon lubricant yellowpages brush randomhashtags
i advised my teenage cousin to checkout the gwu podcast from his reply im an indie rock kinda guy
i cant find my star wars t shirt despondency

6.2. Vector Space Transform

The preprocessed tweets are converted to vector space format to able to work on them. The Vector space transform involves the following transfromations:

TF-IDF transform

Term frequency (tf) measures a word's relevancy in a single text. Document frequency (df) measures a word's overall relevancy across documents. Dividing tf by df yields tf-idf, a simple and elegant measurement of a word's uniqueness in a text when compared to other texts. The documents are converted to sparse numeric vectors after applying this filter.

Stop Words

Stop words are words that are so common (e.g. *each*, *his*,*very*) that they are ignored . The Stop words is set to true to remove them from the corpus.

Stemming

Normalizing the words to their base terms to to normalize words across their varied formats The PORTER stemnming algorithm is used eg. *consisted* and *consistently* are stemmed to *consist*.

Specify Wordcount

The wordcount is specified so that only a specific number of top occurring terms are considered in the features vector. Thus managing dimensionality. I tis typically set to 300,500,1000.

7. Experimentation and Analysis

The preprocessed data from the previous phase of the project is used here to Analyze the data and develop classification models on top of it. In this section, the Data is analyzed using the CLUTO's clustering tool and then classified with varied parameters to gain a better understanding of the model and the data.

Clustering provides insight into the spread and distribution of the data and hence can be used to see the validity of classifying such a dataset. The dataset is then sampled and experimentation is done on the subset of the dataset (10% and 20%) this enables us to perform our experiments efficiently and also be able to understand the behavior of various techniques on the given data. Once, the method of choice is established, the entire dataset is used and the model is generated on it. To facilitate understandability of the data, a decision tree model is also generated as the rules generated are intuitive to follow. Also, the experiments is repeated by decreasing the number of features and seeing the impact on the results. Principal Component Analysis is also explored as a feature reduction technique and the results obtained are compared with the other values.

Data After Preprocessing

Negative Tweets: 15156
Positive Tweets: 15042
Instances: 30198
Attributes: 730

7.1. Data Exploration with CLUTO

In the initial phases of the analysis, the data is clustered using CLUTO, to find clusters of words. These similar words are then analyzed to see if they point to any specific majority class. This enables us to see if there is any inherent patterns in using a word and the class it belongs to.

The clustering is performed with the following parameters:

Method: Repeated Bisection
Criterion Function : I2
#Iterations: 10

The clusters obtained and the report generated is shown below:

Clustering Options

Method: Repeated Bisection **#Clusters:** 10
CRfun: I2 **Simfun:** Cosine
RowModel: None **ColModel:** None **Graph Model:** Asymetric-Direct
ColPrune: 1.000 **EdgePrune:** 0.000 **VertexPrune:** 0.000
Nearest Nieghbors: 4 **MinComponent:** 1 **CSType:** Best
#Trials: 10 **#Iterations:** 10

10-way clustering: [37748 of 37748]

Cluster	Size	ISim	ISdev	ESim	ESdev
0	2077	0.151	0.071	0.008	0.005
1	2328	0.147	0.088	0.008	0.005
2	1704	0.137	0.079	0.006	0.004
3	3592	0.064	0.040	0.007	0.004
4	3095	0.047	0.029	0.005	0.003
5	3600	0.046	0.030	0.006	0.004
6	3015	0.031	0.021	0.005	0.003
7	5152	0.024	0.016	0.005	0.003
8	5617	0.018	0.024	0.003	0.005
9	7568	0.008	0.006	0.003	0.002

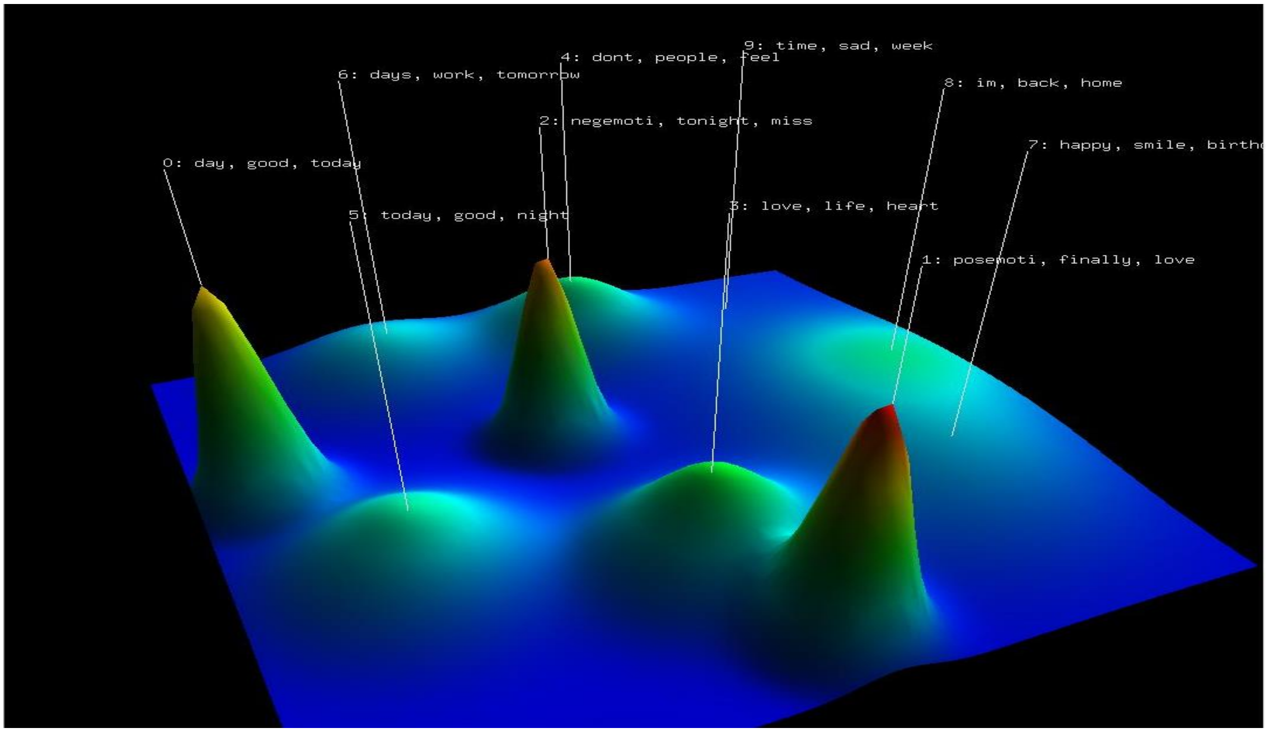
Go to Top

Descriptive & Descriminating Features

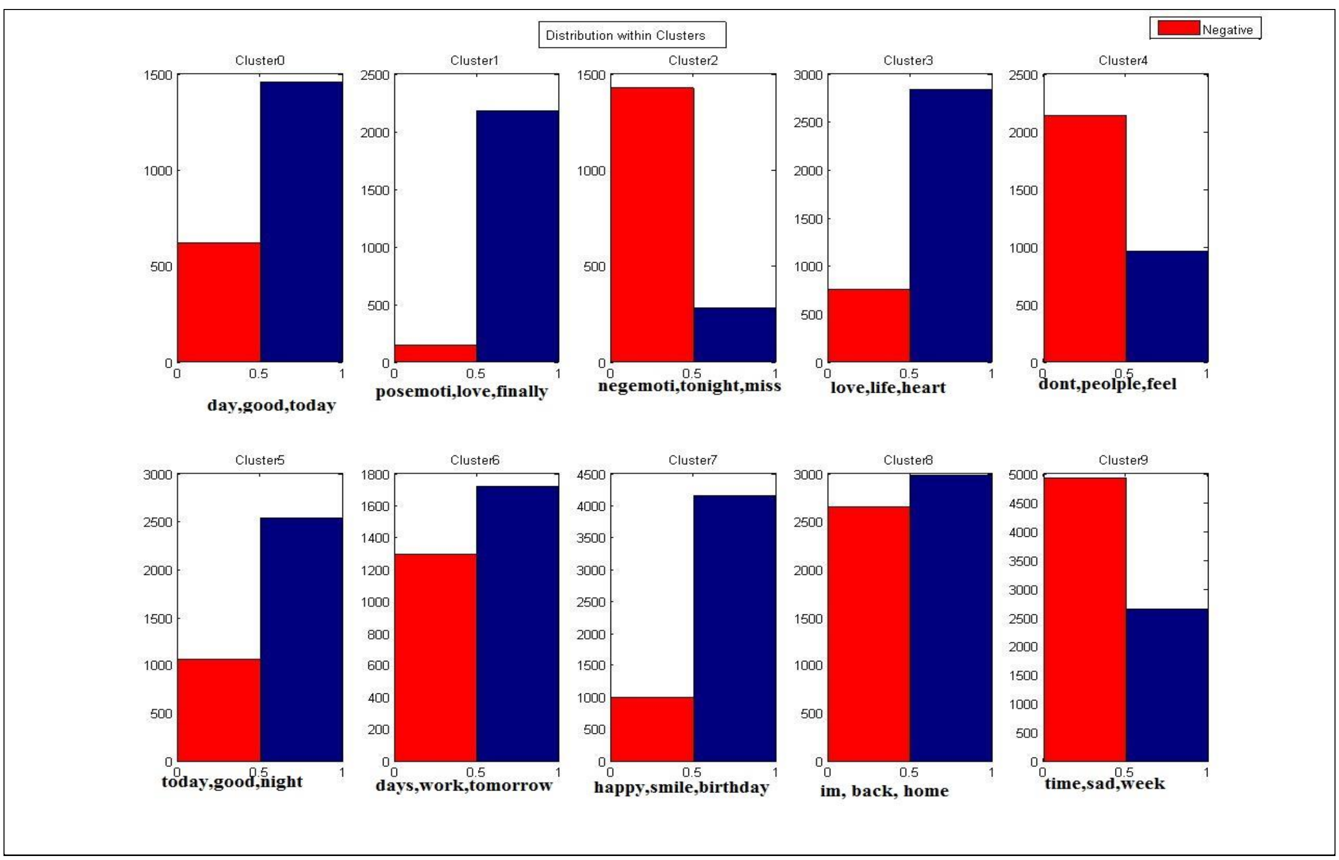
Cluster 0 Size: 2077 **ISim:** 0.151 **ESim:** 0.008								
Descriptive:	day	92.3%	good	1.0%	today	0.8%	posemoti	0.7%
Descriminating:	day	57.9%	love	3.9%	im	2.9%	posemoti	2.6%
Cluster 1 Size: 2328 **ISim:** 0.147 **ESim:** 0.008								
Descriptive:	posemoti	92.1%	finally	3.0%	love	0.4%	thursday	0.3%
Descriminating:	posemoti	54.9%	day	4.6%	love	3.1%	negemoti	2.4%
Cluster 2 Size: 1704 **ISim:** 0.137 **ESim:** 0.006								
Descriptive:	negemoti	89.4%	tonight	5.5%	miss	0.5%	omg	0.4%
Descriminating:	negemoti	53.6%	posemoti	4.7%	love	4.1%	day	3.1%
Cluster 3 Size: 3592 **ISim:** 0.064 **ESim:** 0.007								
Descriptive:	love	66.4%	life	17.7%	heart	3.7%	happiness	1.5%
Descriminating:	love	38.7%	life	9.2%	posemoti	6.3%	day	5.0%
Cluster 4 Size: 3095 **ISim:** 0.047 **ESim:** 0.005								
Descriptive:	dont	38.5%	people	22.6%	feel	13.5%	lol	6.7%
Descriminating:	dont	23.2%	people	13.5%	feel	7.5%	posemoti	5.8%
Cluster 5 Size: 3600 **ISim:** 0.046 **ESim:** 0.006								
Descriptive:	today	38.4%	good	20.9%	night	10.6%	bed	5.7%
Descriminating:	today	22.2%	good	10.5%	love	6.0%	posemoti	6.0%
Cluster 6 Size: 3015 **ISim:** 0.031 **ESim:** 0.005								
Descriptive:	days	27.2%	work	18.7%	tomorrow	13.1%	school	5.7%
Descriminating:	days	17.6%	work	10.8%	tomorrow	7.6%	posemoti	5.8%
Cluster 7 Size: 5152 **ISim:** 0.024 **ESim:** 0.005								
Descriptive:	happy	14.3%	smile	13.2%	birthday	11.6%	girl	8.8%
Descriminating:	smile	7.1%	birthday	6.8%	happy	6.7%	posemoti	6.5%
Cluster 8 Size: 5617 **ISim:** 0.018 **ESim:** 0.003								
Descriptive:	im	74.5%	back	8.4%	home	8.4%	gonna	0.9%
Descriminating:	im	45.9%	posemoti	5.9%	love	4.7%	day	4.5%
Cluster 9 Size: 7568 **ISim:** 0.008 **ESim:** 0.003								
Descriptive:	time	13.6%	sad	5.2%	week	4.0%	miss	3.3%
Descriminating:	time	7.6%	posemoti	7.0%	love	6.8%	day	5.3%

Go to Top

As can be seen from the Report, the Similarity of the Cluster 0 and I is high inside the cluster. The mountain view below depicts clusters and the features describing each cluster.



The class distribution from each of the clusters is as shown below:



From the distribution of the data above, the relevance of differnet words to the clusters is obtained , as shown below.

Positive Clusters: day,good,today, posemoti, finally,love,life,heart, today, happy, smile, birthday.
Negative Cluster: negemoti, tonight, miss, don't, people, feel, time, sad, week.
*Neutral Cluste*r: days,work,tomorrow, im,back,home.

These are the words that are most effective in defining these clusters. It is also interesting to see that Love, good and today are the most effective features for the Positive class while words like sad, miss,etc are used to denote negative emotions. Also, In clusters with no significant majority class, the words are common and bear no strong emotional bearing: days, work, tomorrow, back,etc. The cluster also denote those words which appear closely together in sentences eg: Love, Life, Heart.

We also obtained two distinct clusters one in which we have all the negative emoticons and the other with all the positive emoticons. Thus the corpus has certain features that can be used inorder to classify the tweets based on the moods. It is also worth noting that the classification accuracy might not be high as we have large number of tweets clustered in clusters with no significant majority class.

7.2. Classification on Subset of the Dataset

Since the dataset generated is large and has very high dimensionality, it is intractable and takes a long time to run different algorithms on it. Thus to be able to perform rapid experimentation on the data, a subset of it is taken and experiments are performed on it so that we can choose the method that best fits the data.

Experiments with 10% of the samples

The dataset is sampled and 10% of the data is taken. The resulting dataset has 3774 data samples and 169 attributes. The data is converted to Vector Space Representation and then it is classified as below:

Decision Trees (J48) Classifier

The first classifier used is the Decision tree classifier which although not very efficient for text data, can generate visual rules which are easy to interpret. The parameters of the algorithm chosen were:

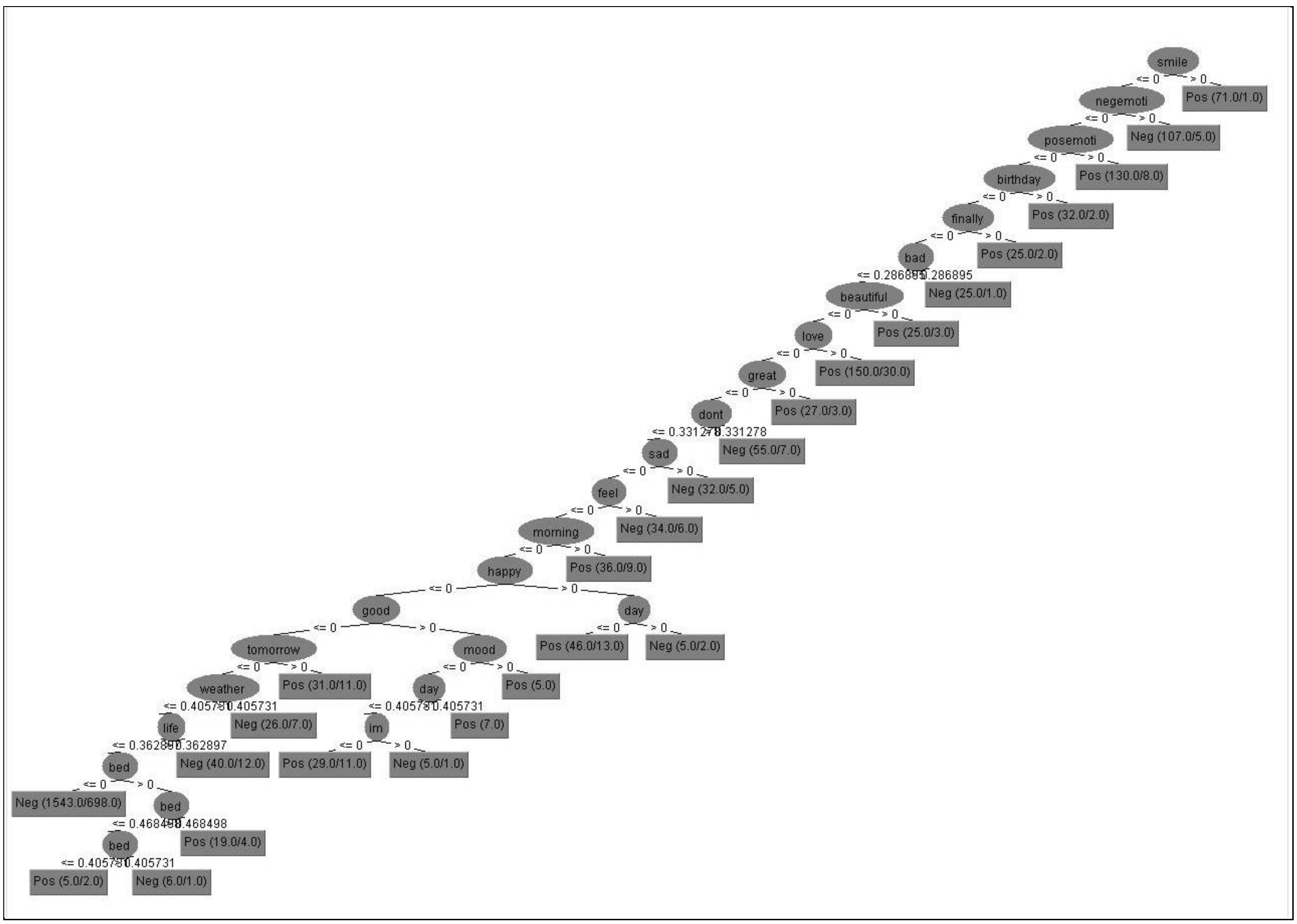
Confidence level= 0.1
Min number of objects in split=5

10 fold crossvalidation was performed and the results are as shown below:

```
=== Summary ===

Correctly Classified Instances       2405             63.7255 %
Incorrectly Classified Instances     1369             36.2745 %
Kappa statistic                         0.2821
Mean absolute error                     0.4092
```

```
Root mean squared error              0.456
Relative absolute error            81.8839 %
Root relative squared error          91.2104 %
Total Number of Instances           3774

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
0.901     0.616     0.584     0.901     0.709     0.704   Neg
0.384     0.099     0.802     0.384     0.519     0.704   Pos
Weighted Avg.   0.637     0.352     0.695     0.637     0.612     0.704

=== Confusion Matrix ===

a    b   <-- classified as
1665  183 |    a = Neg
1186  740 |    b = Pos
```

The accuracy as predicted is not very high but the tree as shown below, enables us to get some insights as to what features are being considered in the classification process.



```
smile <= 0
|  negemoti <= 0
|  |  posemoti <= 0
|  |  |  birthday <= 0
|  |  |  |  finally <= 0
|  |  |  |  |  bad <= 0.286895
|  |  |  |  |  |  beautiful <= 0
|  |  |  |  |  |  |  love <= 0
|  |  |  |  |  |  |  |  great <= 0
|  |  |  |  |  |  |  |  |  dont <= 0.331278
|  |  |  |  |  |  |  |  |  |  sad <= 0
|  |  |  |  |  |  |  |  |  |  |  feel <= 0
|  |  |  |  |  |  |  |  |  |  |  |  morning <= 0
|  |  |  |  |  |  |  |  |  |  |  |  |  happy <= 0
|  |  |  |  |  |  |  |  |  |  |  |  |  |  good <= 0
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  tomorrow <= 0
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  weather <= 0.405731
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  life <= 0.362897
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  bed <= 0: Neg (1543.0/698.0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  bed > 0
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  bed <= 0.468498
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  bed <= 0.405731: Pos (5.0/2.0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  bed > 0.405731: Neg (6.0/1.0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  bed > 0.468498: Pos (19.0/4.0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  life > 0.362897: Neg (40.0/12.0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  weather > 0.405731: Neg (26.0/7.0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  tomorrow > 0: Pos (31.0/11.0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  good > 0
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  mood <= 0
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  day <= 0.405731
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  im <= 0: Pos (29.0/11.0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  im > 0: Neg (5.0/1.0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  day > 0.405731: Pos (7.0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  mood > 0: Pos (5.0)
|  |  |  |  |  |  |  |  |  |  |  |  |  happy > 0
|  |  |  |  |  |  |  |  |  |  |  |  |  |  day <= 0: Pos (46.0/13.0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  day > 0: Neg (5.0/2.0)
|  |  |  |  |  |  |  |  |  |  |  |  morning > 0: Pos (36.0/9.0)
|  |  |  |  |  |  |  |  |  |  |  feel > 0: Neg (34.0/6.0)
|  |  |  |  |  |  |  |  |  |  sad > 0: Neg (32.0/5.0)
|  |  |  |  |  |  |  |  |  dont > 0.331278: Neg (55.0/7.0)
|  |  |  |  |  |  |  |  great > 0: Pos (27.0/3.0)
|  |  |  |  |  |  |  love > 0: Pos (150.0/30.0)
|  |  |  |  |  |  beautiful > 0: Pos (25.0/3.0)
|  |  |  |  |  bad > 0.286895: Neg (25.0/1.0)
|  |  |  |  finally > 0: Pos (25.0/2.0)
|  |  |  birthday > 0: Pos (32.0/2.0)
|  |  posemoti > 0: Pos (130.0/8.0)
|  negemoti > 0: Neg (107.0/5.0)
smile > 0: Pos (71.0/1.0)
```

From the tree above it can be seen that words such as love, life, sad, feel,etc have a lot of influence on the classification task. This was predicted from the cluster analysis performed previously.

Naïve Bayes Classifier

The next classifier used was the standard Naïve Bayes classifier, which should work better with text data as it is based on count on each of the attributes and hence resionates well with TFIDF format of the documents.

The results from the 10-fold crossvalidation are presented below:

```
=== Summary ===
Correctly Classified Instances       2539               67.2761 %
Incorrectly Classified Instances     1235               32.7239 %
Kappa statistic                      0.3507
Mean absolute error                  0.3274
Root mean squared error              0.5333
Relative absolute error             65.5061 %
Root relative squared error        106.6795 %
Total Number of Instances            3774
=== Detailed Accuracy By Class ===
               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.872     0.519     0.617     0.872     0.723      0.76     Neg
                 0.481     0.128     0.797     0.481     0.6        0.758    Pos
Weighted Avg.    0.673     0.319     0.709     0.673     0.66       0.759
=== Confusion Matrix ===
    a    b   <-- classified as
 1612  236 |    a = Neg
  999  927 |    b = Pos
```

The same experiment was performed after applying the *Principal Component Transform* on the dataset. The modified vectors were used in the classification and the results are as follows:

```
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances       2558               67.7795 %
Incorrectly Classified Instances     1216               32.2205 %
Kappa statistic                      0.3521
Mean absolute error                  0.3289
Root mean squared error              0.4991
Relative absolute error             65.8054 %
Root relative squared error         99.8343 %
Total Number of Instances            3774
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.548     0.197     0.727     0.548     0.625      0.768    Neg
                 0.803     0.452     0.649     0.803     0.718      0.768    Pos
Weighted Avg.    0.678     0.327     0.687     0.678     0.672      0.768
=== Confusion Matrix ===

    a    b   <-- classified as
 1012  836 |    a = Neg
  380 1546 |    b = Pos
```

As can be seen the accuracy is not affected much. However the time to execute greatly improves.

Support Vector Machines

Support Vector Machines are known to perform well on high dimensional data and hence are suitable for this dataset as well. The parameters are as given below:

C=1.0
Epsilon:1.0E-12
Kernel: PolyKernel(The polynomial kernel : K(x, y) = <x, y>^p or K(x, y) = (<x, y>+1)^p)
toleranceParameter=0.001

The result from 10-fold cross-validation are presented below:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       2773               73.4764 %
Incorrectly Classified Instances     1001               26.5236 %
Kappa statistic                         0.4716
Mean absolute error                     0.2652
Root mean squared error                 0.515
Relative absolute error                53.0698 %
Root relative squared error           103.0241 %
Total Number of Instances            3774

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.837     0.363      0.688     0.837     0.756      0.737    Neg
                 0.637     0.163      0.803     0.637     0.71       0.737    Pos
Weighted Avg.    0.735     0.261      0.747     0.735     0.732      0.737

=== Confusion Matrix ===
    a    b   <-- classified as
 1547  301 |    a = Neg
  700 1226 |    b = Pos
```

7.3. Experiments with 20% of the Sample data

Next we increase our dataset with 20% of the data and we continue our understanding and exploration of the data.

K-Means Clustering

We cluster the data using Kmeans cluster with k=8 to see if the data has any significant clusters. The results are as shown below:

```
Within cluster sum of squared errors: 10333.782608482701
Clustered Instances
0       14 (  0%)
1      163 (  2%)
2       13 (  0%)
3        2 (  0%)
4       47 (  1%)
5     7308 ( 97%)
6        6 (  0%)
7        1 (  0%)

Class attribute: Class
Classes to Clusters:
   0    1    2    3    4    5    6    7  <-- assigned to cluster
   3   55    1    1   13 3715    1    1 | Neg
  11  108   12    1   34 3593    5    0 | Pos
Cluster 0 <-- No class
Cluster 1 <-- Pos
Cluster 2 <-- No class
Cluster 3 <-- No class
Cluster 4 <-- No class
Cluster 5 <-- Neg
Cluster 6 <-- No class
Cluster 7 <-- No class
Incorrectly clustered instances :    3731.0    49.3911 %
```

Most of the clusters obtained have no majority class but two of them are distinctly classified as Pos and Neg thus we can proceed with classification of the data.

```
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances        5751               76.1319 %
Incorrectly Classified Instances      1803               23.8681 %
Kappa statistic                          0.5226
Mean absolute error                      0.2387
Root mean squared error                  0.4886
Relative absolute error                 47.7369 %
Root relative squared error             97.7107 %
Total Number of Instances             7554
=== Detailed Accuracy By Class ===
               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.775     0.253      0.755     0.775     0.765      0.761    Neg
                 0.747     0.225      0.768     0.747     0.757      0.761    Pos
Weighted Avg.    0.761     0.239      0.761     0.761     0.761      0.761
=== Confusion Matrix ===
    a    b   <-- classified as
 2938  852 |    a = Neg
  951 2813 |    b = Pos
```

As we can see the classification accuracy is ~75% which is still not unacceptable. Thus we pnow proceed with experimentation with the full dataset.

7.4. Experimenting with the Full Dataset

The experiments is performed with the full preprocessed dataset as is and then subsequent resampling and feature reduction steps are performed to it to obtain meaningful results.

Experiments with the full dataset

The full dataset has 37748 samples and 728 attributes. The 10-fold cross-validation results are shown below:

```
Correctly Classified Instances      29692          78.6585 %
Incorrectly Classified Instances     8056          21.3415 %
Kappa statistic                      0.5531
Mean absolute error                  0.2134
Root mean squared error              0.462
Relative absolute error             43.6811 %
Root relative squared error         93.4678 %
Total Number of Instances           37748

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.663     0.122     0.8       0.663     0.725      0.77     Neg
                 0.878     0.337     0.779     0.878     0.826      0.77     Pos
Weighted Avg.    0.787     0.246     0.788     0.787     0.783      0.77

=== Confusion Matrix ===

     a     b   <-- classified as
 10620  5401 |     a = Neg
  2655 19072 |     b = Pos
```

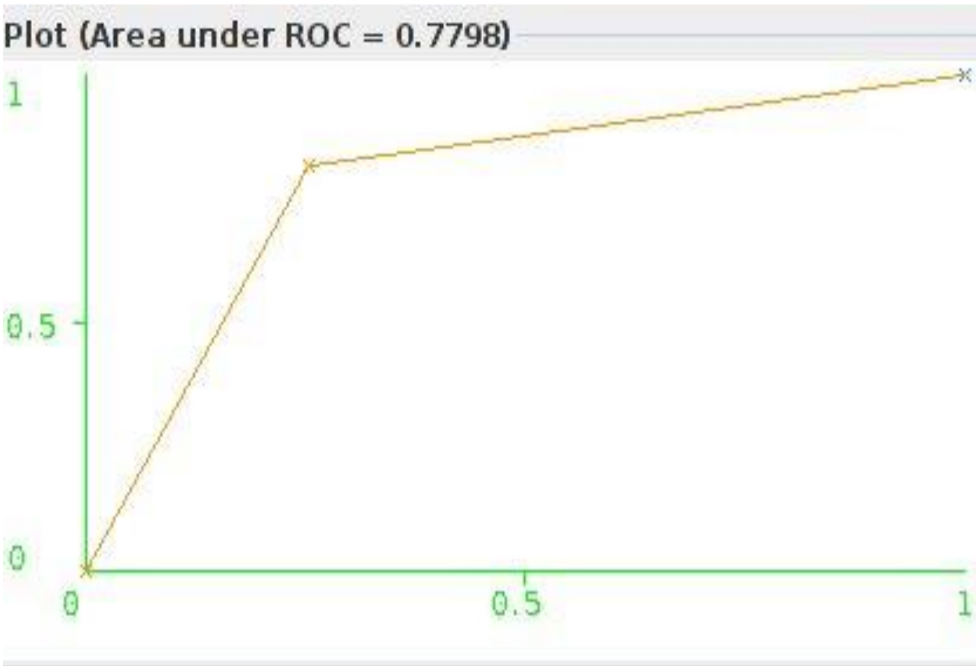
Balanced dataset

Next we perform the experiments with balanced data which is performed by using the resample filter in Weka. The class distribution is now uniform and hence the accuracy is more significant performance measure now.

```
Correctly Classified Instances      29822          79.0029 %
Incorrectly Classified Instances     7926          20.9971 %
Kappa statistic                      0.58
Mean absolute error                  0.21
Root mean squared error              0.4582
Relative absolute error             41.9945 %
Root relative squared error         91.6455 %
Total Number of Instances           37748

=== Detailed Accuracy By Class ===
```

```
            TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
              0.823     0.243      0.773     0.823     0.797      0.79     Neg
              0.757     0.177      0.809     0.757     0.783      0.79     Pos
Weighted Avg.    0.79      0.21       0.791     0.79      0.79       0.79

=== Confusion Matrix ===

     a     b   <-- classified as
 15564  3356 |     a = Neg
  4570 14258 |     b = Pos
```

The Roc Cure obtained for the above classifier is shown below:



The same experiment is performed with Naïve Bayes also, however, the accuracy obtained is not high.

```
Correctly Classified Instances       18964              62.7989 %
Incorrectly Classified Instances     11234              37.2011 %
Kappa statistic                          0.2577
Mean absolute error                      0.372
Root mean squared error                  0.6085
Relative absolute error                 74.4068 %
Root relative squared error            121.695  %
Coverage of cases (0.95 level)          63.3155 %
Mean rel. region size (0.95 level)      50.5315 %
Total Number of Instances            30198

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.319    0.061    0.841      0.319    0.463      0.329    0.781     0.759     Neg
                 0.939    0.681    0.578      0.939    0.715      0.329    0.779     0.750     Pos
Weighted Avg.    0.628    0.370    0.710      0.628    0.589      0.329    0.780     0.754

=== Confusion Matrix ===
     a     b   <-- classified as
  4839 10317 |     a = Neg
   917  125 |     b = Pos
```

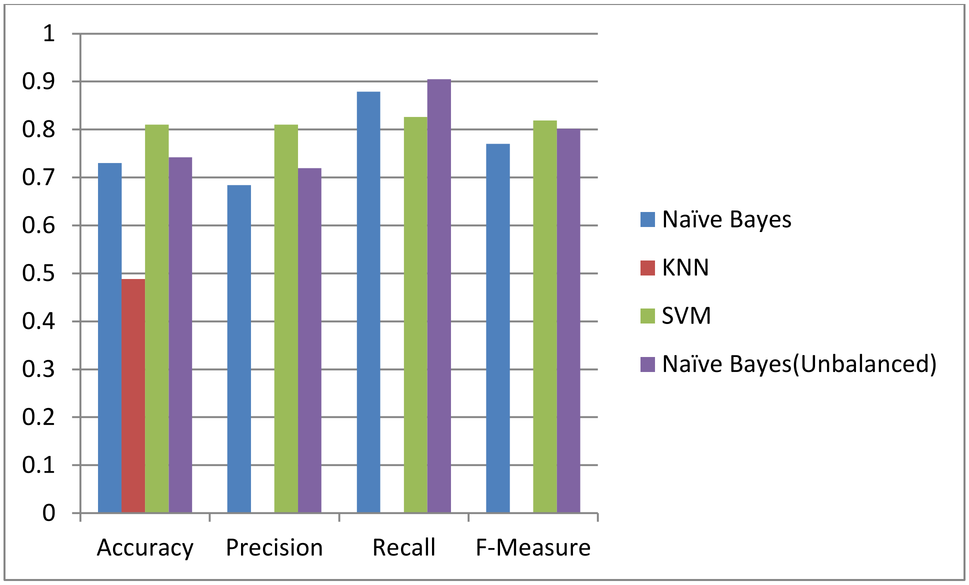
7.5. Experiments with the Full dataset using the Pattern Library in Python

The experiments as performed in Weka are also performed in the Pattern Library of Python . the parameters are varied and the results are evaluated on a sample file of 10 tweets.The parameters of the Corpus used in the experiments are :

Number of Negative Tweets: 16021
Number of Positive Tweets: 16805
number of documents: 32826
number of words: 25098
number of words (average): 5.33244988728

The Performance Measures and parameters for each of the methods on 10-fold cross-validation are as shown below:

Classifier	Accuracy	Precision	Recall	F-Measure	Parameters
Naïve Bayes	0.73	0.684	0.879	0.77	
KNN	0.488	-	-	-	K=20,top300
SVM	0.81	0.81	0.826	0.8187	Linear Kernel
Naïve Bayes	0.742	0.7197	0.9046	0.8016	Unbalanced Data



As is clear from the above results the accuracy of SVM is very high on this data. Also most of the classifiers concentrate on improving the recall rather that the accuracy. With Naïve bayes, Reducing features leads to an increase in the accuracy but also a decrease in precision with the result that all the data points start getting classified to only one class thus giving a high Recall as well.

The resulting SVM classifier is now used to classify a sample file of 20 tweets as shown below. The original file had the first 10 field as negative while the next 10 were positive, The text below denotes the tweet and the result from the SVM classifier. The output 0 denotes a negative class while the output 1 denotes a positive class.

Tweet	Class
in london again cant wait to see my girlfriend negemoti	0
negemoti	0
days till prom still no date	1
bored of this focusing on my work plan already gone weeks without a drop of alcohol and ive had enough passmethejd	0
mins ago i was crying because i didnt wanna go work now im crying because my company has shut down and i dont have a job	0
annoo i want to go soo frickin bad shitweather	0
heady highminded lovers of pleasures more than lovers of god sad lonely Christians	1
having a form of godliness but denying the power thereof from such turn away sad lonely Christians	1
and everyday it feels like im losing you all over again missyousomuch	0
listening to magic makes me tour depressed even though they didnt sing it waa	0
prototype proton supported by sidney samson	1
heart this once again robs working with amazing actors and director mtts	1
heart squee vermont in one week for work of course but it still feels like a mini vacation to me craftbeer	1
posemoti heart sums up my whole mood	1

weeks from today ill be going home yay excited	1
fallinhard yourthebest	1
whole days to myself	1
followers on tumblr	0
on my math achievement test today	1
birthday prezies from my daughter posemoti my first chane	1

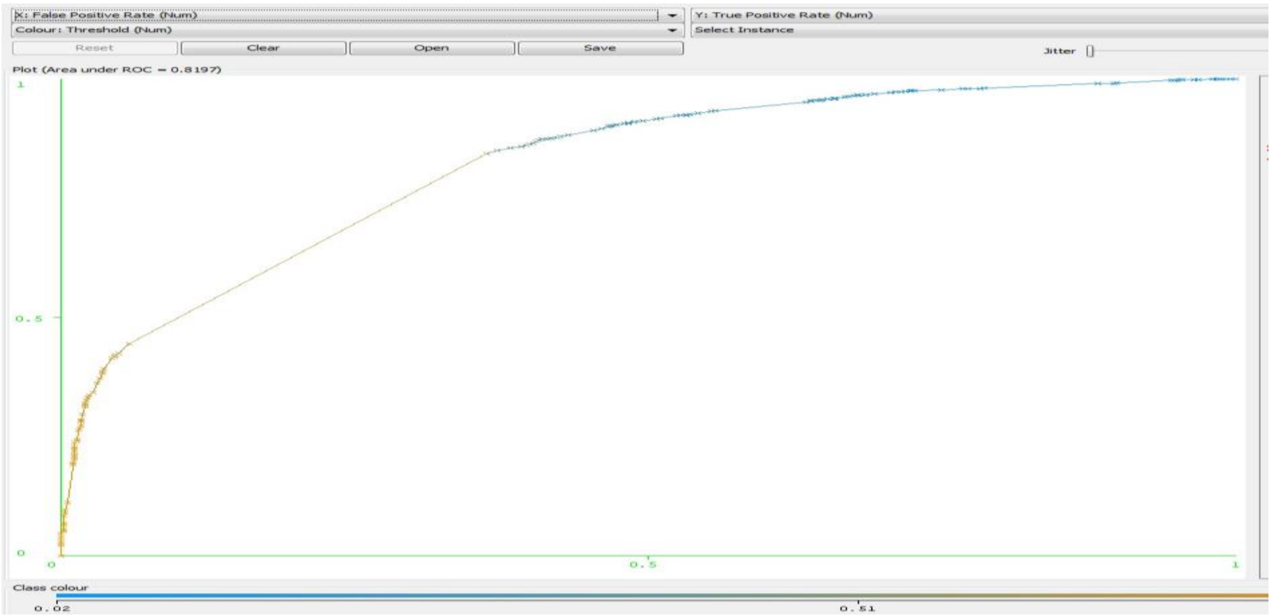
7.6. Experiments with Binary Vector Format of Docuements

Since tweets are short and the occurrence of repeated words don't imply much about its importance in a tweet, we try the same data but we convert it to a binary vector space where the occurrence of a word in a tweet is considered but not its TFIDF. The results obtained are as shown below:

Decision Trees J48: J48 Confidence 0.05 Min 4, Binary Model

```
Correctly Classified Instances       4481             74.1887 %
Incorrectly Classified Instances     1559             25.8113 %
Kappa statistic                         0.4819
Mean absolute error                     0.3315
Root mean squared error                 0.4119
Relative absolute error                66.3096 %
Root relative squared error            82.3735 %
Total Number of Instances            6040
=== Detailed Accuracy By Class ===

           TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
             0.843     0.362      0.706     0.843     0.769      0.82     Neg
             0.638     0.157      0.797     0.638     0.708      0.82     Pos
Weighted Avg.   0.742     0.262      0.751     0.742     0.739      0.82
=== Confusion Matrix ===
    a    b   <-- classified as
 2588  483 |    a = Neg
 1076 1893 |    b = Pos
```

The results for Decision trees are improved a lot by using the Binary model over the TFIDF model. The roc curve is shown below:



<u>Support Vector Machines</u>

```
Correctly Classified Instances      23051            76.3329 %
Incorrectly Classified Instances     7147            23.6671 %
Kappa statistic                         0.5264
Mean absolute error                     0.2367
Root mean squared error                 0.4865
Relative absolute error                47.3349 %
Root relative squared error            97.2984 %
Total Number of Instances           30198

=== Detailed Accuracy By Class ===
            TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
             0.821    0.295     0.737    0.821    0.777      0.763    Neg
             0.705    0.179     0.796    0.705    0.748      0.763    Pos
Weighted Avg.  0.763    0.237     0.767    0.763    0.763      0.763

=== Confusion Matrix ===
     a     b   <-- classified as
 12445  2711 |     a = Neg
  4436 10606 |     b = Pos
```

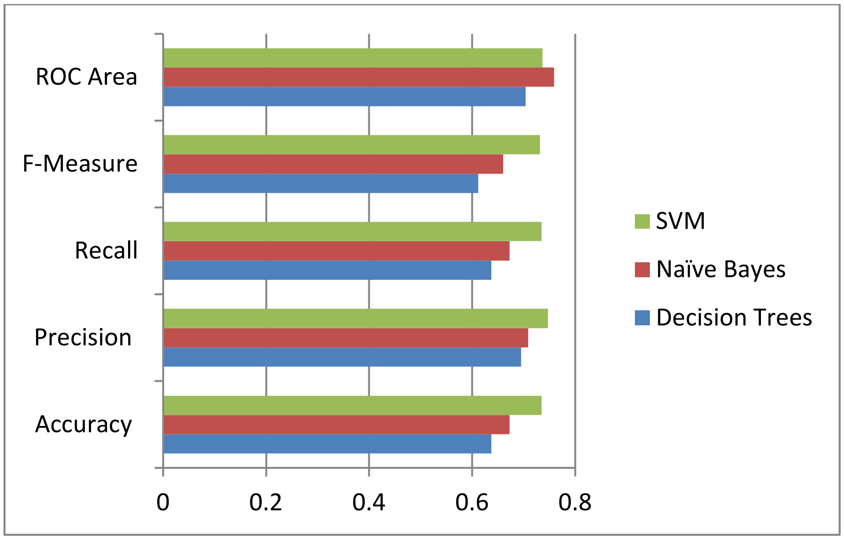
7.7. Summarization of Results

Summarization of the Experiments performed above is as given below.

<u>Results of Experimentation on 10% of the sample data</u>

Classifier	Accuracy	Precision	Recall	F-Measure	ROC Area
Decision Trees	0.637	0.695	0.637	0.612	0.704

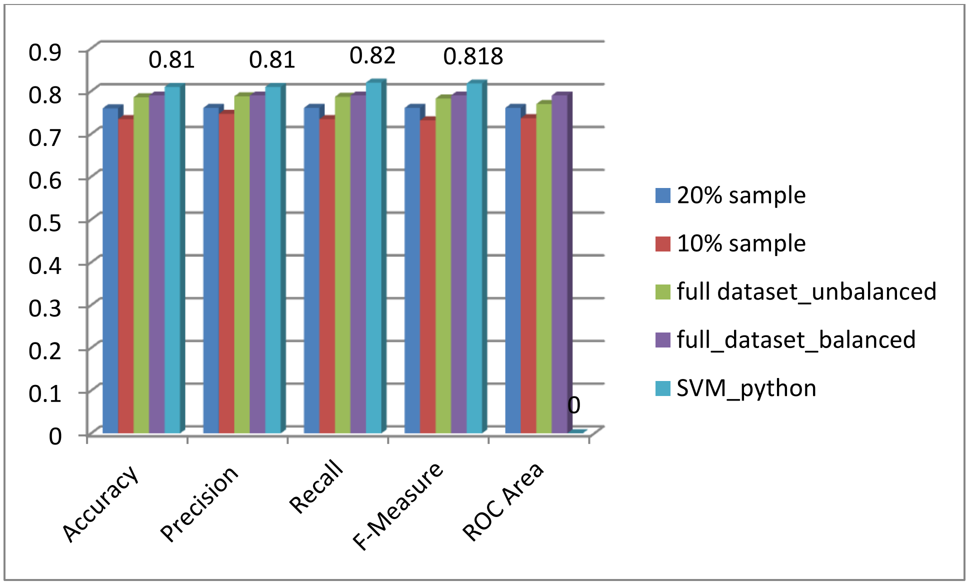
Naïve Bayes	0.6727	0.709	0.673	0.66	0.759
SVM	0.7347	0.747	0.735	0.732	0.737



The above graph shows clearly that the results obtained are the best for SVM. However, NaiveBayes has more AUC.

Results of SVM on different Datasets

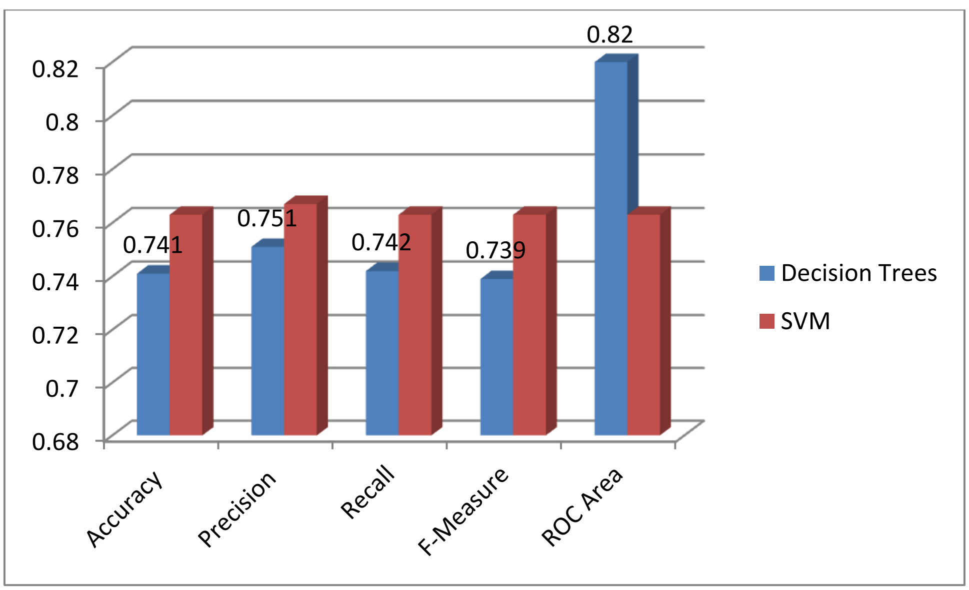
Clasifier	Accuracy	Precision	Recall	F-Measure	ROC Area
20% sample	0.76	0.761	0.761	0.761	0.761
10% sample	0.7347	0.747	0.735	0.732	0.737
full dataset_unbalanced	0.786	0.788	0.787	0.783	0.77
full_dataset_balanced	0.79	0.79	0.79	0.79	0.79
SVM_python	0.81	0.81	0.82	0.818	0



Results obtained for SVM on various dataset sizes are shown above. The highest values are obtained for SVM when using the Python Patterns toolbox. These values are highlighted in the graph above.

Results from Binary Vector Space Model

Classifier	Accuracy	Precision	Recall	F-Measure	ROC Area
Decision Trees	0.741	0.751	0.742	0.739	0.82
SVM	0.763	0.767	0.763	0.763	0.763



What is interesting in this case is that Decision tree outperforms SVM when it comes to binary vector space models. This is understandable as decision tree depends mostly on a count of data.

8. Conclusion

In this project, a proof of concept was implemented aimed at detecting emotions from tweets. A basic implementation of the model involves classification of tweets as either positive or Negative Based on the text in the tweets. The project involved collection of tweets, based on 10 hashtags, half of which were Positive mood denoting while the other half denotes Negative moods. Upon collection, Various cleaning steps were performed on the data to reduce it to a form suitable for classification. Following this Data cleaning, the tweets were converted to the Vector Space format using the TFIDF weighting and removal of Stopwords and Stemming. The obtained corpus was passed through various classifiers to test their ability to classify the data.

To test the performance of various classifier, 10% of the sample data was chosen at first to produce faster results. The Naïve Bayes algorithm and the SVM performed well on this data. Decision trees helped in gaining insights as to how the data was being classified. When the experimentation was moved to using the Full dataset, SVM surpassed all other methods in terms of Accuracy and F-Measure. The ability of SVM to classify high dimensional data was evident by it obtaining an accuracy of ~81% on 10-fold crossvalidation on the entire corpus,Naïve bayes was able to produce an accuracy of only ~74%, while Decision trees was took an enormous amount of time to compute and had to ultimately be shut down.

An interesting observation was that when considered the Binary model for documents, the Decision tree algorithm performed much better than its counterparts. Its accuracy rose to ~74% from its previous value of ~68%. This is because the decision tree algorithm works much better on binary and nominal values than continuous values.

Although SVM was able to produce high accuracy on this data, it is still not sure if the model will work for all tweets in real time. This is because most of the tweets are usually small and the vocabulary is not constant. Thus if a tweet contains words which the model has not yet seen , its performance cannot be judged in such a scenario. Also, the same words could be used to denote very different meanings and emotions, for example people use the term sad and happy in the same tweet. Also the model is currently not equipped for identifying Neutral tweets. This would be an interesting task for the future work.

References

Wang, Wenbo, et al. "Harnessing Twitter" Big Data" for Automatic Emotion Identification." *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*. IEEE, 2012.

De Smedt, T. & Daelemans, W. (2012). Pattern for Python. Journal of Machine Learning Research, 13: 2031–2035.