***Increment-1***

**Team members:**

Rahul Mandaloju, 11514828

Lokesh Punati, 11514358

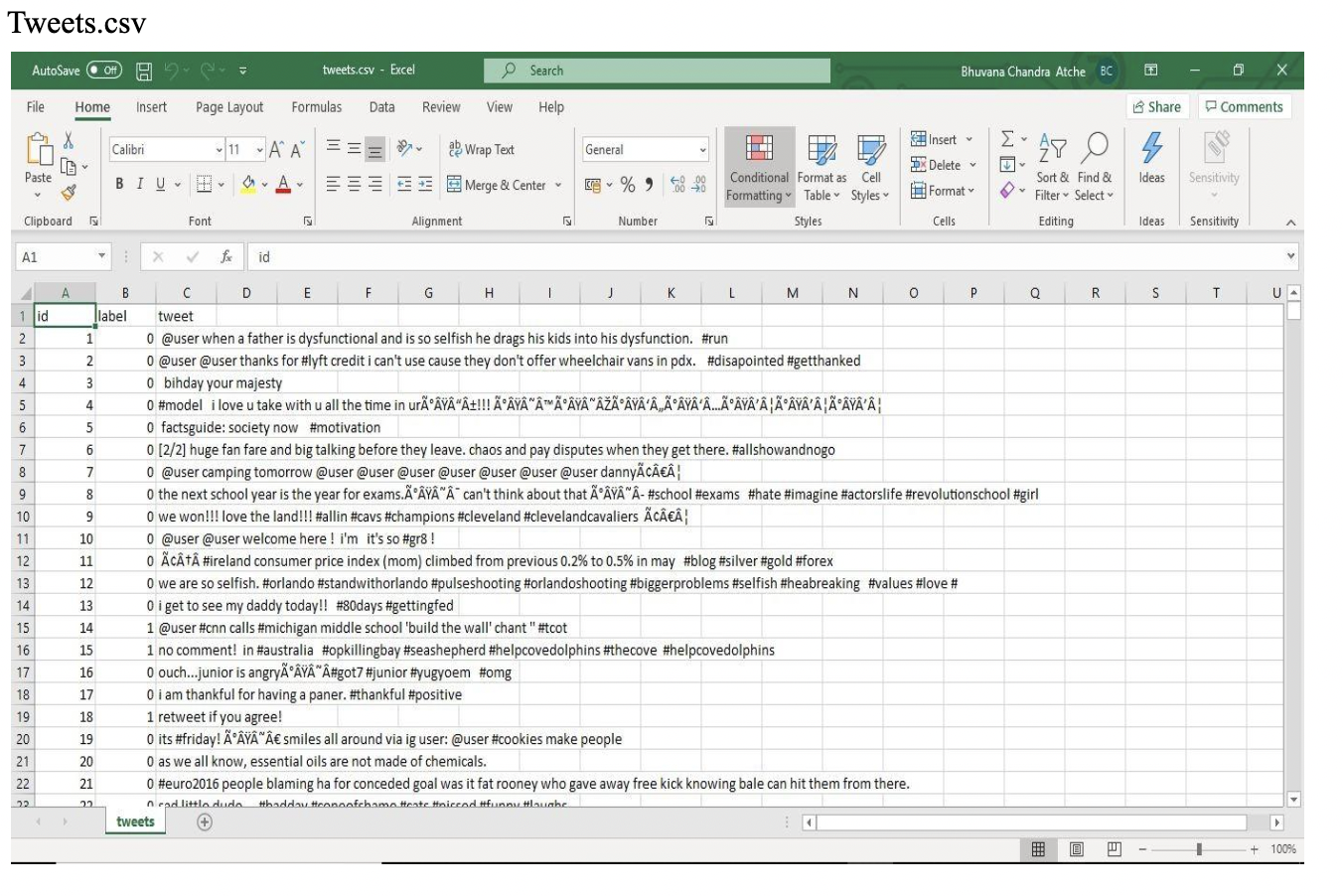
**Goals and Objectives**

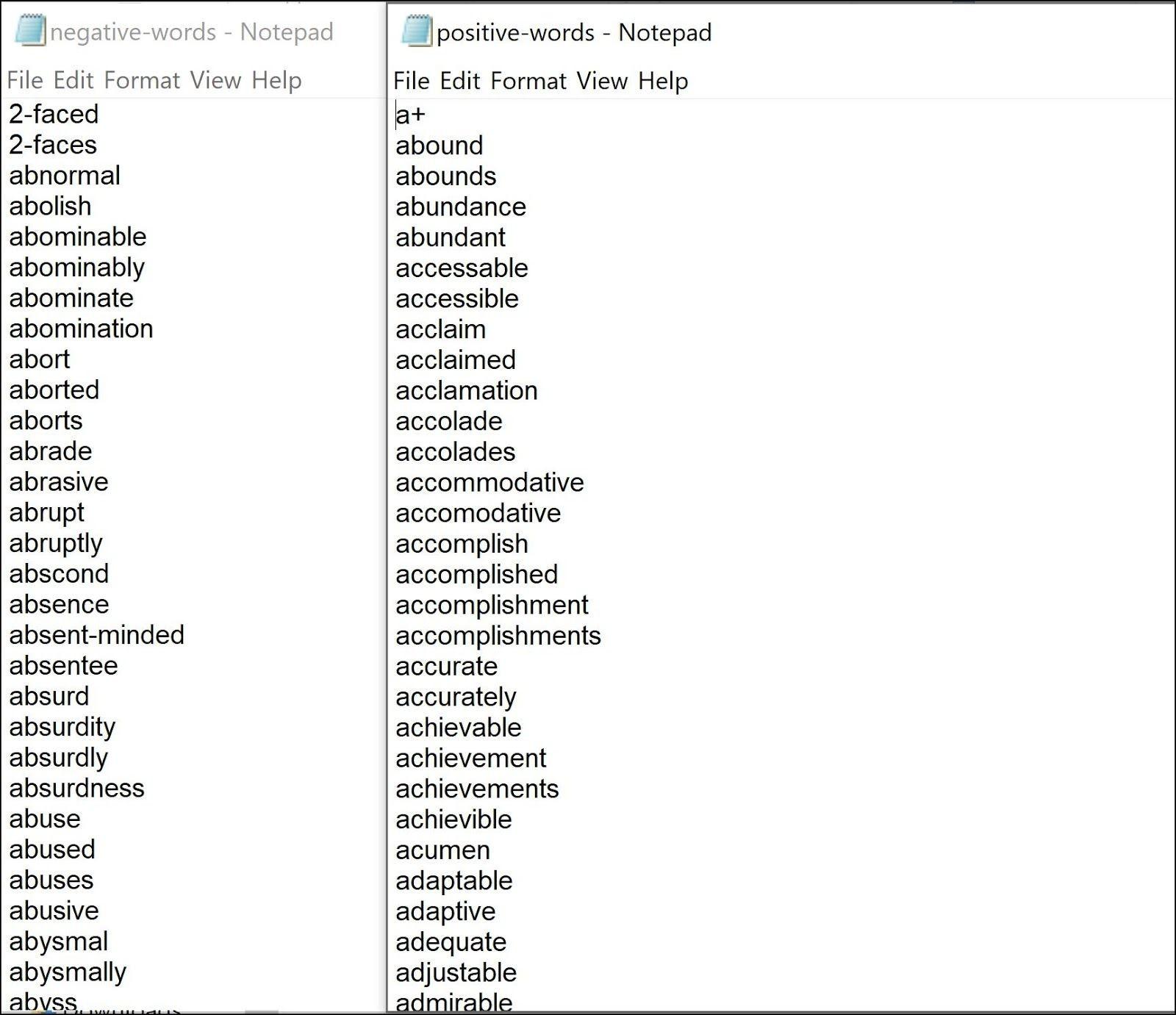
Twitter is a popular social networking website where members create and interact with messages known as “tweets”. This serves as a means for individuals to express their thoughts or feelings about different subjects. Various different parties such as consumers and marketers have done sentiment analysis on tweets to gather insights into products or to conduct market analysis. Furthermore, with the recent advancements in machine learning algorithms, we are able to improve the accuracy of our sentiment analysis predictions. In this report, we will attempt to conduct sentiment analysis on “tweets” using various different machine learning algorithms. We attempt to classify the polarity of the tweet where it is either positive or negative. If the tweet has both positive and negative elements, the more dominant sentiment should be picked as the final label.

We use the dataset from [Kaggle](https://www.kaggle.com/c/cs5228-project-2/data) which was crawled and labeled positive/negative. The data provided comes with emoticons, usernames and hashtags which are required to be processed and converted into a standard form. We also need to extract useful features from the text such as unigrams and bigrams which is a form of representation of the “tweet”. We use various machine learning algorithms to conduct sentiment analysis using the extracted features. However, just relying on individual models did not give a high accuracy so we picked the top few models to generate a model ensemble. Ensembling is a form of meta learning algorithm technique where we combine different classifiers in order to improve the prediction accuracy. Finally, we report our experimental results and findings at the end.

**Data Description**

The data given is in the form of comma-separated values files with tweets and their corresponding sentiments. The training dataset is a csv file of type tweet\_id,sentiment,tweet where thetweet\_id is a unique integer identifying the tweet, sentiment is either 1 (positive) or 0 (negative),and tweet is the tweet enclosed in "". Similarly, the test dataset is a csv file of type tweet\_id,tweet.

Positive and negative words collection in a text file.Around 10000 words in which 5000 are positive and 5000 are negative are collected and compared when labeling tweets as positive or negative.



***Methodology and Implementation***

**Pre-processing:**

Raw tweets scraped from twitter generally result in a noisy dataset. This is due to the casual nature of people’s usage of social media. Tweets have certain special characteristics such as retweets,emotions, user mentions, etc. which have to be suitably extracted. Therefore, raw twitter data has to be normalized to create a dataset which can be easily learned by various classifiers. We have applied an extensive number of pre-processing steps to standardize the dataset and reduce its size. We first do some general pre-processing on tweets which is as follows.

Convert the tweet to lower case.

Replace 2 or more dots (.) with space.

Strip spaces and quotes (" and ’) from the ends of the tweet. Replace 2 or more spaces with a single space.

We handle special twitter features as follows.

**URL**

Users often share hyperlinks to other webpages in their tweets. Any particular URL is not important for text classification as it would lead to very sparse features. Therefore, we replace all the URLs in tweets with the word URL. The regular expression used to match URLs

is ((www\.[\S]+)|(https?://[\S]+)).

**User Mention**

Every twitter user has a handle associated with them. Users often mention other users in their tweets by @handle. We replace all user mentions with the word USER\_MENTION. The regular expression used to match user mention is @[\S]+.

Emoticon(s) Type Regex Replacement

:), : ), :-), (:, ( :, (-:, :’) Smile (:\s?\)|:-\)|\(\s?:|\(-:|:\’\)) EMO\_POS

:D, : D, :-D, xD, x-D, XD, X-D Laugh (:\s?D|:-D|x-?D|X-?D) EMO\_POS

;-), ;), ;-D, ;D, (;, (-; Wink (:\s?\(|:-\(|\)\s?:|\)-:) EMO\_POS

<3, :\* Love (<3|:\\*) EMO\_POS

:-(, : (, :(, ):, )-: Sad (:\s?\(|:-\(|\)\s?:|\)-:) EMO\_NEG

:,(, :’(, :"( Cry (:,\(|:\’\(|:"\() EMO\_NEG

Table 3: List of emoticons matched by our method

**Emoticon**

Users often use a number of different emoticons in their tweets to convey different emotions. It is impossible to exhaustively match all the different emoticons used on social media as the number is ever increasing. However, we match some common emoticons which are used very frequently. We replace the matched emoticons with either EMO\_POS or EMO\_NEG depending on whether it is conveying a positive or a negative emotion.

***Vector Representation***

**Sparse Vector Representation**

Depending on whether or not we are using bigram features, the sparse vector representation of each tweet is either of length 5000 (when considering only unigrams) or 15000 (when considering unigrams and bigrams). Each unigram (and bigram) is given a unique index depending on its rank. The feature vector for a tweet has a positive value at the indices of unigrams (and bigrams) which are present in that tweet and zero elsewhere which is why the vector is sparse. The positive value at the indices of unigrams (and bigrams) depends on the feature type we specify which is one of presence and frequency. • presence In the case of presence feature type, the feature vector has a 1 at indices of unigrams (and bigrams) present in a tweet and 0 elsewhere. • frequency In the case of frequency feature type, the feature vector has a positive integer at indices of unigrams (and bigrams) which is the frequency of that unigram (or bigram) in the tweet and 0 elsewhere. A matrix of such term-frequency vectors is constructed for the entire training dataset and then each term frequency is scaled by the

inverse-document-frequency of the term (idf) to assign higher values to important terms. The inverse-document-frequency of a term t is dened as. idf(t) = log 1 + nd 1 + df(d,t)+ 1 where nd is the total number of documents and df(d,t) is the number of documents in which the term t occurs. Handling Memory Issues Which dealing with sparse vector representations, the feature vector for each tweet is of length 2000 and the total number of tweets in the training set is 16000 which means allocation of memory for a matrix of size 16000×2000. Assuming 4 bytes are required to represent each oat value in the matrix, this matrix needs a memory of 8×1024 bytes (≈ 2 GB) which is far greater than the memory available in common notebooks. To tackle this issue, we used scipy.sparse.lil\_matrix data structure provided by Scipy which is a memory eﬃcient linked list based implementation of sparse matrices. In addition to that, we used Python generators wherever possible instead of keeping the entire dataset in memory.

**Dense Vector Representation**

For dense vector representation we use a vocabulary of unigrams of size 10000 i.e. the top 10000 words in the dataset. We assign an integer index to each word depending on its rank (starting from 1) which means that the most common word is assigned the number 1, the second most common word is assigned the number 2 and so on. Each tweet is then represented by a vector of these indices which is a dense vector.

**Feature Extraction**

We extract two types of features from our dataset, namely unigrams and bigrams. We create a frequency distribution of the unigrams and bigrams present in the dataset and choose top N unigrams and bigrams for our analysis.

**Conclusion**

***Summary of achievements***

The provided tweets were a mixture of words, emoticons, URLs, hashtags, user mentions, and symbols. Before training we pre-process the tweets to make it suitable for feeding into models. We implemented several machine learning algorithms like Naive Bayes and Decision Tree to classify the polarity of the tweet. We used two types of features namely unigrams and bigrams for classication and observes that augmenting the feature vector with bigrams improved the accuracy. Once the feature has been extracted it is represented as either a sparse vector or a dense vector. It has been observed that presence in the sparse vector representation recorded a better performance than frequency.

Our model achieved an accuracy of 83.4% on the Kaggle dataset.

**Future directions**

Handling emotion ranges: We can improve and train our models to handle a range of sentiments. Tweets don’t always have positive or negative sentiment. At times they may have no sentiment i.e. neutral. Sentiment can also have gradations like the sentence, This is good, is positive but the sentence, This is extraordinary. is somewhat more positive than the rst. We can therefore classify the sentiment in ranges, say from -2 to +2.

Using symbols: During our pre-processing, we discard most of the symbols like commas, full-stops, and exclamation marks. These symbols may be helpful in assigning sentiment to a sentence.

References: https://www.kaggle.com/c/cs5228-project-2/data