SENTIMENT ANALYSIS OF TWEETS

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Project Report Of
Statistical Methods In AI

Guided By:-

Dr. Ravi Kiran

TA Sarthak Sharma

Submitted By:-

Sopnesh Gandhi (2018201064)

Rushitkumar Jasani (2018201034)

Priyendu Mori (2018201103)

Niharika Khare (2018201002)

SENTIMENT ANALYSIS OF TWEETS

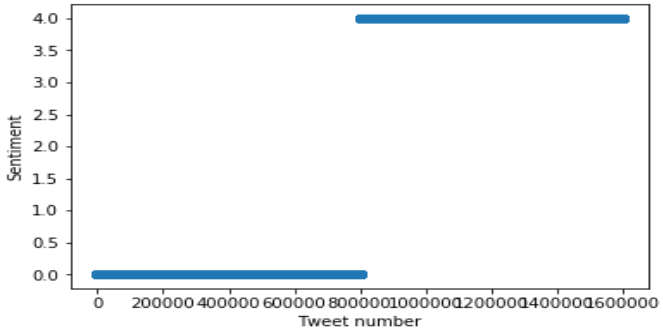
Sentiment analysis is contextual mining of text which identifies and extracts subjective information in source material.Sentiment Analysis is the most common text classification tool that analyses an incoming message and tells whether the underlying sentiment is positive or negative. Sentiment analysis has many applications and benefits to your business and organization. It can be used to give your business valuable insights into how people feel about your product, brand or service. When applied to social media channels, it can be used to identify spikes in sentiment, thereby allowing you to identify potential product advocates or social media influencers.

DATASET

For the twitter sentiment analysis model, I have chosen a dataset of tweets labelled either positive or negative. The dataset for training the model is from "Sentiment140", a dataset originated from Stanford University. More info on the dataset can be found from the below link.
http://help.sentiment140.com/for-students/

The dataset can be downloaded from the below link.
http://cs.stanford.edu/people/alecmgo/trainingandtestdata.zip
By looking at the description of the dataset from the link, the information on each field can be found.
0 - the polarity of the tweet (0 = negative, 4 = positive)
1 - the id of the tweet (2087)
2 - the date of the tweet (Sat May 16 23:58:44 UTC 2009)
3 - the query (lyx). If there is no query, then this value is NO_QUERY.
4 - the user that tweeted (robotickilldozer)
5 - the text of the tweet (Lyx is cool)

We have also used data from facebook and amazon reviews during testing to make sure that model perform good with greater length text.



APPROACH

Flow of the project is divided into sub-parts:-

1. Data collection and analysis
2. Pre-processing of tweets
3. Extracting features from processed tweets
4. Model building
5. Testing

Pre-Processing Of Tweets

The preprocessing of the text data is an essential step as it makes the raw text ready for mining, i.e., it becomes easier to extract information from the text and apply machine learning algorithms to it. If we skip this step then there is a higher chance that you are working with noisy and inconsistent data. The objective of this step is to clean noise those are less relevant to find the sentiment of tweets. Cleaning of noise includes:

1. **Removing username** :- The Twitter handles are already masked as @user due to privacy concerns. So, these Twitter handles are hardly giving any information about the nature of the tweet.

2. **Converting string to lowercase :-** Conversion of string to lowercase does not affect the sentiment of the tweet and also provides generalisation.

3. **Removing Hyperlinks :-** Hyperlinks adds no information to the sentiment of the text.

4. **Removing hash of hashtags :-** Removing hash of hashtag and extracting tag as some tags such as '#beautiful' may provide some useful information.

5. **Removing of words with length less than three :-** Most of the smaller words do not add much value. For example, 'is','as',etc . So, we removed them from our data.

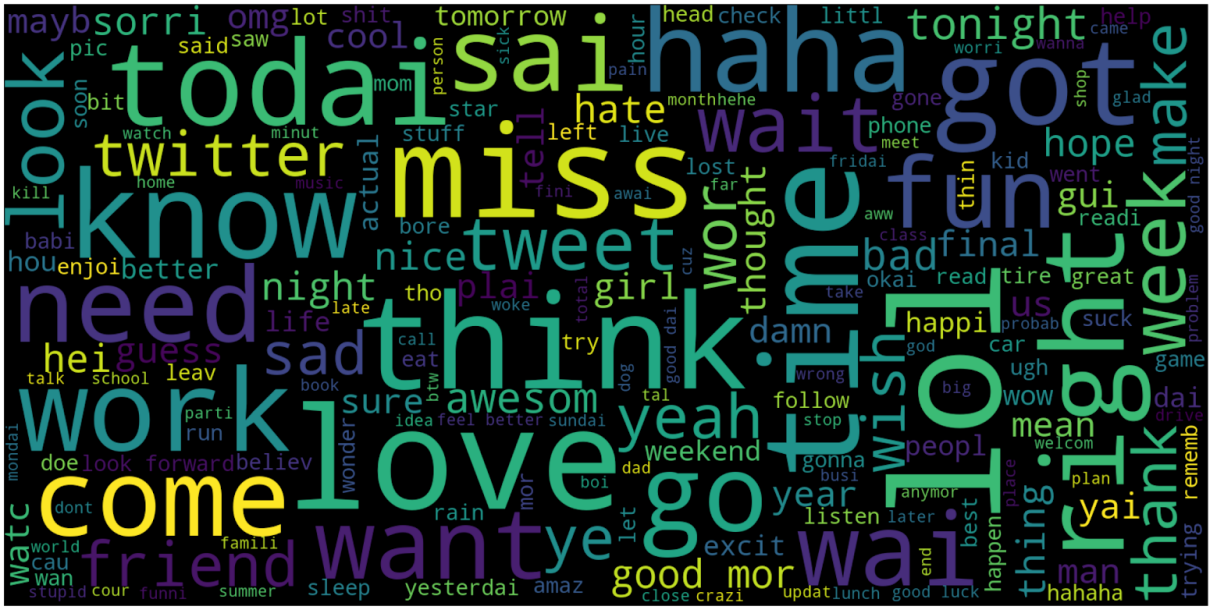
6. **Removing Punctuations :-** We can also think of getting rid of the punctuations and even special characters since they wouldn't help in differentiating different kinds of tweets.

7. **Removing stop words :-** Stop words such as 'the','for' etc do not add any information to the sentiment of tweet hence we removed them from our data.

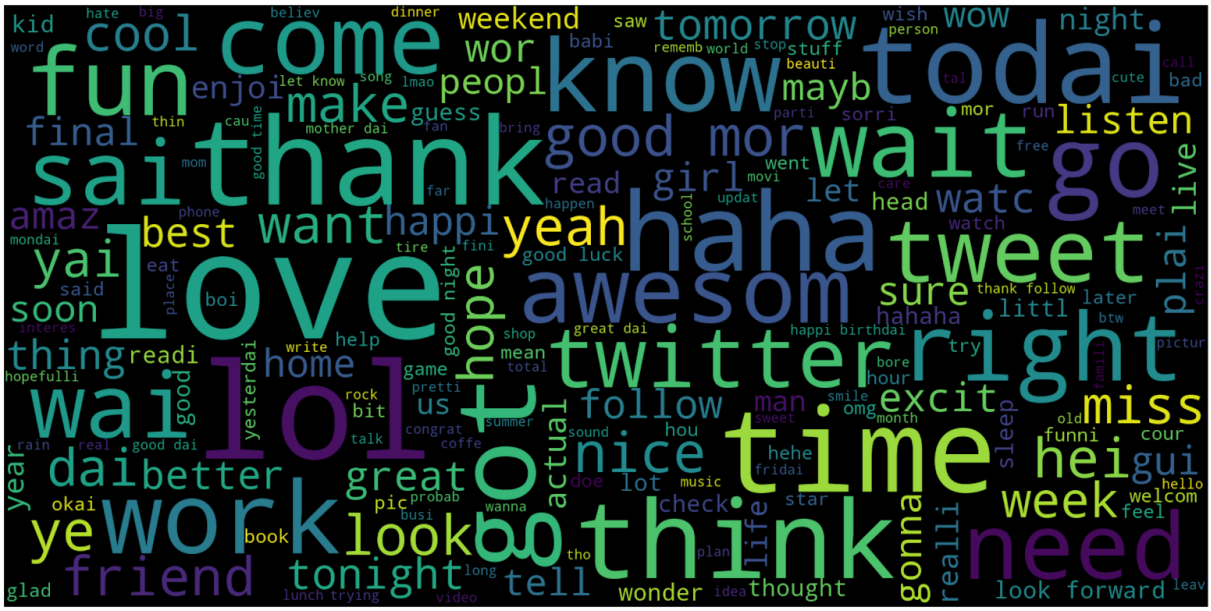
8. **Stemming :-** We might have terms like love,loves, loving, lovable, etc. in the data. These terms are often used in the same context. If we can reduce them to their root word, which is 'love', then we can reduce the total number of unique words in our data

without losing a significant amount of information. We have implemented porter stemmer function of nltk for the same.

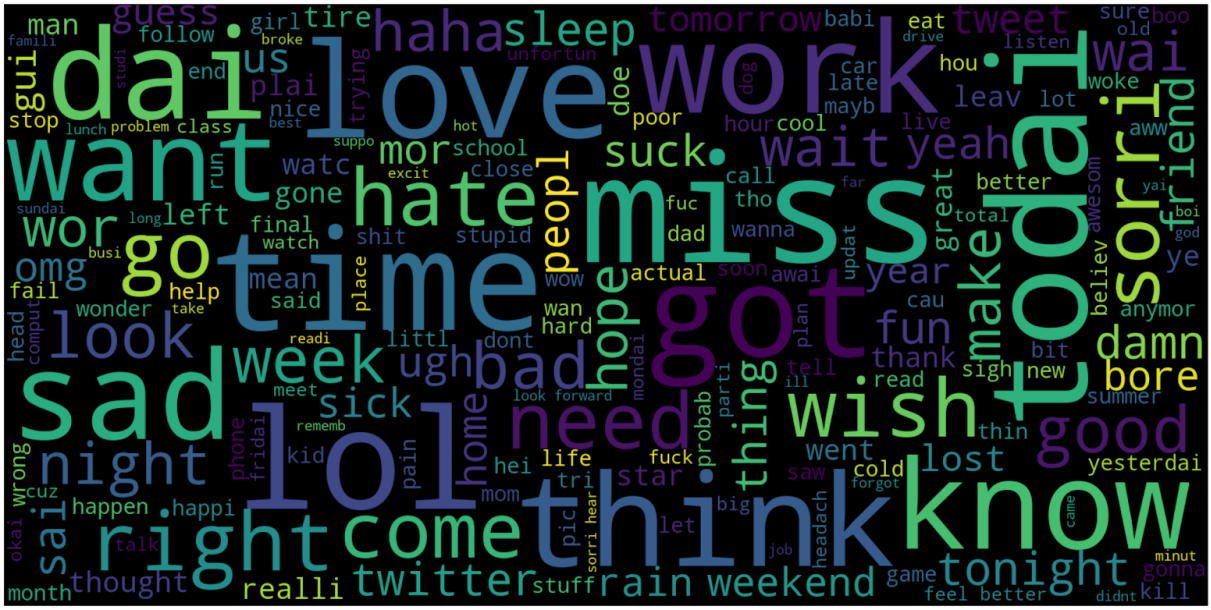
All words word cloud



Positive words word cloud



Negative words word cloud



Extracting Features From Processed Tweets

To analyze a preprocessed data, it needs to be converted into features. Depending upon the usage, text features can be constructed using assorted techniques. We have implemented feature extraction using two techniques, bag-of-words and TF-IDF.

Bag Of Words :-

The bag-of-words model is a simplifying representation in which a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. Bag-of-word is an order less document representation, that is, only the counts of words mattered. For instance, in the above example "John likes to watch movies. Mary likes movies too", the bag-of-words representation will not reveal that the verb "likes" always follows a person's name in this text.

In practice, the Bag-of-words model is mainly used as a tool of feature generation. After transforming the text into a "bag of words", we can calculate various measures to characterize the text. The most common type of characteristics, or features calculated from the Bag-of-words model is term frequency, namely, the number of times a term appears in the text.

TF-IDF :-

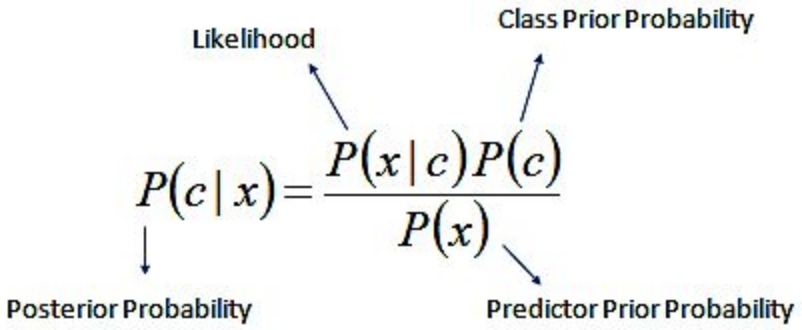
Bag-of-words represents each term and the frequency of its occurrence and use them as feature for the classifier. However, term frequencies are not necessarily the best representation for the text. Common words like "the", "a", "to" are almost always the terms with highest frequency in the text. Thus, having a high raw count does not necessarily mean that the corresponding word is more important. To address this problem, one of the most popular ways to "normalize" the term frequencies is to weight a term by the inverse of document frequency, or tf–idf. The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.
Variations of the tf–idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. tf–idf can be successfully used for stop-words filtering in various subject fields, including text summarization and classification.

Model Building

We have built three models and implemented each with bag-of-words and TF-IDF. With each implementation of bag-of-words and TF-IDF we have used concept of unigram and bigram, so, we have total of twelve models.

Naive Bayes Classifier

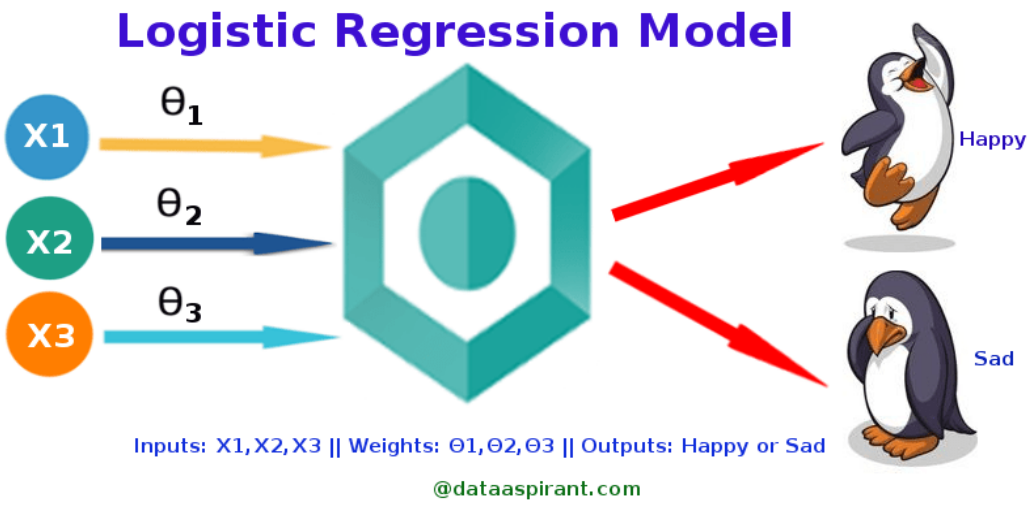
They are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with the assumption that features are independent to each other. This is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set.



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

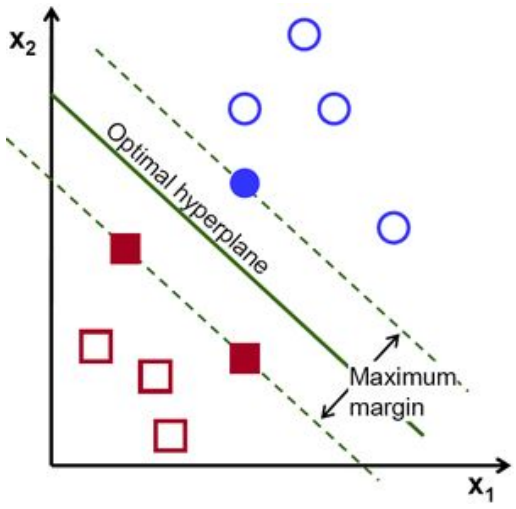
Logistic Regression

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.



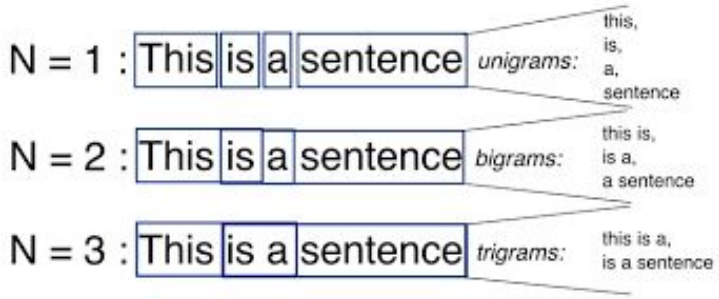
Support Vector Machine

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.



N-grams

N-grams are simply all combinations of adjacent words or letters of length *n* that you can find in your source text. For example, given the word fox, all 2-grams (or “bigrams”) are fo and ox. You may also count the word boundary – that would expand the list of 2-grams to #f, fo, ox, and x#, where # denotes a word boundary.



DIVISION OF WORK

Division of work was not possible before feature extraction step. After feature extraction we divided our group into two sub-groups. With each group implementing two of four models (Naive Bayes, Logistic Regression, SVM, NLTK).

CHALLENGES

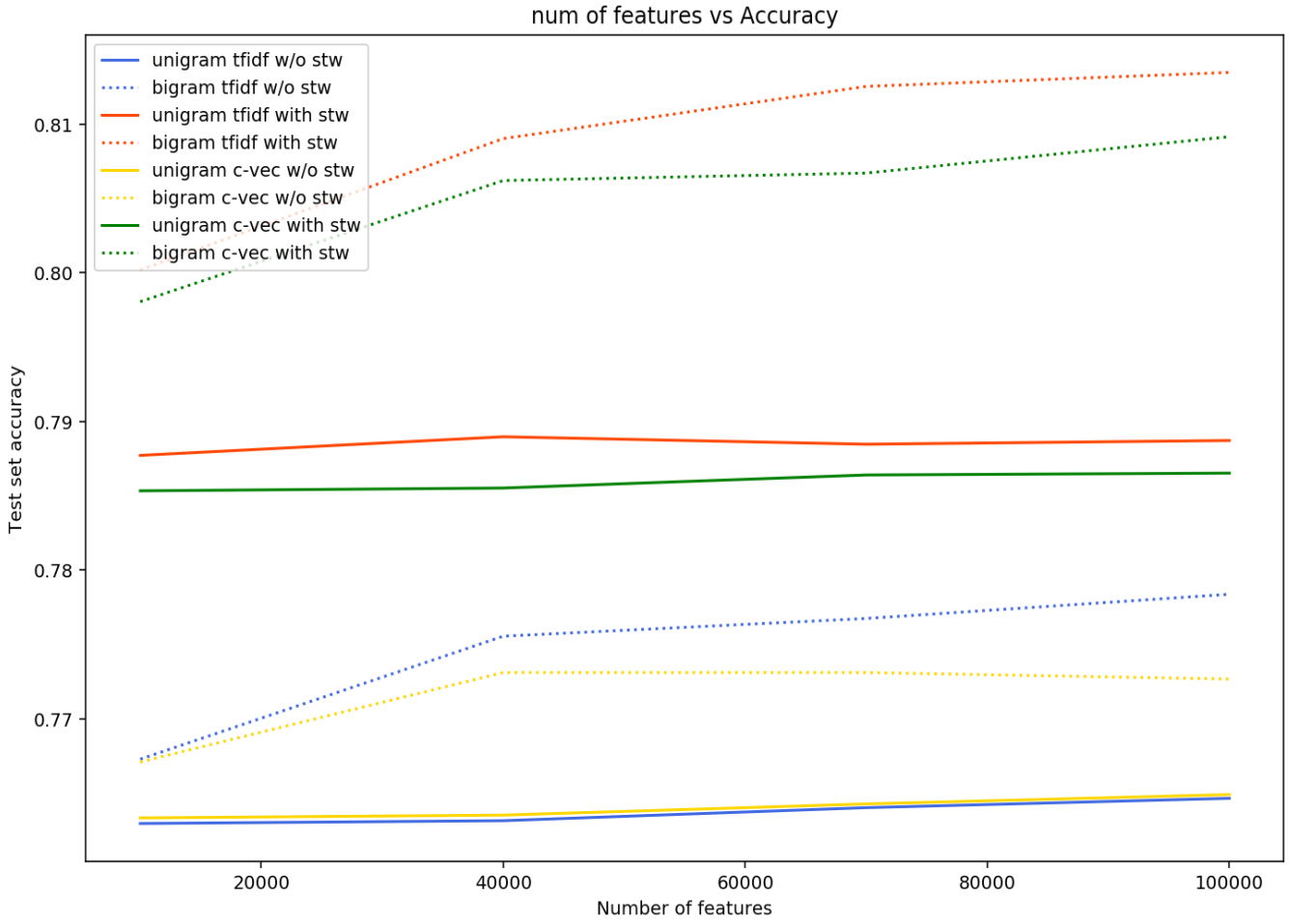
1. Understanding how to work with text and pre-process data.
2. Implementing stemming and other functions that are inbuilt in NLTK etc.
3. Deciding the models for training.
4. Working with huge data leads to huge training time especially on a low computing device like a personal computer.

OVERCOMING

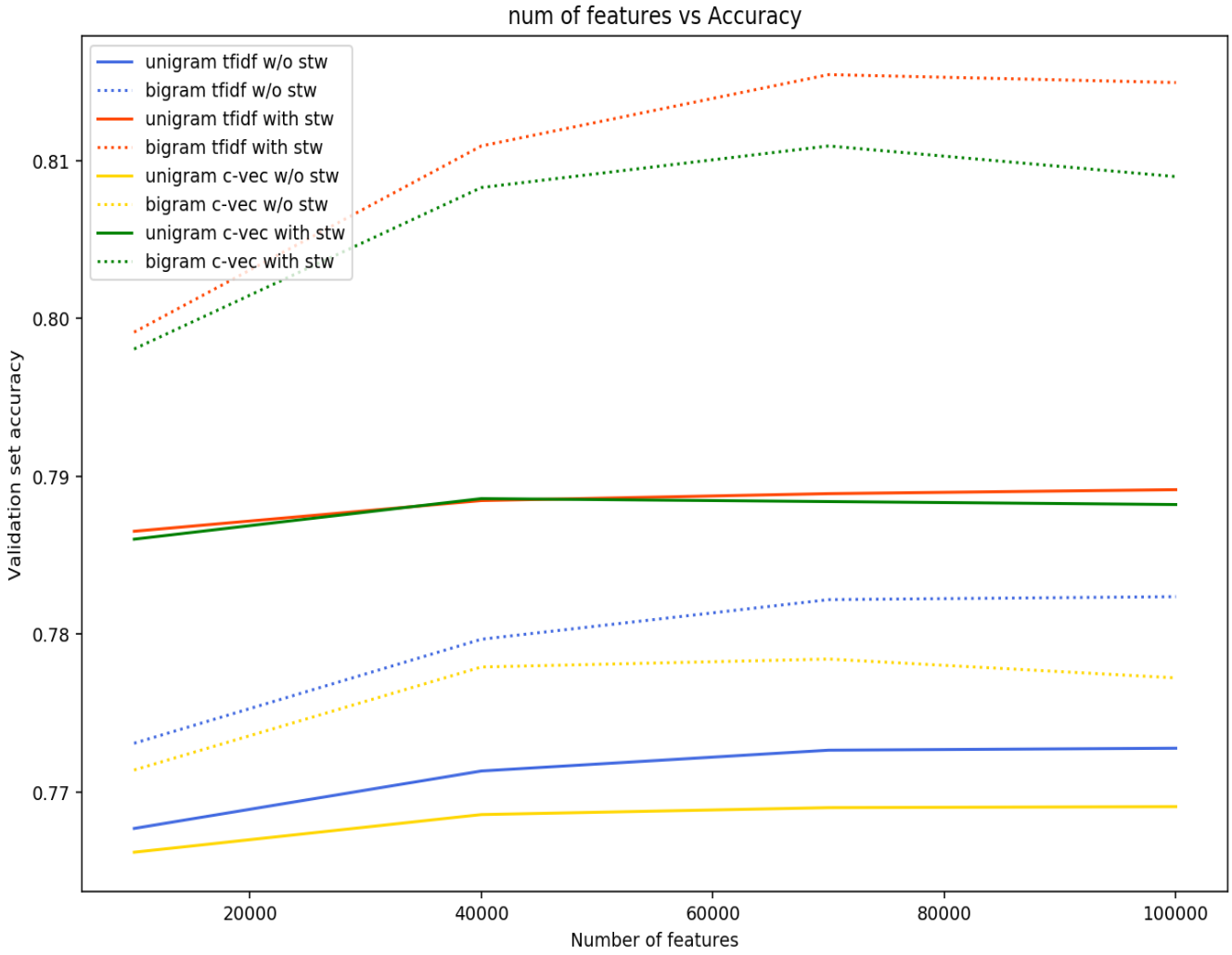
1. Used various resources to understand data processing and tried with various approaches.
2. Learnt about which model could be better for text processing.
3. Used concept of pipeline to reduce run time of code.

RESULT

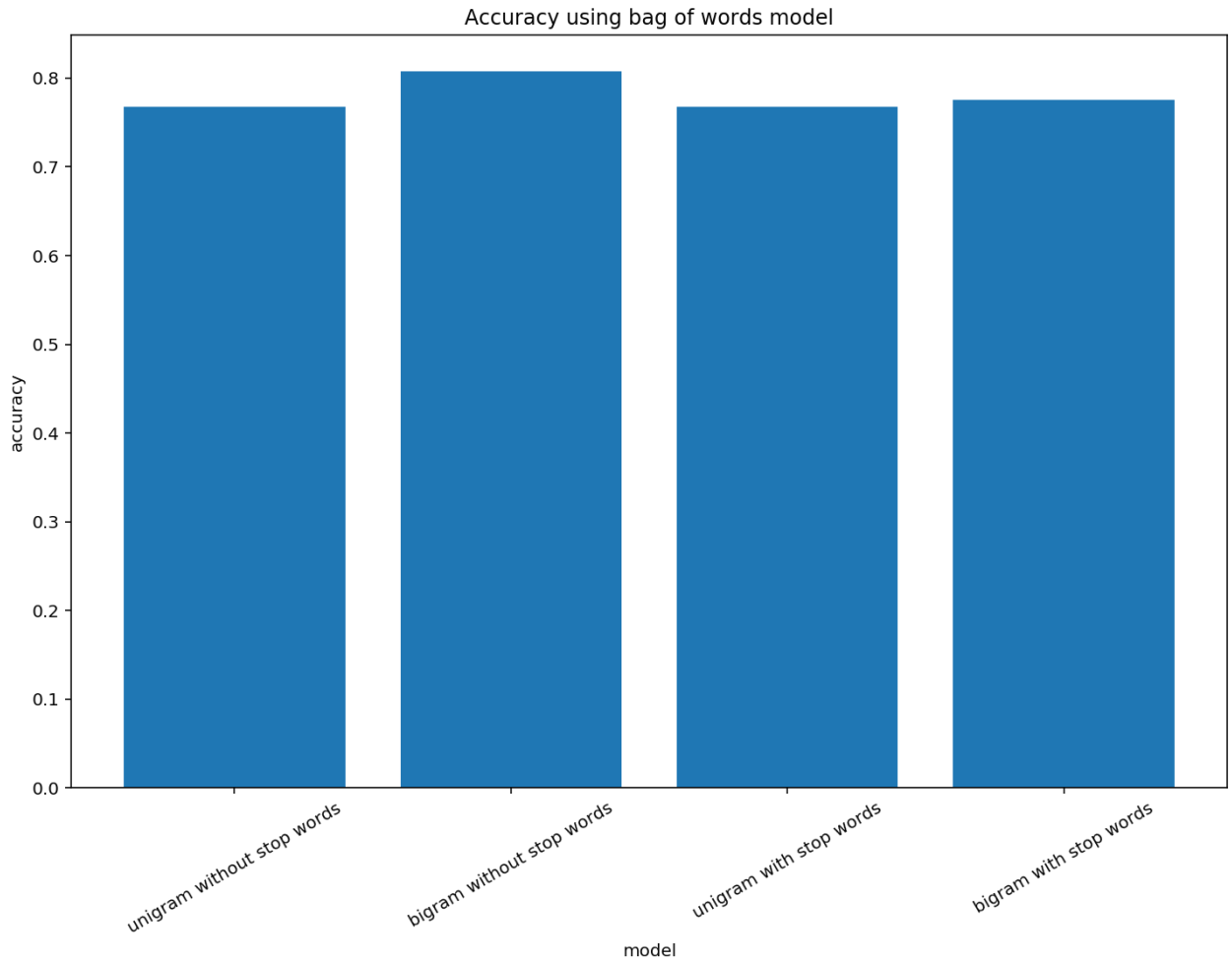
Logistic Regression Testing



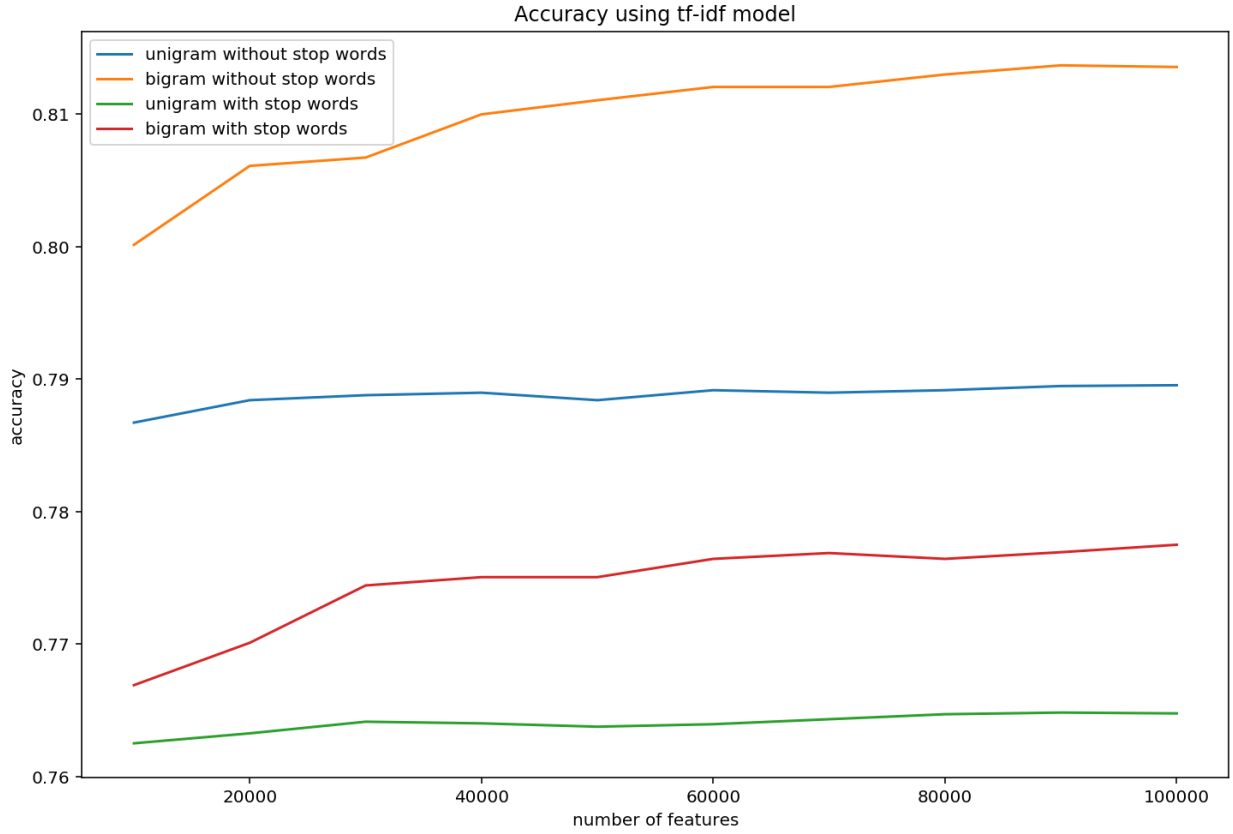
Logistic Regression Validation



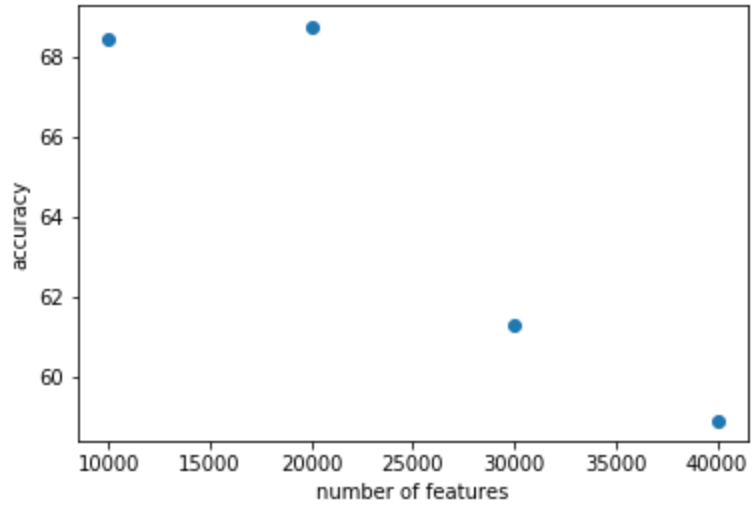
SVM with bag-of-words

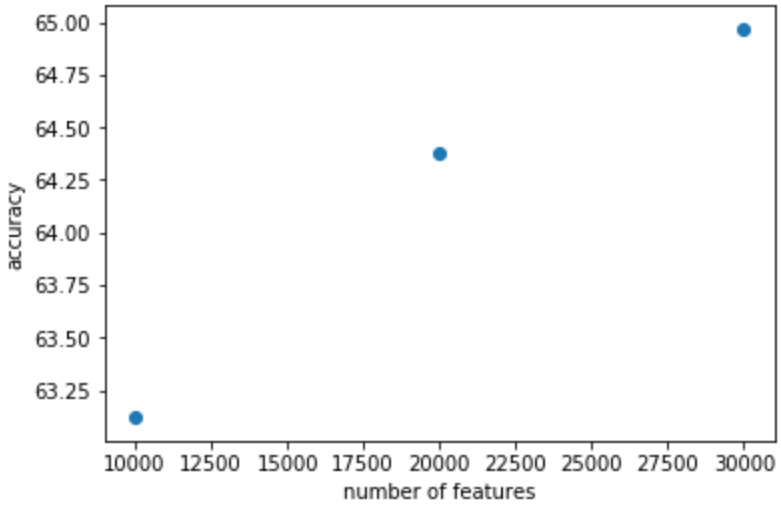


SVM using TF-IDF



Naive Bayes using bag-of-words




65.00
64.75
64.50
64.25
64.00
63.75
63.50
63.25
accuracy
10000
12500
15000
17500
20000
22500
25000
27500
30000
number of features

REFERENCES

https://pythonspot.com/category/nltk/
https://monkeylearn.com/sentiment-analysis/
https://en.wikipedia.org/wiki/Sentiment_analysis
https://en.wikipedia.org/wiki/Logistic_regression
https://en.wikipedia.org/wiki/Naive_Bayes_classifier
https://en.wikipedia.org/wiki/Support-vector_machine
https://github.com/abdulfatir/twitter-sentiment-analysis/blob/master/code/naivebayes.py
https://www.analyticsvidhya.com/blog/2018/07/hands-on-sentiment-analysis-dataset-python/#
https://www.researchgate.net/publication/33038304_The_Porter_stemming_algorithm_Then_and_now
http://snowball.tartarus.org/algorithms/porter/stemmer.html