REPORT FOR TWITTER SENTIMENT ANALYSIS

As a project work for Course

PYTHON PROGRAMMING (INT 213)

Name : Prashant Kumar & Aman Kumar
Reg. No : 12011059 & 12008676
Roll No : RK20RGB61 & RK20RGB09
Section : K20RG
Semester : 3rd Sem
Branch : CSE B.Tech
University : LPU

Date Of Submission : 10th/Nov/2021

TWITTER SENTIMENT ANALYSIS

10th / Nov / 2021

ABSTRACT:-

Twitter Sentiment Analysis is an extensive project where our task was to identify the racist/sexist or non-racist/sexist. Sentimental Analysis is one of the many applications of NLP (Natural Language Processing), This technique is used for extracting subjective information from text or speeches, like opinions or attitude or in general terms positive, negative, or neutral

Table of contents

1. Abstract	02
2. Libraries	06
3. Data Inspection	09
4. Data Cleaning	11
5. Visualization from the tweets	14
6. Logistic Regression	21
7.Support Vector Machine(SVM)	23
8. Summary	28

INTRODUCTION

Context:

This Project is part of my Final Project in course (INT 213: Python Programming). We had three months to finish the project.

Motivation:

Being immensely devoted to Data science, we were excited to work on this project which would push us beyond our limits. I love the fact that in Data Science I can assign everything a number and then play using pure mathematics is really amusing also it is my first time using NLP and I am really looking forward to complete it.

Problem Statement

The objective of project is to detect hate speech in tweets, for the sake of simplicity we can say a tweet contains hate speech if it has a sexist or racist sentiment associated with it, so we will classify the racist or sexist tweets from other tweets.

Officially, we were given a training sample of tweets and labels where ‘1’ denotes the tweet is racist or sexist and label ‘0’ denotes the tweet is neutral or positive (non-racist/sexist)

Libraries

NumPy

It is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.

As the whole project is based on whole complex stats ,we will use this fast calculations and provide results.

Pandas

Pandas is the most popular python library that is used for data analysis. We will provide highly optimized performance with back-end source code with the use of Pandas.

Matplotlib

Matplotlib tries to make easy things easy and hard things possible. We will generate plots, histograms, scatterplots, etc.,to make our project more appealing and easier to understand.

Seaborn

We will use it for statistical data visualization as **Seaborn** is a **Python** data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Warning

Warnings are provided to warn the developer of situations that aren't necessarily exceptions. Usually, a warning occurs when there is some obsolete of certain programming elements, such as keyword, function or class, etc. A warning in a program is distinct from an error. Python program terminates immediately if an error occurs. Conversely, a warning is not critical. It shows some message, but the program runs. The warn() function defined in the 'warning' module is used to show warning messages. The warning module is actually a subclass of Exception which is a built-in class in Python

Re

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.
RegEx can be used to check if a string contains the specified search pattern.

Nltk

The **Natural Language Toolkit**, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language.

Word cloud

Word Cloud is a **data visualization technique** used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud

Sklearn

The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction.

XGBoost

XGBoost is one of the most popular machine learning algorithms these days. Regardless of the type of prediction task at hand; regression or classification. XGBoost (Extreme Gradient Boosting) belongs to a family of boosting algorithms and uses the gradient boosting (GBM) framework at its core. It is an optimized distributed gradient boosting library

Gensim

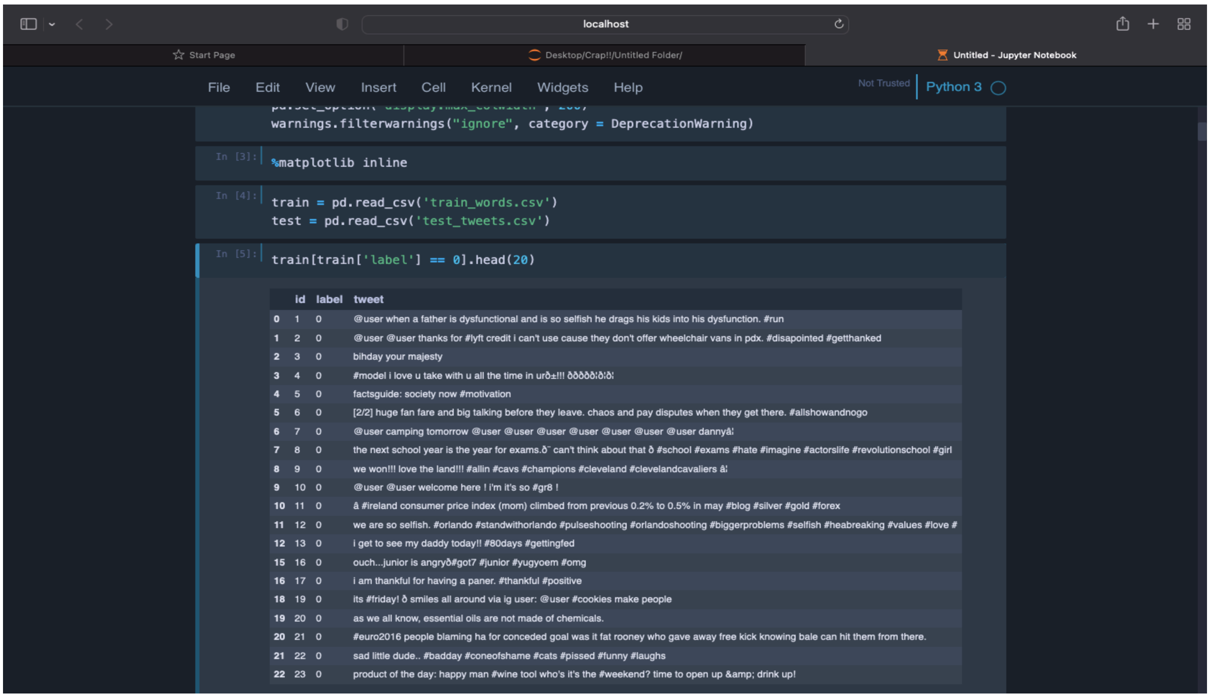
The fastest library for training of vector embeddings – **Python** or otherwise. The core algorithms in **Gensim** use battle-hardened, highly optimized **a free open-source Python library for representing documents as semantic vectors**, as efficiently (computer-wise) and painlessly (human-wise) as possible. Gensim is designed to process raw, unstructured digital texts (”plain text”) using unsupervised machine learning algorithms.

TQDM

tqdm is a library in Python which is used for creating Progress Meters or Progress Bars. tqdm got its name from the Arabic name ***taqaddum*** which means ‘progress’. Implementing tqdm can be done effortlessly in our loops, functions or even Pandas. Progress bars are pretty useful in Python

Data Inspection

checking out a few non racist/sexist tweets.



```
warnings.filterwarnings("ignore", category = DeprecationWarning)
```

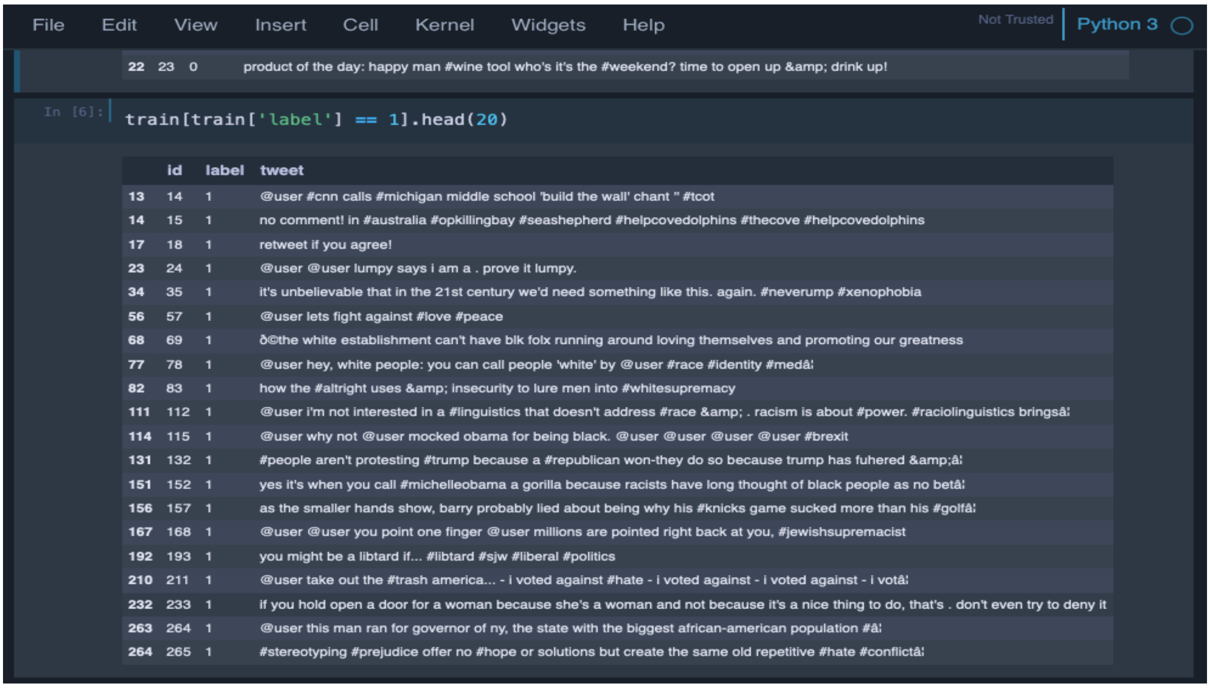
In [3]:
```
%matplotlib inline
```

In [4]:
```
train = pd.read_csv('train_words.csv')
test = pd.read_csv('test_tweets.csv')
```

In [5]:
```
train[train['label'] == 0].head(20)
```

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run
1	2	0	@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disapointed #getthanked
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in urð±!!! ðððð¦ð¦ð¦
4	5	0	factsguide: society now #motivation
5	6	0	[2/2] huge fan fare and big talking before they leave. chaos and pay disputes when they get there. #allshowandnogo
6	7	0	@user camping tomorrow @user @user @user @user @user @user @user dannyâ¦
7	8	0	the next school year is the year for exams.ð¯ can't think about that ð #school #exams #hate #imagine #actorslife #revolutionschool #girl
8	9	0	we won!!! love the land!!! #allin #cavs #champions #cleveland #clevelandcavaliers â¦
9	10	0	@user @user welcome here ! i'm it's so #gr8 !
10	11	0	â #ireland consumer price index (mom) climbed from previous 0.2% to 0.5% in may #blog #silver #gold #forex
11	12	0	we are so selfish. #orlando #standwithorlando #pulseshooting #orlandoshooting #biggerproblems #selfish #heabreaking #values #love #
12	13	0	i get to see my daddy today!! #80days #gettingfed
15	16	0	ouch...junior is angryð#got7 #junior #yugyoem #omg
16	17	0	i am thankful for having a paner. #thankful #positive
18	19	0	its #friday! ð smiles all around via ig user: @user #cookies make people
19	20	0	as we all know, essential oils are not made of chemicals.
20	21	0	#euro2016 people blaming ha for conceded goal was it fat rooney who gave away free kick knowing bale can hit them from there.
21	22	0	sad little dude.. #badday #coneofshame #cats #pissed #funny #laughs
22	23	0	product of the day: happy man #wine tool who's it's the #weekend? time to open up & drink up!

Now checking out a few racist/sexist tweets.



In [6]:
```
train[train['label'] == 1].head(20)
```

	id	label	tweet
13	14	1	@user #cnn calls #michigan middle school 'build the wall' chant " #tcot
14	15	1	no comment! in #australia #opkillingbay #seashepherd #helpcovedolphins #thecove #helpcovedolphins
17	18	1	retweet if you agree!
23	24	1	@user @user lumpy says i am a . prove it lumpy.
34	35	1	it's unbelievable that in the 21st century we'd need something like this. again. #neverump #xenophobia
56	57	1	@user lets fight against #love #peace
68	69	1	ð©the white establishment can't have blk folx running around loving themselves and promoting our greatness
77	78	1	@user hey, white people: you can call people 'white' by @user #race #identity #medâ¦
82	83	1	how the #altright uses & insecurity to lure men into #whitesupremacy
111	112	1	@user i'm not interested in a #linguistics that doesn't address #race & . racism is about #power. #raciolinguistics bringsâ¦
114	115	1	@user why not @user mocked obama for being black. @user @user @user @user #brexit
131	132	1	#people aren't protesting #trump because a #republican won-they do so because trump has fuhered &â¦
151	152	1	yes it's when you call #michelleobama a gorilla because racists have long thought of black people as no betâ¦
156	157	1	as the smaller hands show, barry probably lied about being why his #knicks game sucked more than his #golfâ¦
167	168	1	@user @user you point one finger @user millions are pointed right back at you, #jewishsupremacist
192	193	1	you might be a libtard if... #libtard #sjw #liberal #politics
210	211	1	@user take out the #trash america... - i voted against #hate - i voted against - i voted against - i votâ¦
232	233	1	if you hold open a door for a woman because she's a woman and not because it's a nice thing to do, that's . don't even try to deny it
263	264	1	@user this man ran for governor of ny, the state with the biggest african-american population #â¦
264	265	1	#stereotyping #prejudice offer no #hope or solutions but create the same old repetitive #hate #conflictâ¦

There are quite a many words and characters which are not really required. So, we will try to keep only those words which are important and add value.

But first check dimensions of the train and test dataset.

```
In [7]: train.shape

(31962, 3)

In [8]: test.shape

(17197, 2)
```

Train set has 31,962 tweets and test set has 17,197 tweets.

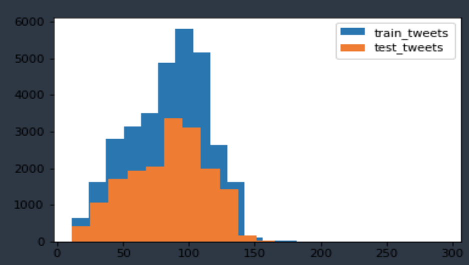
label-distribution in the train dataset.

```
In [9]: train["label"].value_counts()

0    29720
1     2242
Name: label, dtype: int64
```

In the train dataset, we have 2,242 (~7%) tweets labeled as racist or sexist, and 29,720 (~93%) tweets labeled as non racist/sexist. So, it is an imbalanced classification challenge.

checking the distribution of length of the tweets, in terms of words, in both train and test data.

```
In [10]: length_train = train['tweet'].str.len()
         length_test = test['tweet'].str.len()

In [11]: plt.hist(length_train, bins=20, label="train_tweets")
         plt.hist(length_test, bins=20, label="test_tweets")
         plt.legend()
         plt.show()
```



Data Cleaning

Our objective of opting this step is to clean noise those are less relevant to find the sentiment of tweets such as punctuation, special characters, numbers, and terms which don't carry much weightage in context to the text.

first we would combine train and test datasets. Combining the datasets will make it convenient for us to preprocess the data. Later we will split it back into train and test data.

```
In [12]: combine = train.append(test, ignore_index=True)
```

```
In [13]: combine.shape
```

```
(49159, 3)
```

Creating a user-defined function to remove unwanted text patterns from the tweets.

```
In [14]: def remove_pattern(input_txt, pattern):
             r = re.findall(pattern, input_txt)
             for i in r:
                 input_txt = re.sub(i, '', input_txt)
             return input_txt
```

Now, Removing (@user)

File Edit View Insert Cell Kernel Widgets Help — Not Trusted — Python 3

```
In [15]: combine['cleaned_tweet'] = np.vectorize(remove_pattern)(combine['tweet'], "@[\w]*")
         combine.head(20)
```

	id	label	tweet	cleaned_tweet
0	1	0.0	@user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run	when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run
1	2	0.0	@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disapointed #getthanked	thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disapointed #getthanked
2	3	0.0	bihday your majesty	bihday your majesty
3	4	0.0	#model i love u take with u all the time in urð±!!! ðððð¦ð¦ð¦	#model i love u take with u all the time in urð±!!! ðððð¦ð¦ð¦
4	5	0.0	factsguide: society now #motivation	factsguide: society now #motivation
5	6	0.0	[2/2] huge fan fare and big talking before they leave. chaos and pay disputes when they get there. #allshowandnogo	[2/2] huge fan fare and big talking before they leave. chaos and pay disputes when they get there. #allshowandnogo
6	7	0.0	@user camping tomorrow @user @user @user @user @user @user @user dannyâ¦	camping tomorrow dannyâ¦
7	8	0.0	the next school year is the year for exams.ð¯ can't think about that ð #school #exams #hate #imagine #actorslife #revolutionschool #girl	the next school year is the year for exams.ð¯ can't think about that ð #school #exams #hate #imagine #actorslife #revolutionschool #girl
8	9	0.0	we won!!! love the land!!! #allin #cavs #champions #cleveland #clevelandcavaliers â¦	we won!!! love the land!!! #allin #cavs #champions #cleveland #clevelandcavaliers â¦
9	10	0.0	@user @user welcome here ! i'm it's so #gr8 !	welcome here ! i'm it's so #gr8 !
10	11	0.0	â #ireland consumer price index (mom) climbed from previous 0.2% to 0.5% in may #blog #silver #gold #forex	â #ireland consumer price index (mom) climbed from previous 0.2% to 0.5% in may #blog #silver #gold #forex
11	12	0.0	we are so selfish. #orlando #standwithorlando #pulseshooting #orlandoshooting #biggerproblems #selfish #heabreaking #values #love #	we are so selfish. #orlando #standwithorlando #pulseshooting #orlandoshooting #biggerproblems #selfish #heabreaking #values #love #
12	13	0.0	i get to see my daddy today!! #80days #gettingfed	i get to see my daddy today!! #80days #gettingfed
13	14	1.0	@user #cnn calls #michigan middle school 'build the wall' chant " #tcot	#cnn calls #michigan middle school 'build the wall' chant " #tcot
14	15	1.0	no comment! in #australia #opkillingbay #seashepherd #helpcovedolphins #thecove #helpcovedolphins	no comment! in #australia #opkillingbay #seashepherd #helpcovedolphins #thecove #helpcovedolphins
15	16	0.0	ouch...junior is angryð#got7 #junior #yugyoem #omg	ouch...junior is angryð#got7 #junior #yugyoem #omg
16	17	0.0	i am thankful for having a paner. #thankful #positive	i am thankful for having a paner. #thankful #positive
17	18	1.0	retweet if you agree!	retweet if you agree!
18	19	0.0	its #friday! ð smiles all around via ig user: @user #cookies make people	its #friday! ð smiles all around via ig user: #cookies make people
19	20	0.0	as we all know, essential oils are not made of chemicals.	as we all know, essential oils are not made of chemicals.

Removing (**Punctuations, Numbers, and Special Characters**)

Except alphabets and '#'

File Edit View Insert Cell Kernel Widgets Help   Not Trusted   Python 3

```
In [16]: combine['cleaned_tweet'] = combine['cleaned_tweet'].str.replace("[^a-zA-Z#]", " ")
         combine.head(20)
```

```
<ipython-input-16-3bd148e24999>:1: FutureWarning: The default value of regex will change from True to False in a future version.
  combine['cleaned_tweet'] = combine['cleaned_tweet'].str.replace("[^a-zA-Z#]", " ")
```

	id	label	tweet	cleaned_tweet
0	1	0.0	@user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run	when a father is dysfunctional and is so selfish he drags his kids into his dysfunction #run
1	2	0.0	@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disapointed #getthanked	thanks for #lyft credit i can t use cause they don t offer wheelchair vans in pdx #disapointed #getthanked
2	3	0.0	bihday your majesty	bihday your majesty
3	4	0.0	#model i love u take with u all the time in urð±!!! ðððð¦ð¦ð¦	#model i love u take with u all the time in ur
4	5	0.0	factsguide: society now #motivation	factsguide society now #motivation
5	6	0.0	[2/2] huge fan fare and big talking before they leave. chaos and pay disputes when they get there. #allshowandnogo	huge fan fare and big talking before they leave chaos and pay disputes when they get there #allshowandnogo
6	7	0.0	@user camping tomorrow @user @user @user @user @user @user @user dannyâ¦	camping tomorrow danny
7	8	0.0	the next school year is the year for exams.ð¯ can't think about that ð- #school #exams #hate #imagine #actorslife #revolutionschool #girl	the next school year is the year for exams can t think about that #school #exams #hate #imagine #actorslife #revolutionschool #girl
8	9	0.0	we won!!! love the land!!! #allin #cavs #champions #cleveland #clevelandcavaliers â¦	we won love the land #allin #cavs #champions #cleveland #clevelandcavaliers
9	10	0.0	@user @user welcome here ! i'm it's so #gr8 !	welcome here i m it s so #gr
10	11	0.0	â #ireland consumer price index (mom) climbed from previous 0.2% to 0.5% in may #blog #silver #gold #forex	#ireland consumer price index mom climbed from previous to in may #blog #silver #gold #forex
11	12	0.0	we are so selfish. #orlando #standwithorlando #pulseshooting #orlandoshooting #biggerproblems #selfish #heabreaking #values #love #	we are so selfish #orlando #standwithorlando #pulseshooting #orlandoshooting #biggerproblems #selfish #heabreaking #values #love #
12	13	0.0	i get to see my daddy today!! #80days #gettingfed	i get to see my daddy today # days #gettingfed
13	14	1.0	@user #cnn calls #michigan middle school 'build the wall' chant " #tcot	#cnn calls #michigan middle school build the wall chant #tcot
14	15	1.0	no comment! in #australia #opkillingbay #seashepherd #helpcovedolphins #thecove #helpcovedolphins	no comment in #australia #opkillingbay #seashepherd #helpcovedolphins #thecove #helpcovedolphins
15	16	0.0	ouch...junior is angryð#got7 #junior #yugyoem #omg	ouch junior is angry #got #junior #yugyoem #omg
16	17	0.0	i am thankful for having a paner. #thankful #positive	i am thankful for having a paner #thankful #positive
17	18	1.0	retweet if you agree!	retweet if you agree

Removing Short Words

Words which are less then or equals to 3 characters

```
In [19]: combine['cleaned_tweet'] = combine['cleaned_tweet'].apply(lambda x: ' '.join([w for w in x.spli
```

```
In [20]: combine.head(5)
```

	id	label	tweet	cleaned_tweet
0	1	0.0	@user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run	when father dysfunctional selfish drags kids into dysfunction #run
1	2	0.0	@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disapointed #getthanked	thanks #lyft credit cause they offer wheelchair vans #disapointed #getthanked
2	3	0.0	bihday your majesty	bihday your majesty
3	4	0.0	#model i love u take with u all the time in urð±!!! ðððð¦ð¦ð¦	#model love take with time
4	5	0.0	factsguide: society now #motivation	factsguide society #motivation

Text Normalization and Tokenization

we will use nltk's PorterStemmer() function to normalize the tweets. But before that we will have to tokenize the tweets. Tokens are individual terms or words, and tokenization is the process of splitting a string of text into tokens.

```
In [19]: tokenized_tweet = combine['cleaned_tweet'].apply(lambda x: x.split())
         tokenized_tweet.head()

0                 [when, father, dysfunctional, selfish, drags, kids, into, dysfunction, #run]
1    [thanks, #lyft, credit, cause, they, offer, wheelchair, vans, #disapointed, #getthanked]
2                                                                   [bihday, your, majesty]
3                                                          [#model, love, take, with, time]
4                                                        [factsguide, society, #motivation]
Name: cleaned_tweet, dtype: object
```

Now we can normalize the tokenized tweets and also stitching these tokens back together. It can easily be done using nltk's MosesDetokenizer function.

```
In [20]: stemmer = PorterStemmer()
         tokenized_tweet = tokenized_tweet.apply(lambda x: [stemmer.stem(i) for i in x])

In [21]: for i in range(len(tokenized_tweet)):
             tokenized_tweet[i] = ' '.join(tokenized_tweet[i])
         combine['tidy_tweet'] = tokenized_tweet
```

Visualization from the Tweets

Understanding the common words used in the tweets using WordCloud:

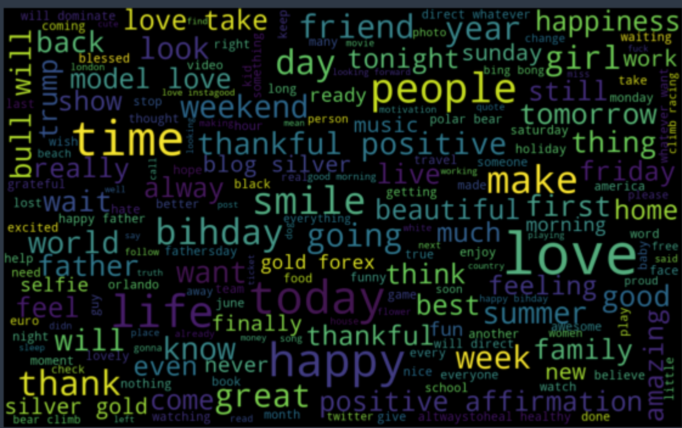
A wordcloud is a visualization wherein the most frequent words appear in large size and the less frequent words appear in smaller sizes.

visualizing all the words our data using the wordcloud plot.

```
In [22]: all_words = ' '.join([text for text in combine['cleaned_tweet']])
```

```
In [23]: wordcloud = WordCloud(width=1300, height=800, random_state=21, max_font_size=110).generate(all_
```

```
In [24]: plt.figure(figsize=(10, 7))
         plt.imshow(wordcloud, interpolation="bilinear")
         plt.axis('off')
         plt.show()
```



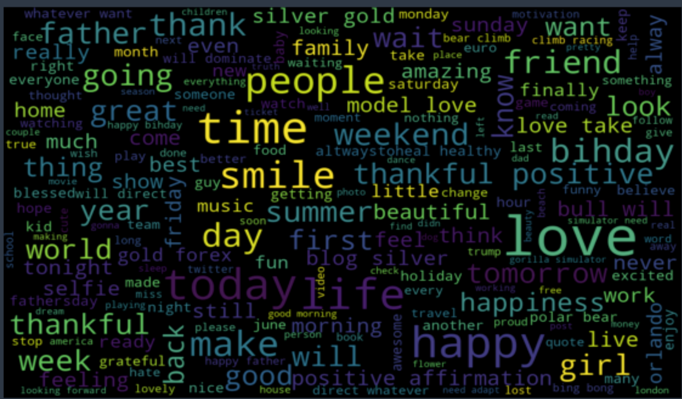
We can see most of the words are positive or neutral. Words like love, great, friend, life are the most frequent ones. It doesn't give us any idea about the words associated with the racist/sexist tweets. Hence, we will plot separate wordclouds for both the classes (racist/sexist or not) in our train data.

Words in non racist/sexist tweets

```
In [25]: normal_words =' '.join([text for text in combine['cleaned_tweet'][combine['label'] == 0]])
```

```
In [26]: wordcloud = WordCloud(width=1300, height=800, random_state=21, max_font_size=110).generate(norm
```

```
In [27]: plt.figure(figsize=(10, 7))
         plt.imshow(wordcloud, interpolation="bilinear")
         plt.axis('off')
         plt.show()
```



Most of the frequent words are compatible with the sentiment, i.e, non-racist/sexists tweets. Similarly, we will plot the word cloud for the other sentiment. Expect to see negative, racist, and sexist terms.

Racist/Sexist Tweets

```
In [28]: negative_words = ' '.join([text for text in combine['cleaned_tweet'][combine['label'] == 1]])
```

```
In [29]: wordcloud = WordCloud(width=1300, height=800, random_state=21, max_font_size=110).generate(nega
```

```
In [30]: plt.figure(figsize=(10, 7))
         plt.imshow(wordcloud, interpolation="bilinear")
         plt.axis('off')
         plt.show()
```

As we can clearly see, most of the words have negative connotations. So, it seems we have a pretty good text data to work on.

Understanding the impact of Hashtags on tweets sentiment

Hashtags in twitter are synonymous with the ongoing trends on twitter at any particular point in time. We should try to check whether these hashtags add any value to our sentiment analysis task, i.e., they help in distinguishing tweets into the different sentiments.The tweet seems sexist in nature and the hashtags in the tweet convey the same feeling. We will store all the trend terms in two separate lists — one for non-racist/sexist tweets and the other for racist/sexist tweets.

```
In [31]: def hashtag_extract(x):
             hashtags = []
             for i in x:
                 ht = re.findall(r"#(\w+)", i)
                 hashtags.append(ht)
             return hashtags
```

```
In [32]: HT_regular = hashtag_extract(combine['cleaned_tweet'][combine['label'] == 0])
```

```
In [33]: HT_negative = hashtag_extract(combine['cleaned_tweet'][combine['label'] == 1])
```

```
In [34]: HT_regular = sum(HT_regular,[])
         HT_negative = sum(HT_negative,[])
```

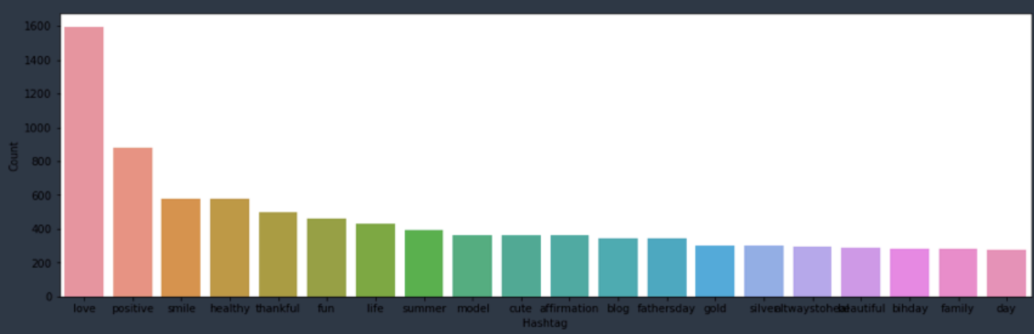
Now that we have prepared our lists of hashtags for both the sentiments, we can plot the top ‘n’ hashtags. So, first let’s check the hashtags in the non-racist/sexist tweets.

Non-Racist/Sexist Tweets

```
In [35]: a = nltk.FreqDist(HT_regular)
         d = pd.DataFrame({'Hashtag': list(a.keys()), 'Count': list(a.values())})
```

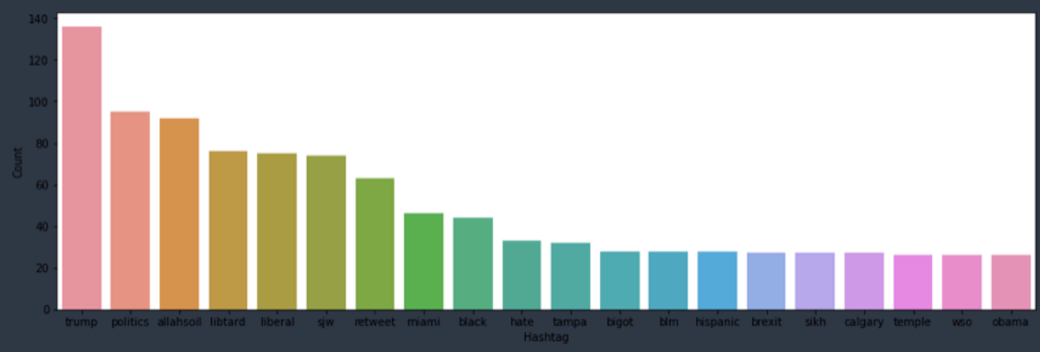
```
In [36]: d = d.nlargest(columns="Count", n = 20)
```

```
In [37]: plt.figure(figsize=(16,5))
         ax = sns.barplot(data=d, x= "Hashtag", y = "Count")
         ax.set(ylabel = 'Count')
         plt.show()
```



Racist/Sexist Tweets

```
In [38]: b = nltk.FreqDist(HT_negative)
         e = pd.DataFrame({'Hashtag': list(b.keys()), 'Count': list(b.values())})

In [39]: e = e.nlargest(columns="Count", n = 20)
         plt.figure(figsize=(16,5))
         ax = sns.barplot(data=e, x= "Hashtag", y = "Count")
```



As expected, most of the terms are negative with a few neutral terms as well. So, it's not a bad idea to keep these hashtags in our data as they contain useful information.

Techniques used:-

Bag-of-Words Features

```
In [40]: bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000, stop_words='english'
         bow = bow_vectorizer.fit_transform(combine['cleaned_tweet'])
         bow.shape

(49159, 1000)
```

TF-IDF Feature

```
In [41]: tfidf_vectorizer = TfidfVectorizer(max_df=0.90, min_df=2, max_features=1000, stop_words='englis

In [42]: tfidf = tfidf_vectorizer.fit_transform(combine['cleaned_tweet'])

In [43]: tfidf.shape

(49159, 1000)
```

Word2Vec Feature

```
In [44]: tokenized_tweet = combine['cleaned_tweet'].apply(lambda x: x.split())

In [45]: model_w2v = gensim.models.Word2Vec(tokenized_tweet, vector_size = 200, window=5, min_count=2, s

In [46]: model_w2v.train(tokenized_tweet, total_examples= len(combine['cleaned_tweet']), epochs=20)

(6479298, 7536020)

In [47]: model_w2v.wv.most_similar(positive="dinner")

[('lamb', 0.6026326417922974),
 ('burritos', 0.5883433818817139),
 ('spaghetti', 0.5874906778335571),
 ('desse', 0.5735955834388733),
 ('noodle', 0.5657437443733215),
 ('#toast', 0.559001624584198),
 ('#avocado', 0.5525112748146057),
 ('enroute', 0.5486379861831665),
 ('alfredo', 0.545955002307891 8),
 ('melanie', 0.5449588298797607)]
```

Preparing Vectors for Tweets

```
In [48]: model_w2v.wv.most_similar(positive="boys")
```

```
[('firearm', 0.5404597520828247),
 ('#firstgame', 0.5318750739097595),
 ('#tweetfromyourseat', 0.5254544019699097),
 ('#threelions', 0.4774954617023468),
 ('moro', 0.4721876382827759),
 ('rugby', 0.4710134267807007),
 ('#gerpol', 0.4652010798454284 7),
 ('elevated', 0.4600367844104767),
 ('mclain', 0.4575196802616 1194),
 ('moaning', 0.4530889689922333)]
```

```
In [49]: model_w2v.wv.most_similar(positive="trump")
```

```
[('hillary', 0.5472602844238281),
 ('unfavorability', 0.5405436754226685),
 ('commie', 0.5384860038757324),
 ('donald', 0.5333816409111023),
 ('chopra', 0.5253068208694458),
 ('#delegaterevolt', 0.5242677330970764),
 ('phony', 0.5229575634002686),
 ('endorses', 0.5113083720207214),
 ('truism', 0.5096306800842285),
 ('battered', 0.5095114707946777)]
```

```
In [52]: def word_vector(tokens, size):
             vec = np.zeros(size).reshape((1, size))
             count = 0.
             for word in tokens:
                 try:
                     vec += model_w2v.wv[word].reshape((1, size))
                     count += 1.
                 except KeyError:
                     continue
             if count != 0:
                 vec /= count
             return vec
```

```
In [53]: wordvec_arrays = np.zeros((len(tokenized_tweet), 200))
         for i in range(len(tokenized_tweet)):
             wordvec_arrays[i,:] = word_vector(tokenized_tweet[i], 200)
             wordvec_df = pd.DataFrame(wordvec_arrays)
             wordvec_df.shape
```

```
In [54]: def add_label(twt):
             output = []
             for i, s in zip(twt.index, twt):
                 output.append(TaggedDocument(s, ["tweet_" + str(i)]))
             return output
```

```
In [55]: TaggedDocument = add_label(tokenized_tweet)
```

```
In [57]: TaggedDocument[:6]
```

```
[TaggedDocument(words=['when', 'father', 'dysfunctional', 'selfish', 'drags', 'kids', 'into', 'dysfunction', '#run'], tags=['twee
t_0']),
 TaggedDocument(words=['thanks', '#lyft', 'credit', 'cause', 'they', 'offer', 'wheelchair', 'vans', '#disapointed', '#getthanked'
], tags=['tweet_1']),
 TaggedDocument(words=['bihday', 'your', 'majesty'], tags=['tweet_2']),
 TaggedDocument(words=['#model', 'love', 'take', 'with', 'time'], tags=['tweet_3']),
 TaggedDocument(words=['factsguide', 'society', '#motivation'], tags=['tweet_4']),
 TaggedDocument(words=['huge', 'fare', 'talking', 'before', 'they', 'leave', 'chaos', 'disputes', 'when', 'they', 'there', '#alls
howandnogo'], tags=['tweet_5'])]
```

training a **doc2vec** model.

```
In [59]: model_d2v = gensim.models.Doc2Vec(dm=1, dm_mean=1, vector_size=200, window=5,
         negative=7, min_count=5, workers=3, alpha=0.1, seed = 23)
```

```
In [61]: model_d2v.build_vocab([i for i in tqdm(TaggedDocument)])
         model_d2v.train(TaggedDocument, total_examples= len(combine['tidy_tweet']), epochs=15)

         100%|██████████| 49159/49159 [00:00<00:00, 1192402.12it/s]
```

Preparing doc2vec Feature Set

```
In [62]: docvec_arrays = np.zeros((len(tokenized_tweet), 200))
         for i in range(len(combine)):
             docvec_arrays[i,:] = model_d2v.docvecs[i].reshape((1,200))

         docvec_df = pd.DataFrame(docvec_arrays)
         docvec_df.shape

         (49159, 200)
```

Evaluation Metric

F1 score is being used as the evaluation metric. It is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. It is suitable for uneven class distribution problems. The important components of F1 score are:

1. True Positives (TP) - These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes.
2. True Negatives (TN) - These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no.
3. False Positives (FP) – When actual class is no and predicted class is yes.
4. False Negatives (FN) – When actual class is yes but predicted class in no.

Precision = TP/TP+FP

Recall = TP/TP+FN

F1 Score = 2(*Recall* Precision) / (Recall + Precision)

Logistic Regression

Logistic Regression is a classification algorithm. It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function.

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta(Age)$$

Bag-of-Words Features

```
In [67]: train_bow = bow[:31962,:]
         test_bow = bow[31962:,:]

In [69]: xtrain_bow,xvalid_bow, ytrain, yvalid = train_test_split(train_bow, train['label'], random_stat

In [71]: lreg = LogisticRegression()
         lreg.fit(xtrain_bow, ytrain)

         LogisticRegression()

In [73]: prediction = lreg.predict_proba(xvalid_bow)

In [74]: prediction_int = prediction[:,1] >= 0.3

In [75]: prediction_int = prediction_int.astype(np.int)

In [76]: f1_score(yvalid, prediction_int)

         0.5017421602787456
```

making predictions for the test dataset and create a submission file.

```
In [67]: test_pred = lreg.predict_proba(test_bow)
         test_pred_int = test_pred[:,1] >= 0.3
         test_pred_int = test_pred_int.astype(np.int)
         test['label'] = test_pred_int
         submission = test[['id','label']]
         submission.to_csv('sub_lreg_bow.csv', index=False)
```

TF-IDF Features

```
In [80]: train_tfidf = tfidf[:31962,:]
         test_tfidf = tfidf[31962:,:]
         xtrain_tfidf = train_tfidf[ytrain.index]
         xvalid_tfidf = train_tfidf[yvalid.index]
```

```
In [81]: lreg.fit(xtrain_tfidf, ytrain)
         prediction = lreg.predict_proba(xvalid_tfidf)
         prediction_int = prediction[:,1] >= 0.3
         prediction_int = prediction_int.astype(np.int)
         f1_score(yvalid, prediction_int)

         0.5091240875912408
```

Word2Vec Features

```
In [82]: train_w2v = wordvec_df.iloc[:31962,:]
         test_w2v = wordvec_df.iloc[31962:,:]
         xtrain_w2v = train_w2v.iloc[ytrain.index,:]
         xvalid_w2v = train_w2v.iloc[yvalid.index,:]
```

```
In [83]: lreg.fit(xtrain_w2v, ytrain)
         prediction = lreg.predict_proba(xvalid_w2v)
         prediction_int = prediction[:,1] >= 0.3
         prediction_int = prediction_int.astype(np.int)
         f1_score(yvalid, prediction_int)

         0.602247191011236
```

Doc2Vec Features

```
In [84]: train_d2v = docvec_df.iloc[:31962,:]
         test_d2v = docvec_df.iloc[31962:,:]
         xtrain_d2v = train_d2v.iloc[ytrain.index,:]
         xvalid_d2v = train_d2v.iloc[yvalid.index,:]
```

```
In [85]: lreg.fit(xtrain_d2v, ytrain)
         prediction = lreg.predict_proba(xvalid_d2v)
         prediction_int = prediction[:,1] >= 0.3
         prediction_int = prediction_int.astype(np.int)
         f1_score(yvalid, prediction_int)

         0.32857142857142857
```

Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes

Bag-of-Words Features

```
In [87]: svc = svm.SVC(kernel='linear', C=1, probability=True).fit(xtrain_bow, ytrain)
         prediction = svc.predict_proba(xvalid_bow)
         prediction_int = prediction[:,1] >= 0.3
         prediction_int = prediction_int.astype(np.int)
         f1_score(yvalid, prediction_int)

         0.48658649398704906
```

Again making predictions for the test dataset and create another submission file.

```
In [88]: test_pred = svc.predict_proba(test_bow)
         test_pred_int = test_pred[:,1] >= 0.3
         test_pred_int = test_pred_int.astype(np.int)
         test['label'] = test_pred_int
         submission = test[['id','label']]
         submission.to_csv('sub_svm_bow.csv', index=False)
```

TF-IDF Features

```
In [89]: svc = svm.SVC(kernel='linear',
         C=1, probability=True).fit(xtrain_tfidf, ytrain)
         prediction = svc.predict_proba(xvalid_tfidf)
         prediction_int = prediction[:,1] >= 0.3
         prediction_int = prediction_int.astype(np.int)
         f1_score(yvalid, prediction_int)

         0.479108635097493
```

Word2Vec Features

```
In [91]:
svc = svm.SVC(kernel='linear', C=1, probability=True).fit(xtrain_w2v, ytrain)
prediction = svc.predict_proba(xvalid_w2v)
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)
f1_score(yvalid, prediction_int)

0.603273577552611
```

Doc2Vec Features

```
In [92]:
svc = svm.SVC(kernel='linear', C=1, probability=True).fit(xtrain_d2v, ytrain)
prediction = svc.predict_proba(xvalid_d2v)
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)
f1_score(yvalid, prediction_int)

0.12830188679245283
```

RandomForest

Random Forest is a versatile machine learning algorithm capable of performing both regression and classification tasks. It is a kind of ensemble learning method, where a few weak models combine to form a powerful model

Bag-of-Words Features

```
In [68]:
rf = RandomForestClassifier(n_estimators=400, random_state=11).fit(xtrain_bow, ytrain)
prediction = rf.predict(xvalid_bow)
f1_score(yvalid, prediction)

0.5216680294358136
```

Making predictions for the test dataset and create another submission file.

```
In [101]: test_pred = rf.predict(test_bow)
          test['label'] = test_pred
          submission = test[['id','label']]
          submission.to_csv('sub_rf_bow.csv', index=False)
```

TF-IDF Features

```
In [102]: rf = RandomForestClassifier(n_estimators=400, random_state=11).fit(xtrain_tfidf, ytrain)
          prediction = rf.predict(xvalid_tfidf)
          f1_score(yvalid, prediction)

          0.5148698884758364
```

Word2Vec Features

```
In [103]: rf = RandomForestClassifier(n_estimators=400, random_state=11).fit(xtrain_w2v, ytrain)
          prediction = rf.predict(xvalid_w2v)
          f1_score(yvalid, prediction)

          0.5
```

Doc2Vec Features

```
In [104]: rf = RandomForestClassifier(n_estimators=400, random_state=11).fit(xtrain_d2v, ytrain)
          prediction = rf.predict(xvalid_d2v)
          f1_score(yvalid, prediction)

          0.05405405405405406
```

XGBoost

Extreme Gradient Boosting (xgboost) is an advanced implementation of gradient boosting algorithm. It has both linear model solver and tree learning algorithms. Its ability to do parallel computation on a single machine makes it extremely fast. It also has additional features for doing cross validation and finding important variables.

Bag-of-Words Features

```
In [91]: xgb_model = XGBClassifier(max_depth=6, n_estimators=1000, use_label_encoder=False).fit(xtrain_b
         prediction = xgb_model.predict(xvalid_bow)
         f1_score(yvalid, prediction)

[13:19:01] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metri
c used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to re
store the old behavior.

0.5042492917847025
```

Making predictions for the test dataset and create another submission file.

```
In [92]: test_pred = xgb_model.predict(test_bow)
         test['label'] = test_pred
         submission = test[['id','label']]
         submission.to_csv('sub_xgb_bow.csv', index=False)
```

TF-IDF Features

```
In [93]: xgb = XGBClassifier(max_depth=6, n_estimators=1000,use_label_encoder=False).fit(xtrain_tfidf, yt
         prediction = xgb.predict(xvalid_tfidf)
         f1_score(yvalid, prediction)

[13:19:14] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metri
c used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to re
store the old behavior.

0.5027829313543599
```

Word2Vec Features

```
In [87]: xgb = XGBClassifier(max_depth=6, n_estimators=1000, nthread= 3, use_label_encoder=False).fit(xt
         prediction = xgb.predict(xvalid_w2v)
         f1_score(yvalid, prediction)
```

```
[16:38:49] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metri
c used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to re
store the old behavior.

0.6491387126019946
```

Doc2Vec Features

```
In [90]: xgb = XGBClassifier(max_depth=6, n_estimators=1000, nthread= 3, use_label_encoder=False).fit(xt
         prediction = xgb.predict(xvalid_d2v)
         f1_score(yvalid, prediction)
```

```
[17:31:02] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metri
c used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to re
store the old behavior.

0.35280898876404493
```

Summary

initially we cleaned our raw text data, then we learned about 4 different types of feature-set that we can extract from any text data, and finally we used these feature-sets to build models for sentiment analysis. Below is a summary table showing F1 scores for different models and feature-sets.

		Vector-Space			
Model		Bag-Of-Words	TF-IDF	Word2Vec	Doc2Vec
	Logistic Regression	0.501	0.509	0.602	0.328
	SVM	0.486	0.479	0.603	0.128
	RandomForest	0.521	0.514	0.5	0.054
	XGBoost	0.504	0.502	0.649	0.352

Word2Vec features turned out to be most useful. Whereas **XGBoost with Word2Vec features** was the best model for this problem. This clearly shows the power of word embeddings in dealing with NLP problems.

THE END