Sentiment Analysis on Tweets

Objective

- The objective of this task is to detect hate speech in tweets. For the sake of simplicity, we say a tweet contains hate speech if it has a racist or sexist sentiment associated with it. So, the task is to classify racist or sexist tweets from other tweets.

- Formally, given a training sample of tweets and labels, where label '1' denotes the tweet is racist/sexist and label '0' denotes the tweet is not racist/sexist, your objective is to predict the labels on the test dataset. This project is focused on comparing different text representations and learning models on a classification task.

Project Workflow

We will work on a large supervised dataset of English tweet (twitter-hate-speech).We will focus on 4 text representation strategies & 3 classifiers

1. Bag-of-Words (Bow)
2. TF-IDF
3. Word2Vec (w2v)
4. doc2vec (d2v)

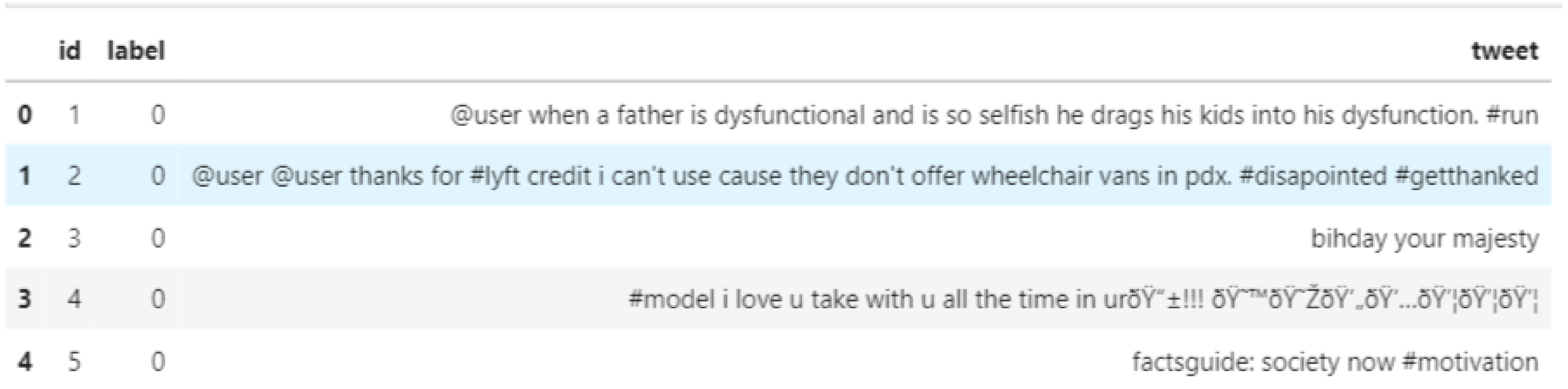
   3 classifiers:

5. Logistic Regression
6. Support Vector Machine
7. XGBoost

(17197, 2))

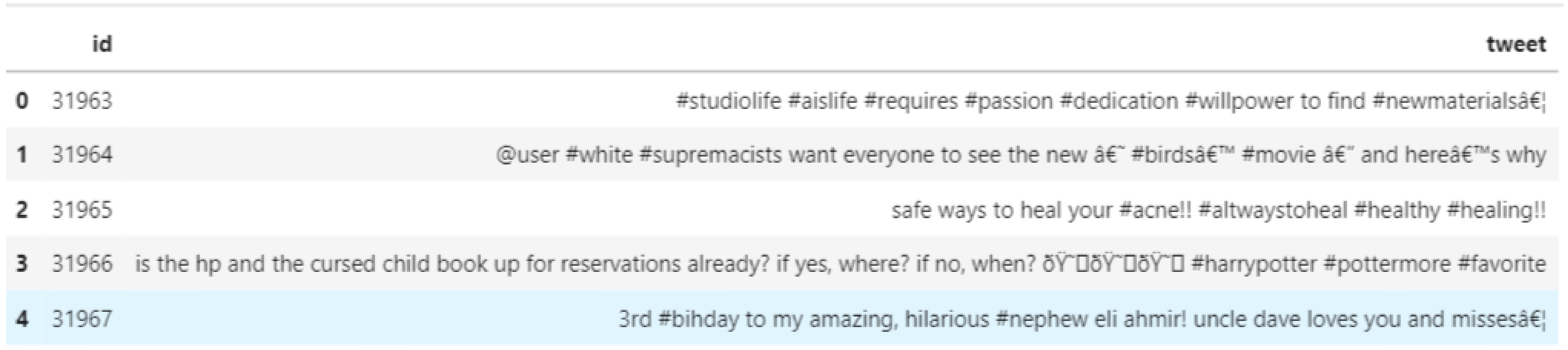
Dataset

- Train data (31972,3)



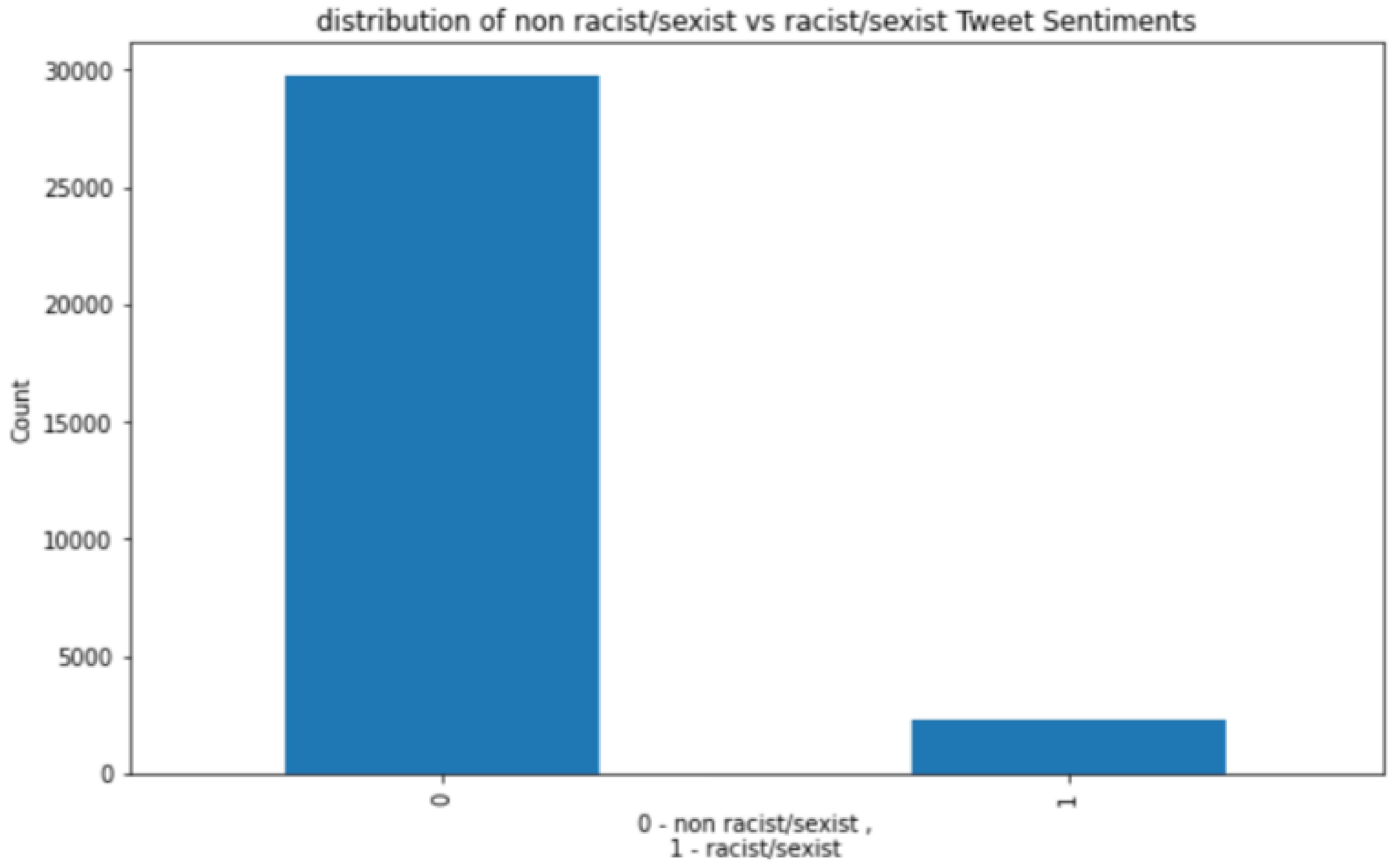
	id	label	tweet
0	1	0	@user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run
1	2	0	@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disapointed #getthanked
2	3	0	bihday your majesty
3	4	0	#model i love u take with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

- Test data (17197,2)



	id	tweet
0	31963	#studiolife #aislife #requires #passion #dedication #willpower to find #newmaterialsâ€¦
1	31964	@user #white #supremacists want everyone to see the new â€˜ #birdsâ€™ #movie â€” and hereâ€™s why
2	31965	safe ways to heal your #acne!! #altwaystoheal #healthy #healing!!
3	31966	is the hp and the cursed child book up for reservations already? if yes, where? if no, when? ðŸ˜‚ðŸ˜‚ðŸ˜‚ #harrypotter #pottermore #favorite
4	31967	3rd #bihday to my amazing, hilarious #nephew eli ahmir! uncle dave loves you and missesâ€¦

Distribution of training data

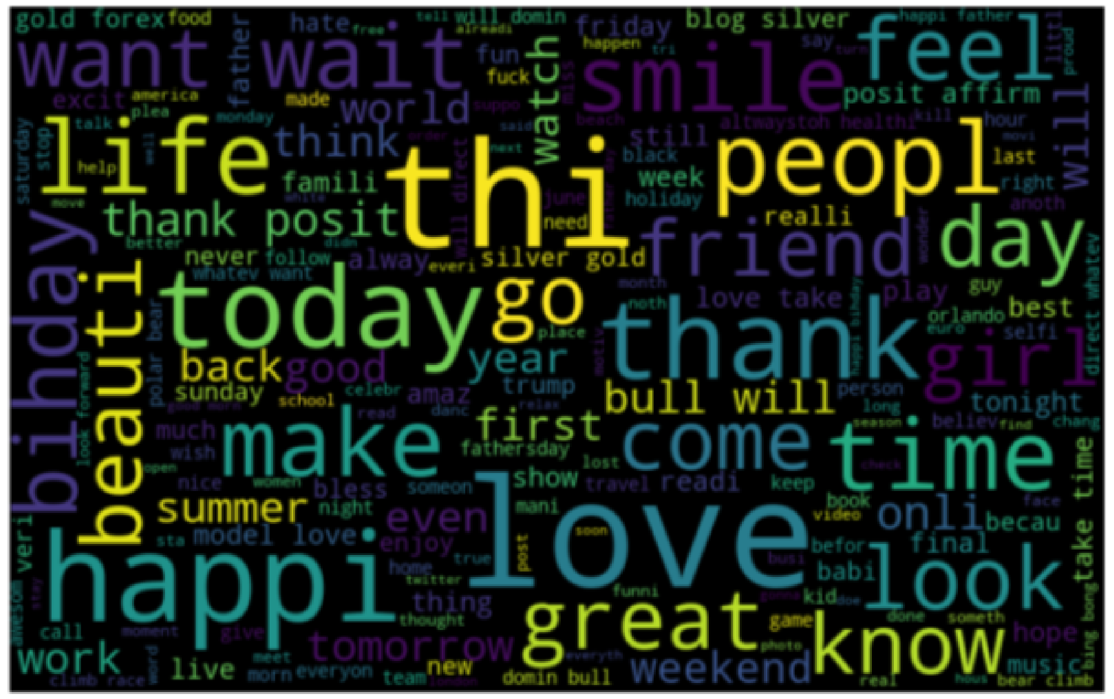


Preprocessing

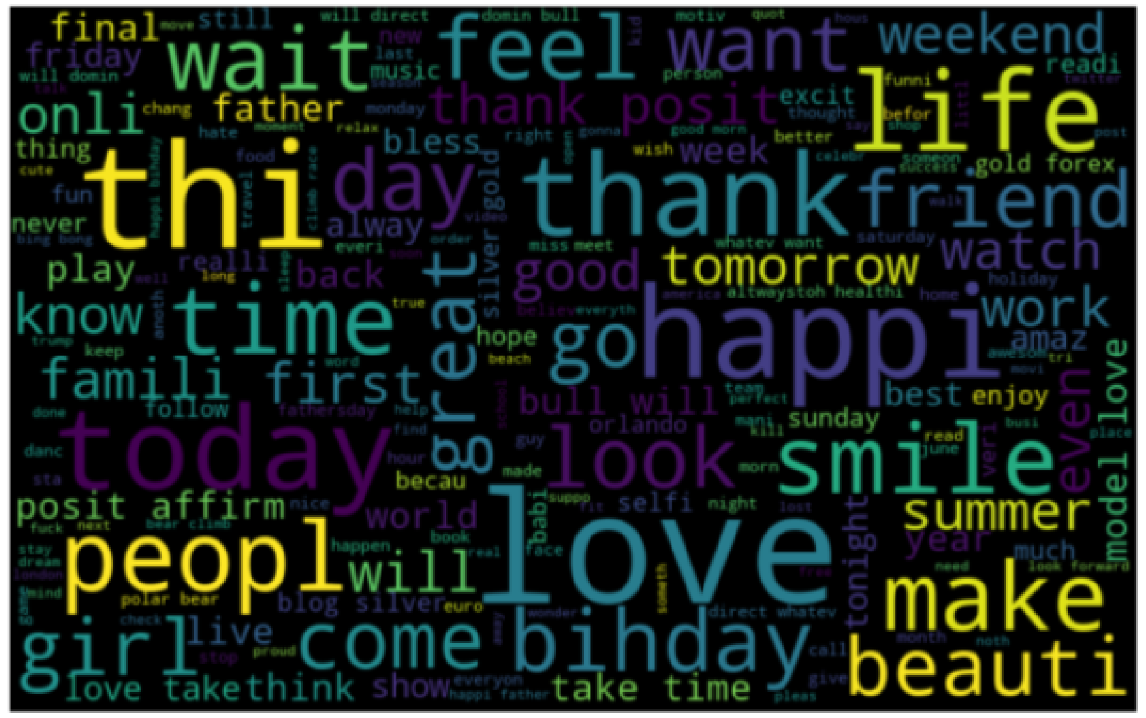
- Remove unwanted words
- Remove Punctuations
- Remove Numbers
- Remove Special Characters
- Remove short/stop words
- Normalizing
- Stemming

Common words

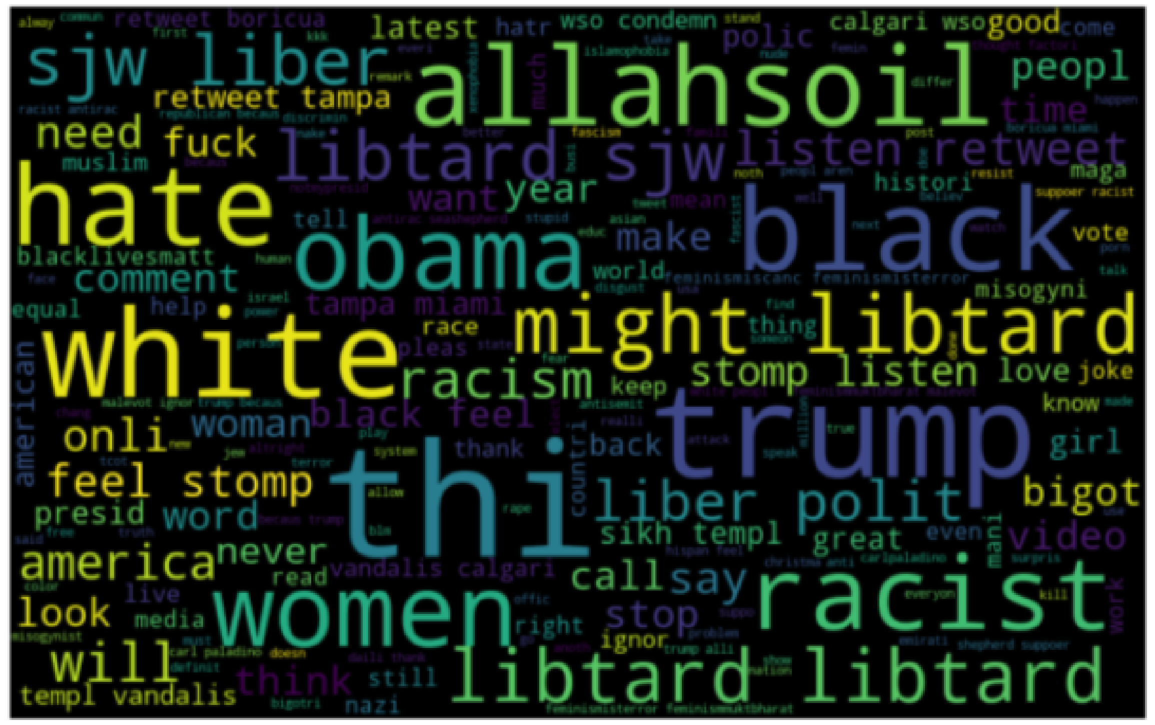
tweets



O label

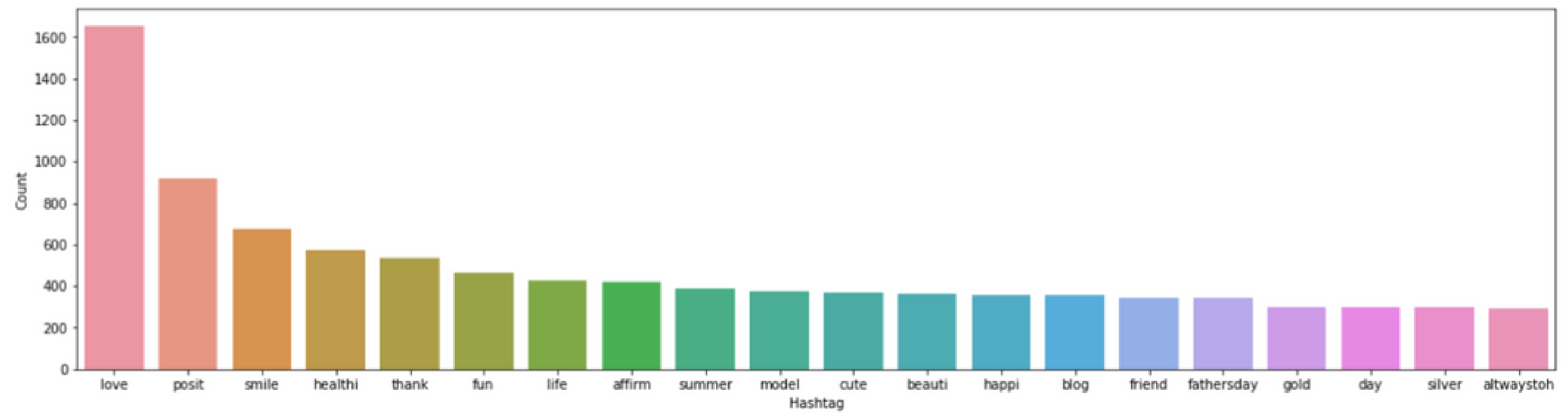


1 label

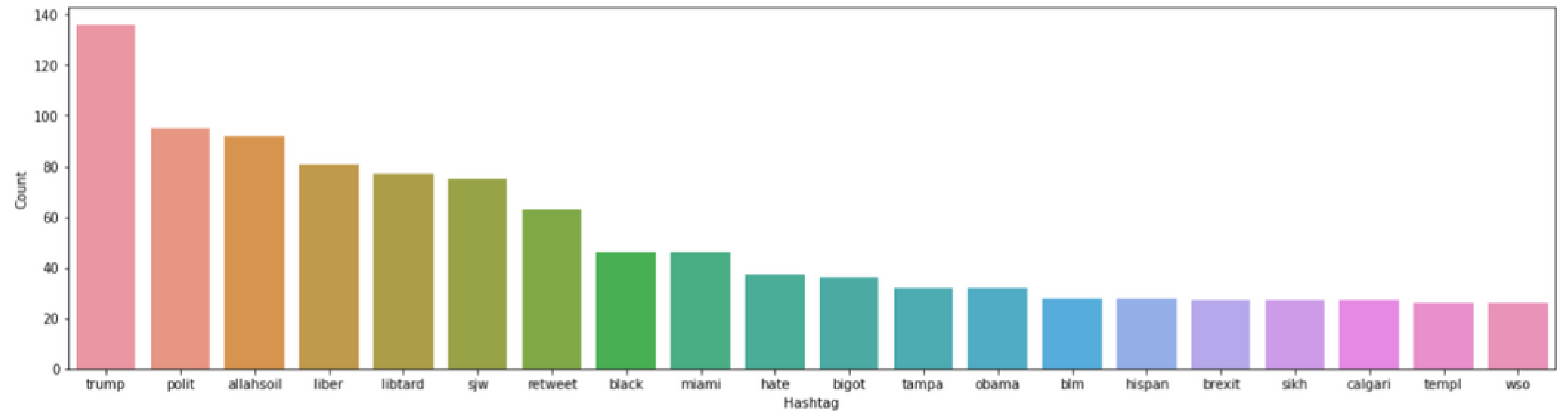


Hashtags in Tweets

0 label

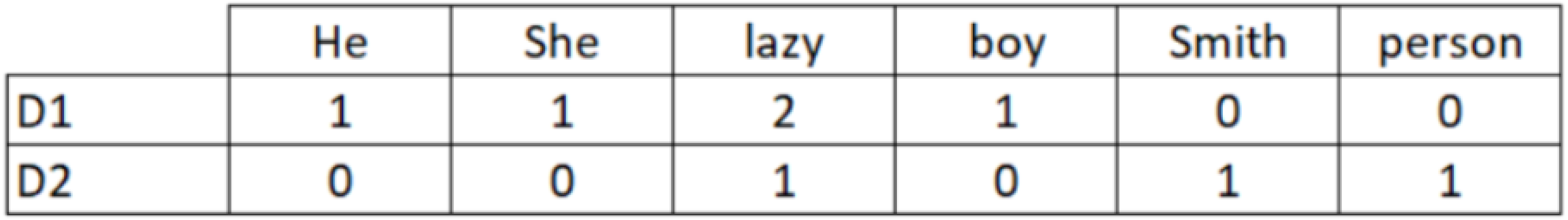


1 label



Bag of Words

- To analyze a preprocessed data, it needs to be converted into features.
- **BOW** is a representation of text that describes the occurrence of words within a document
- Consider a Corpus C of D documents {d1,d2…..dD} and N unique tokens extracted out of the corpus C. The N tokens (words) will form a dictionary and the size of the bag-of-words matrix M will be given by D X N. Each row in the matrix M contains the frequency of tokens in document D(i).
- Let us understand this using a simple example.
- D1: He is a lazy boy. She is also lazy.
- D2: Smith is a lazy person.
- The dictionary created would be a list of unique tokens in the corpus =[‘He','She','lazy','boy','Smith','person']
- Here, D=2, N=6
- The matrix M of size 2 X 6 will be represented as –
- CountVectorizer



	He	She	lazy	boy	Smith	person
D1	1	1	2	1	0	0
D2	0	0	1	0	1	1

TF-IDF

- This is another method which is based on the frequency method but it is different to the bag-of-words approach in the sense that it takes into account not just the occurrence of a word in a single document (or tweet) but in the entire corpus.
- TF-IDF works by penalising the common words by assigning them lower weights while giving importance to words which are rare in the entire corpus but appear in good numbers in few documents.
- Let’s have a look at the important terms related to **TF-IDF**:
- TF = (Number of repetitions of word in a document) / (No of words in a document)
- IDF =Log[(Number of documents) / (Number of documents containing the word)]
- TF-IDF = TF*IDF
- TfidfVectorizer

TF-IDF -2

D1: Good, Boy

D2: Good, Girl

D3: Boy Girl Good

Words	Frequency
Good	3
Boy	2
Girl	2

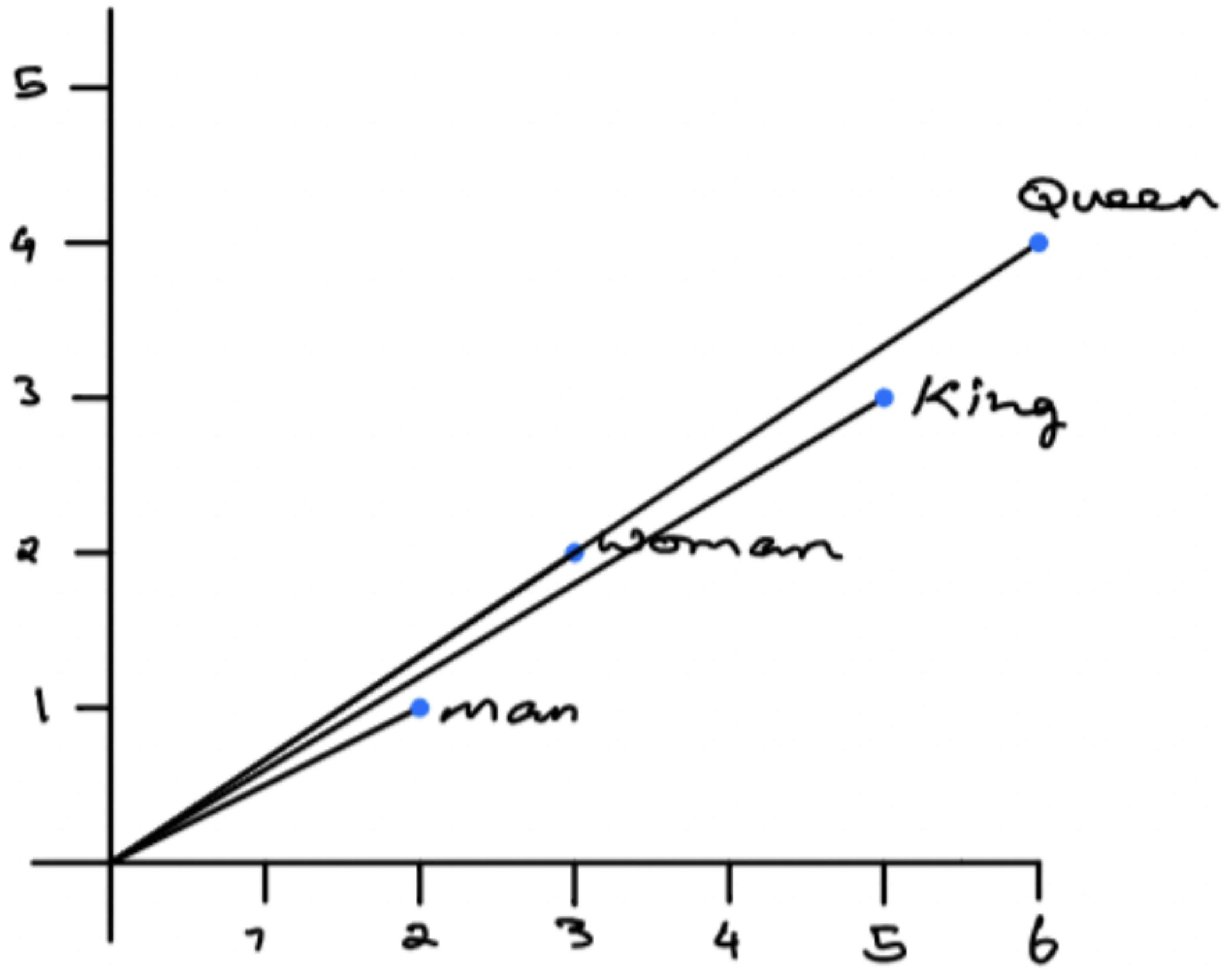
TF			
	D1	D2	D3
Good	1/2	1/2	1/3
Boy	1/2	0	1/3
Girl	0	1/2	1/3

IDF	
Good	log (3/3) =0
Boy	log (3/2)
Girl	log (3/2)

TF* IDF			
	F1	F2	F3
	Good	Boy	Girl
D1	0	½*log (3/2)	0
D2	0	0	½*log (3/2)
D3	0	1/3*log (3/2)	1/3 * log (3/2)

Word2vec

- Word embeddings - Mapping of the words in a vector space
- Word embeddings are the modern way of representing words as vectors. The objective of word embeddings is to redefine the high dimensional word features into low dimensional feature vectors by preserving the contextual similarity in the corpus.
- They are able to achieve tasks like King -man +woman = Queen, which is mind-blowing.
- The Word2Vec model is used to extract the notion of relatedness across words or products such as semantic relatedness, synonym detection, concept categorization, selection preferences, and analogy.
- A Word2Vec model learns meaningful relations and encodes the relatedness into vector similarity.



```
King     -     Man    +    Woman    =     Queen
[5,3]    -     [2,1]  +    [3, 2]   =     [6,4]
```

Word2vec -2

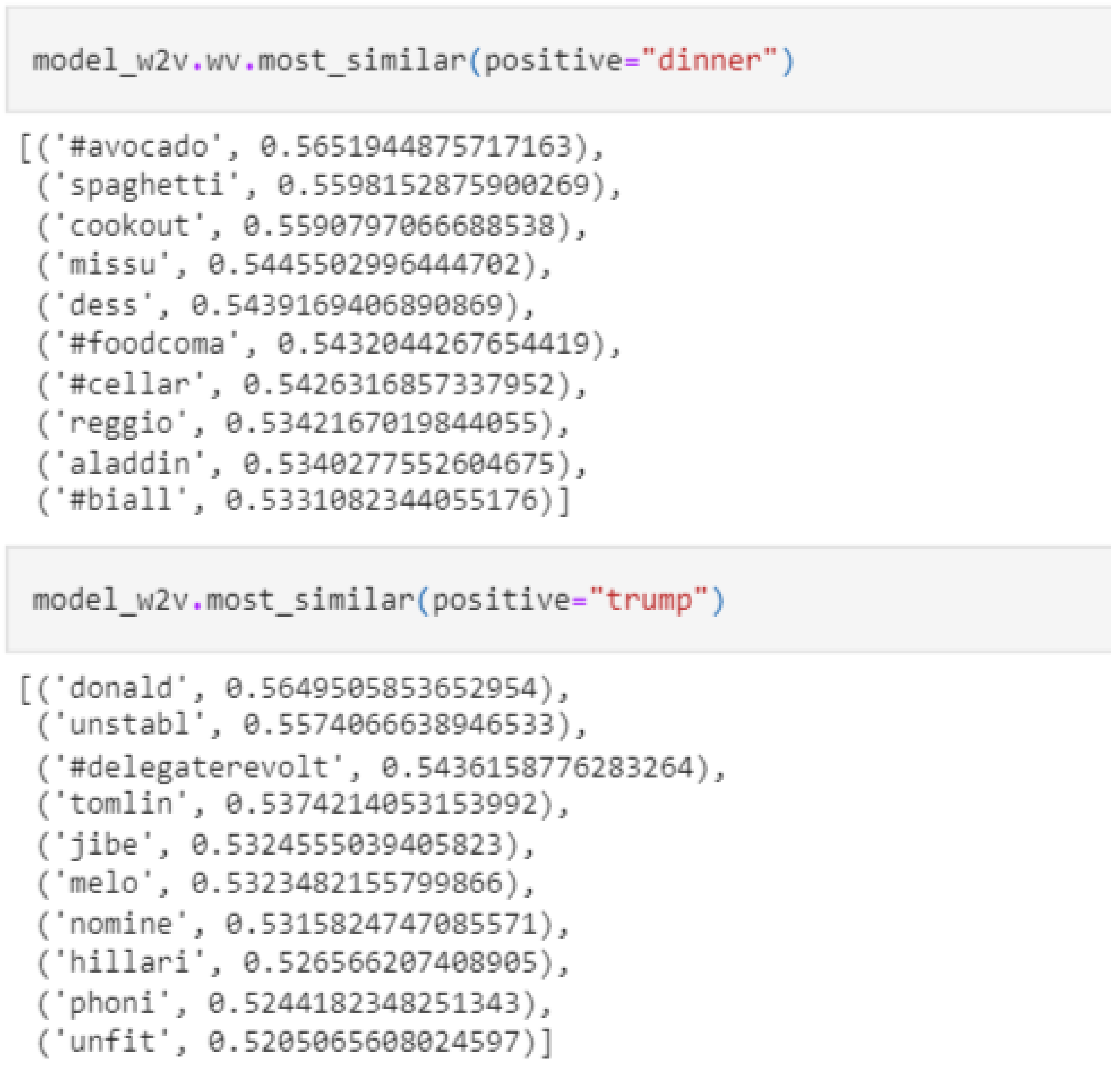
- Word2Vec model takes a text corpus as input and produces the word embedding vectors as output.Word2Vec is not a single algorithm but a combination of two techniques
- CBOW (Continuous bag of words) – predict a word on the basis of its neighbors
- Skip-gram model – predict the neighbors of a word

- Pre-trained model – genism

Word2vec - 3

- The advantages of using word embeddings over BOW or TF-IDF are:
- Dimensionality reduction - significant reduction in the no. of features required to build a model.
- It capture meanings of the words, semantic relationships and the different types of contexts they are used in.
- Deals with addition of new words in the vocabulary
- Better results in ML & DL applications

Word2vec - 4



```
model_w2v.wv.most_similar(positive="dinner")
```

```
[('#avocado', 0.5651944875717163),
 ('spaghetti', 0.5598152875900269),
 ('cookout', 0.5590797066688538),
 ('missu', 0.544502996444702),
 ('dess', 0.5439169406890869),
 ('#foodcoma', 0.5432044267654419),
 ('#cellar', 0.5426316857337952),
 ('reggio', 0.534216701984055),
 ('aladdin', 0.5340277552604675),
 ('#biall', 0.5331082344055176)]
```

```
model_w2v.most_similar(positive="trump")
```

```
[('donald', 0.5649505853652954),
 ('unstabl', 0.5574066638946533),
 ('#delegaterevolt', 0.5436158776283264),
 ('tomlin', 0.5374214053153992),
 ('jibe', 0.5324555039405823),
 ('melo', 0.5323482155799866),
 ('nomine', 0.5315824747085571),
 ('hillari', 0.526566207408905),
 ('phoni', 0.5244182348251343),
 ('unfit', 0.5205065608024597)]
```

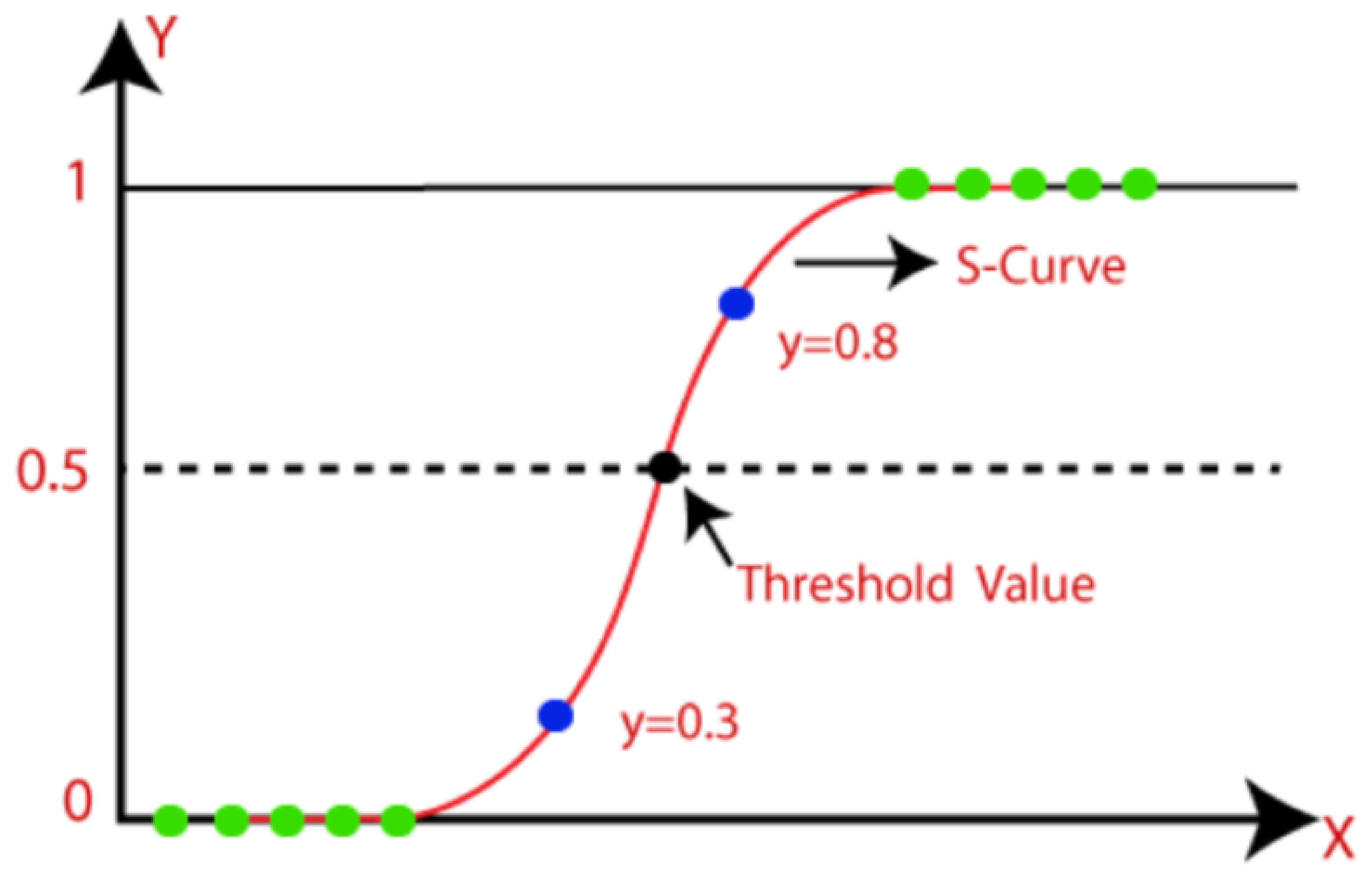
Word2vec does a good job in finding the most similar words in a given word because it has learned vectors for every unique word in our data and it uses cosine similarity to find out the most similar vectors (words).

Doc2vec

- Doc2Vec model is an unsupervised algorithm to generate vectors for sentence/paragraphs/documents.
- This approach is an extension of the word2vec.
- The major difference between the two is that doc2vec provides an additional context which is unique for every document in the corpus. This additional context is nothing but another feature vector for the whole document. This document vector is trained along with the word vectors.
- To implement doc2vec, labelise or tag each tokenised tweet with unique IDs. This was done by using Gensim’s LabeledSentence() function.

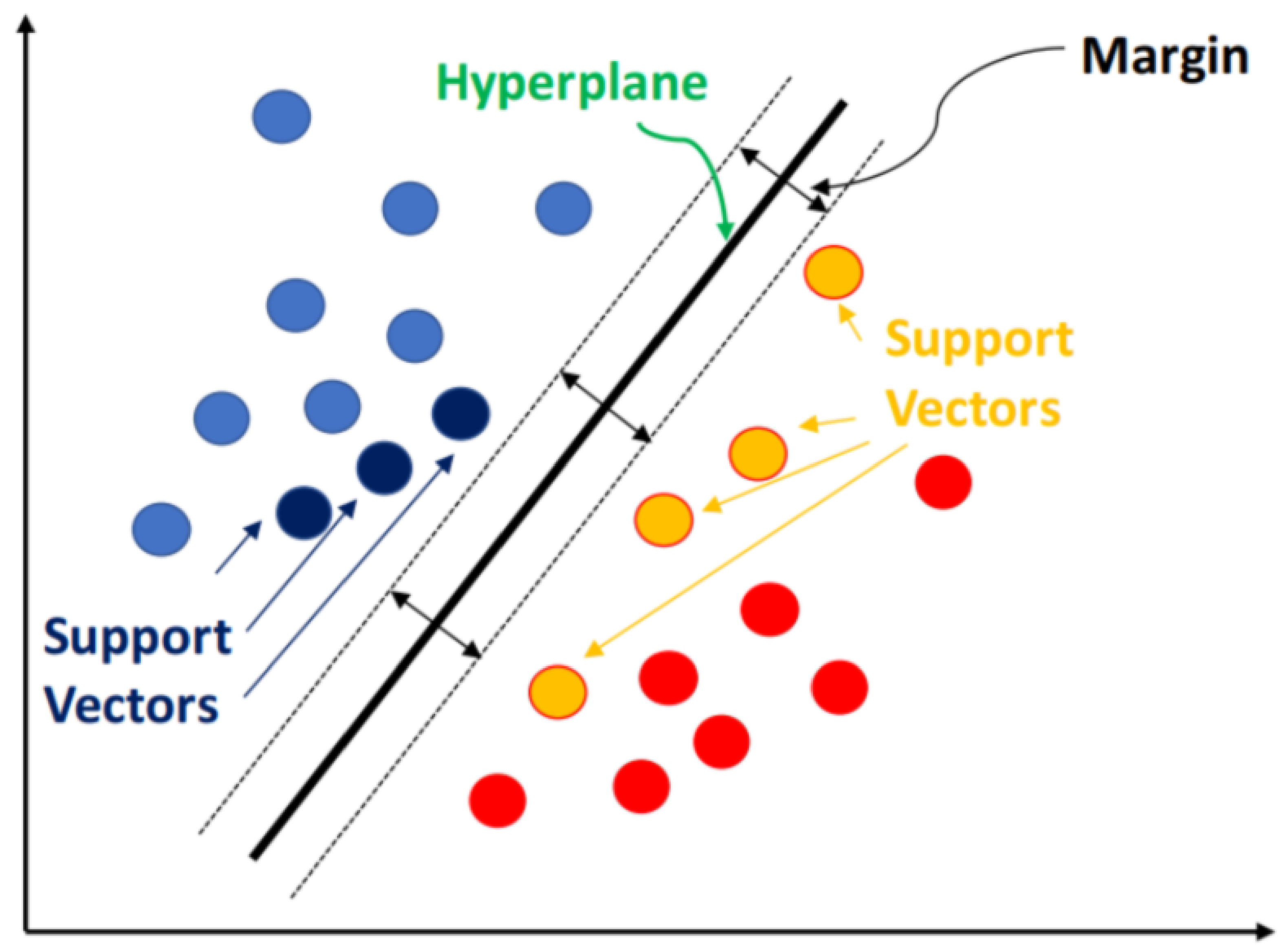
Logistic Regression

- 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.
- Probability never goes below 0 and above 1.



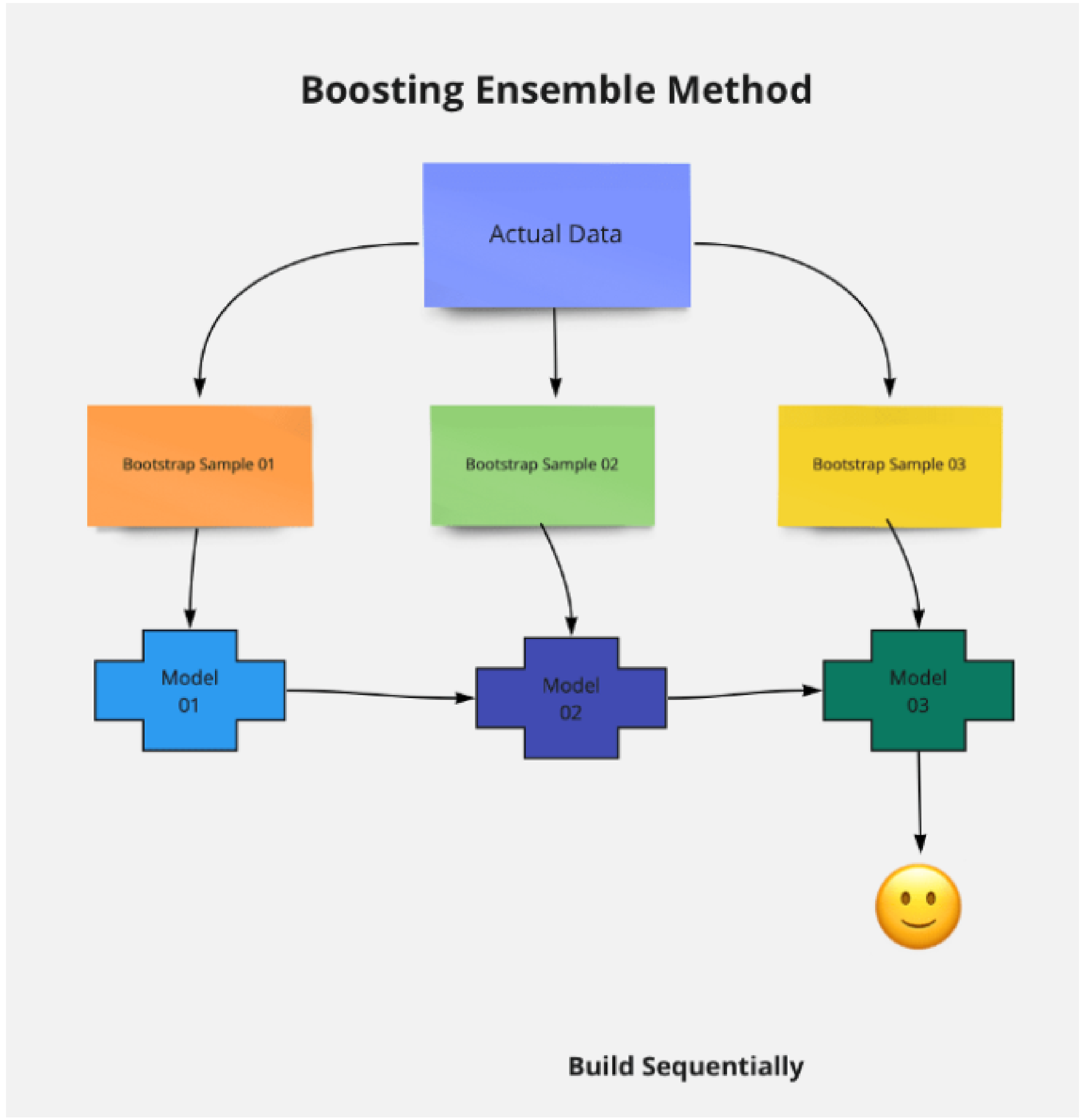
Support Vector Machine

- 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 as shown in the plot below:



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. There are many parameters which need to be controlled to optimize the model.
- Some key benefits of XGBoost are:
- Regularization - helps in reducing overfitting
- Parallel Processing - XGBoost implements parallel processing and is blazingly faster as compared to GBM.
- Handling Missing Values - It has an in-built routine to handle missing values.



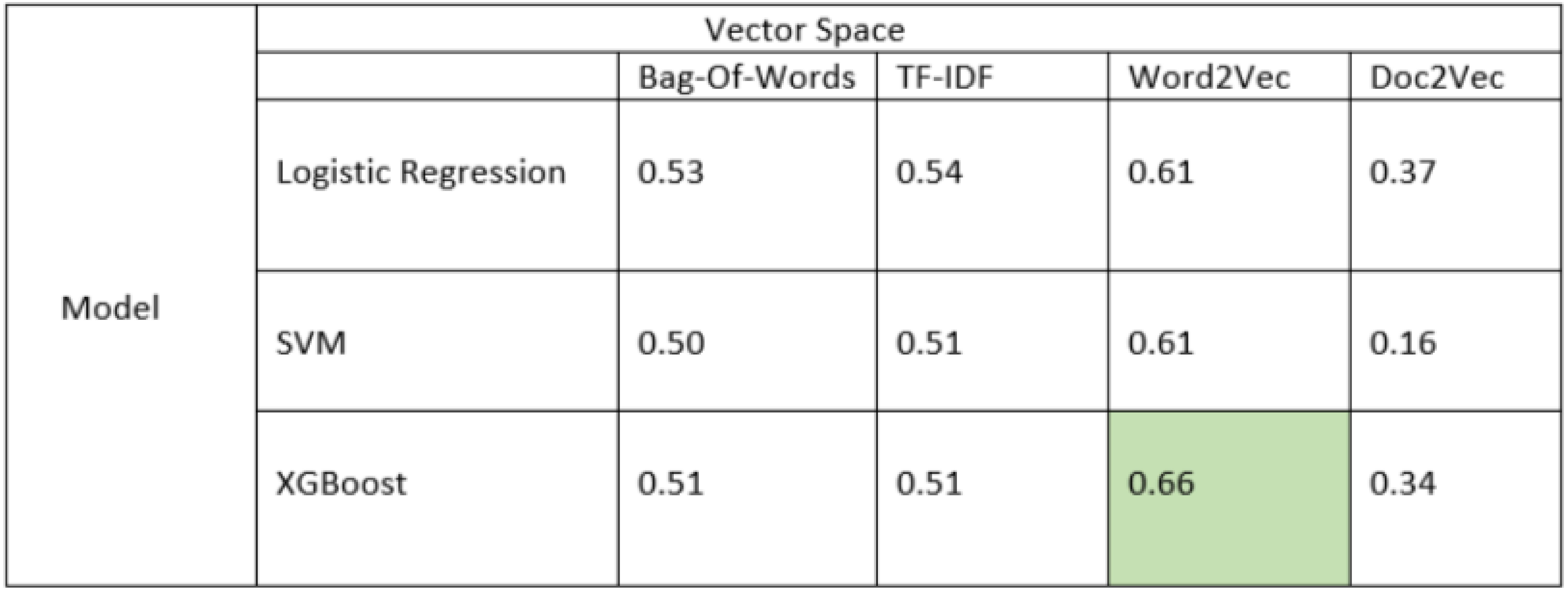
Evaluation Metrics

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)

Validation F1 score



	Vector Space				
		Bag-Of-Words	TF-IDF	Word2Vec	Doc2Vec
	Logistic Regression	0.53	0.54	0.61	0.37
Model	SVM	0.50	0.51	0.61	0.16
	XGBoost	0.51	0.51	0.66	0.34

Hyperparameter Tuning XGBoost + Word2vec

1. learning rate of 0.3
2. Tune tree-specific parameters such as max_depth, min_child_weight, subsample, colsample_bytree keeping the learning rate fixed.
3. Tune the learning rate.
4. Finally tune gamma to avoid overfitting.

Results

- After hyperparameter tuning the model’s performance was improved and the accuracy reached to 68%.

Thank You