

Machine Learning Engineer Nanodegree

Capstone Project

Challa Chaitra

February 20th, 2019

Tweets-Sentiment Analysis

Definition

Project Overview

Sentiment analysis is the process of determining the emotional tone behind a series of words which is used to gain an understanding of the attitudes, opinions and emotions expressed within an online mention. It is extremely useful in social media monitoring as it allows us to gain an overview of the wider public opinion behind certain topics.

Twitter is American online news and social networking site where people communicate in short messages called tweets. It is a micro-blogging site where people write about whatever they want, whether it be politics, sport, cooking, fashion etc. The opinion expressed may be positive, negative or neutral. Extracting this opinion of people from twitter will be of great help to an organization, to a company, to a restaurant, to government etc. in order to upgrade their products, modify the laws, improvise the software etc.

So, here I use process, Sentiment analysis. Twitter Sentiment Analysis may, therefore, be described as a text mining technique for analyzing the underlying sentiment of a text message, i.e., a tweet.

The dataset I used for this project is downloaded from

<https://datahack.analyticsvidhya.com/contest/practice-problem-twitter-sentiment-analysis/>

It has three columns: tweet_id, tweet_label, tweet (message). And I am going to build a model which can classify these tweets correctly.

Problem Statement

The aim of this project is to correctly classify tweets from the dataset as either positive or negative.

I am going to use various machine learning classifiers for this, build models using each of the classifiers and train them on the data. And then select the model which performs best.

So, the tasks involved to get to the solution for this problem are:

- Downloading the dataset
- Visualizing the data.
- Pre-processing the dataset
- Splitting the dataset into train and test sets to train classifier models.
- Choosing the best classifier based on the evaluation metric we have chosen.

Metrics

Accuracy and F1 score are the metrics I used here.

Accuracy: It is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

TP-True Positives

TN-True Negatives

FP-False Positives

FN-False Negatives

True positive and true negatives are the observations that are correctly predicted. False positives and false negatives, these values occur when your actual class contradicts with the predicted class.

Precision: It is the ratio of correctly predicted positive observations to the total predicted positive observations.

$$\text{Precision} = \frac{TP}{TP+FP}$$

Recall: It is the ratio of correctly predicted positive observations to the all positive observations in actual class.

$$\text{Recall} = \text{TP} / \text{TP} + \text{FN}$$

F1 Score: It is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

$$\text{F1 Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

Accuracy works well when the dataset is balanced. But when it comes to imbalanced dataset, F1 score evaluates better. So, in this project, I use these two metrics for evaluation.

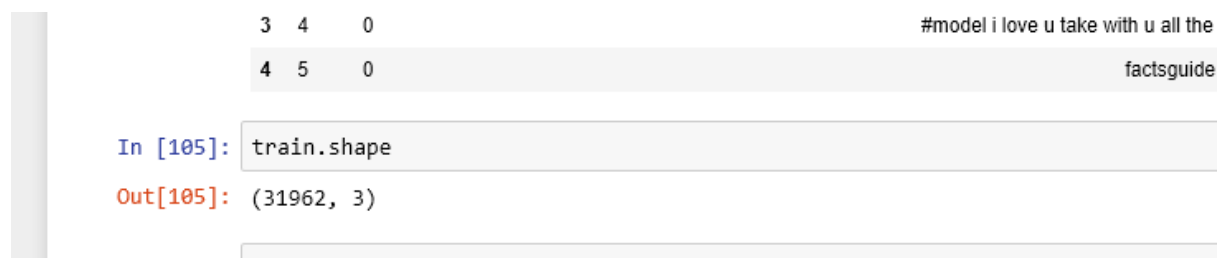
Analysis

Data Exploration and Visualization

The dataset used in this project, sentiment analysis of tweets, is downloaded from

<https://datahack.analyticsvidhya.com/contest/practice-problem-twitter-sentiment-analysis/>

From the below screenshot, it is evident that the dataset has 31962 rows and 3 columns.



The screenshot shows a Jupyter Notebook interface. At the top, there is a snippet of a tweet: "#model i love u take with u all the factsguide". Below this, there is a code cell with the following content:

```
In [105]: train.shape
```

The output of the code cell is displayed below the code:

```
Out[105]: (31962, 3)
```

The dataset consists of 31,962 tweets.

For each instance of the dataset:

Number of Attributes: 3(1 predictive, 1 non-predictive, 1 target)

They are:

Id (non-predictive variable): Tweet id

Label (Target variable): It has 2 categories of values.

Tweet (Predictive variable): It is the text or message which is a mixture of alphanumeric and special characters.

```

In [104]: train.head()

Out[104]:
   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. #disappointed #getthanked
2   3     0                                     bihday your majesty
3   4     0  #model i love u take with u all the time in urð±!!! ððððð\ð\ð!
4   5     0  factsguide: society now #motivation

In [105]: train.shape

Out[105]: (31962, 3)

```

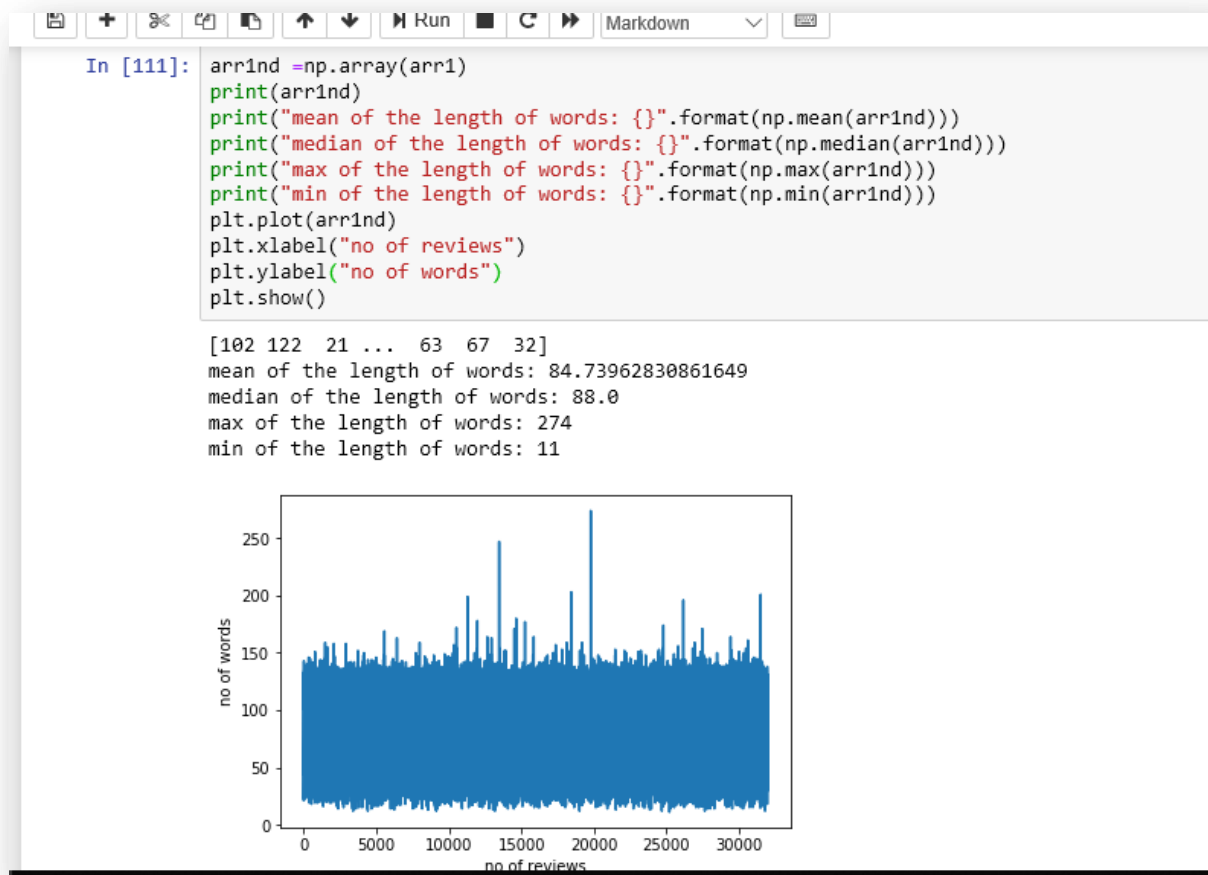
I used head() to display the first few instances of the dataset.



On using the value_counts() on the “label” feature (target variable), we can see that a very large number of tweets are positive which says the dataset is **imbalanced**.

This point is strengthened by result of the countplot() shown above.

And below, we can see the details on the length of tweets. The longest tweet is of length 274 and the shortest one is of 11 characters.



Algorithms and Techniques

The following are the **classification algorithms** I used:

- ✚ Naive bayes
- ✚ KNN
- ✚ Logistic Regression
- ✚ Random Forest
- ✚ Decision tree

Naive Bayes:

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem.

Bayes Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

Types of Naive Bayes Classifier:

1. **Multinomial Naive Bayes:**

This is mostly used for document classification problem i.e., whether a document belongs to the category of sports, politics, technology etc.

2. **Bernoulli Naive Bayes:**

This is similar to the multinomial naive bayes but the predictors are Boolean variables. The parameters that we use to predict the class variable take up only values yes or no.

3. **Gaussian Naive Bayes:**

When the predictors take up a continuous value and are not discrete, we assume that these values are sampled from a Gaussian distribution.

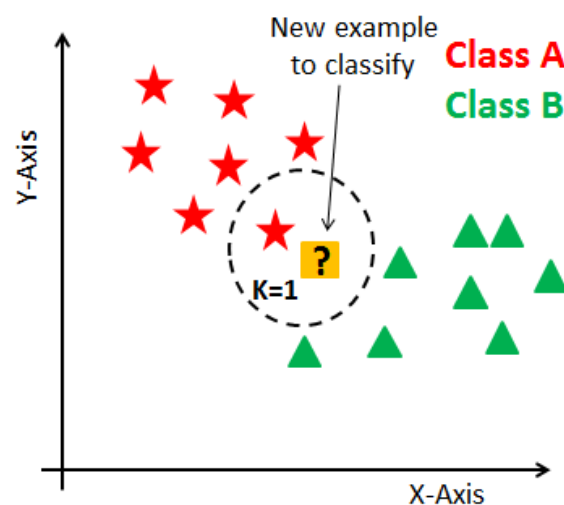
Naive Bayes algorithms are mostly used in **sentiment analysis**, spam filtering, and recommendation systems etc. They are fast and easy to implement but their biggest **disadvantage** is that the requirement of predictors to be independent. In most of the real life cases, the predictors are dependent; this hinders the performance of the classifier.

KNN:

K Nearest Neighbour (KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning algorithms. KNN used in the variety of applications such as finance, healthcare, political science, handwriting detection, image recognition and video recognition.

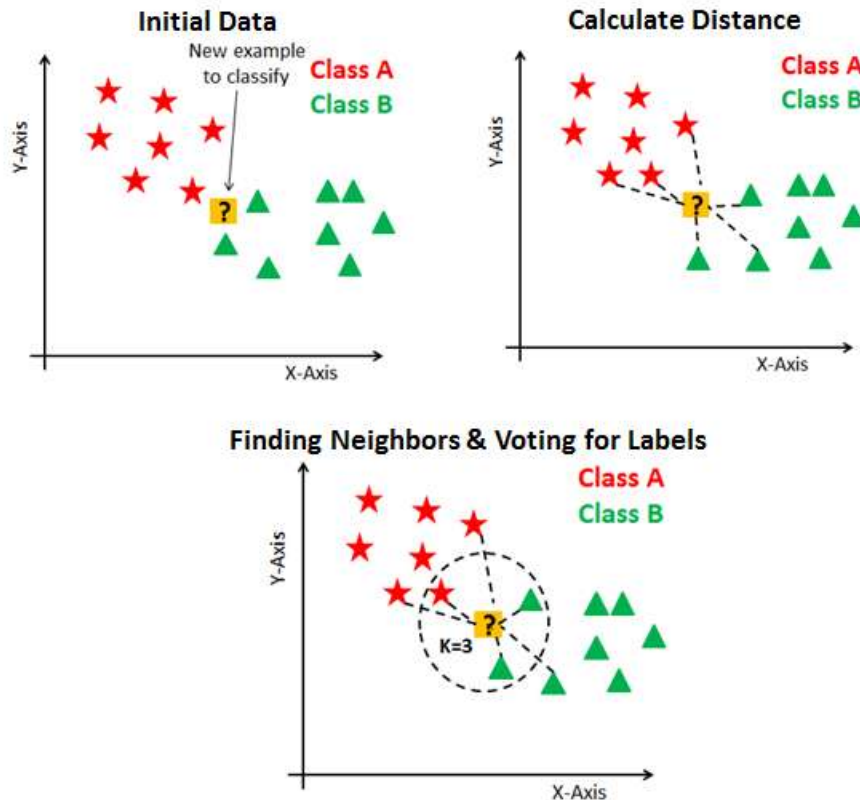
How does the KNN algorithm work?

In KNN, K is the number of nearest neighbours. The number of neighbours is the core deciding factor. K is generally an odd number if the number of classes is 2. When $K=1$, then the algorithm is known as the nearest neighbour algorithm. This is the simplest case. Suppose P1 is the point, for which label needs to predict. First, you find the one closest point to P1 and then the label of the nearest point assigned to P1.



Suppose P1 is the point, for which label needs to predict. First, you find the k closest point to P1 and then classify points by majority vote of its k neighbours. Each object votes for their class and the class with the most votes is taken as the prediction. For finding closest similar points, you find the distance between points using distance measures such as Euclidean distance, Hamming distance, Manhattan distance and Minkowski distance. KNN has the following basic steps

1. Calculate distance
2. Find closest neighbours
3. Vote for labels



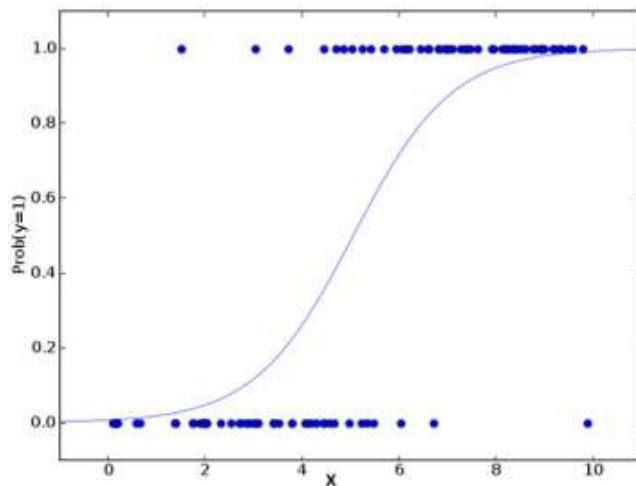
Logistic Regression:

Logistic Regression is one of the basic and popular algorithms to solve a classification problem. It is named as 'Logistic Regression', because its underlying technique is quite the same as Linear Regression.

An explanation of logistic regression can begin with an explanation of the standard logistic function. The logistic function is a sigmoid function, which takes any real value between zero and one. It is defined as

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

And if we plot it, the graph will be **S** curve,



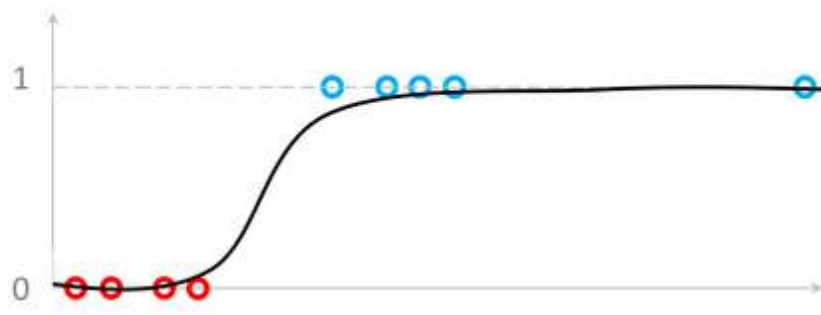
Let's consider t as linear function in a univariate regression model.

$$t = \beta_0 + \beta_1 x$$

So the Logistic Equation will become

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Now, when logistic regression model come across an outlier, it will take care of it.



But sometime it will shift its y axis to left or right depending on outliers' positions.

How to check performance?

To check the performance, we can use confusion matrix and AUC - ROC Curve.

Confusion matrix:

It is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

It is extremely useful for measuring Recall, Precision, Specificity, Accuracy and AUC-ROC Curve.

TP-True Positives

TN-True Negatives

FP-False Positives

FN-False Negatives

Random Forest Classifier:

Random Forest Classifier is ensemble algorithm. Ensemble algorithms are those which combine more than one algorithms of same or different kind for classifying objects.

For example, running prediction over Naive Bayes, SVM and Decision Tree and then taking vote for final consideration of class for test object.

Random forest classifier creates a set of decision trees from randomly selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object.

In Laymen's term,

Suppose training set is given as: [X1, X2, X3, X4] with corresponding labels as [L1, L2, L3, L4], random forest may create three decision trees taking input of subset for example,

1. [X1, X2, X3]
2. [X1, X2, X4]
3. [X2, X3, X4]

So finally, it predicts based on the majority of votes from each of the decision trees made. This works well because a single decision tree may be prone to a noise, but aggregate of many decision trees reduce the effect of noise giving more accurate results.

Alternatively, the random forest can apply weight concept for considering the impact of result from any decision tree. Tree with high error rate are given low weight value and vice versa. This would increase the decision impact of trees with low error rate.

Basic parameters to Random Forest Classifier can be total number of trees to be generated and decision tree related parameters like minimum split, split criteria etc.

Decision Tree:

Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter.

The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.

Pseudo code

- Place the best attribute of dataset at the **root** of the tree.

- Split the training set into **subsets**. They should be made in such a way that each subset contains data with the same value for an attribute.
- Repeat the above 2 steps on each subset until you find **leaf nodes** in all the branches of the tree.

Techniques:

Before applying these algorithms, the text (tweets in our case) needs to be converted into vectors. So, I used the word vectors like Bag-of-words, Tf-Idf and word2vec.

Machine learning algorithms cannot work with raw text directly; the text must be converted into numbers. Specifically, vectors of numbers.

This is called **feature extraction** or **feature encoding**.

Bag-of-words:

- The bag-of-words model is a way of representing text data when modelling text with machine learning algorithms.
- It is simple to understand and implement.
- It is a representation of text that describes the occurrence of words within a document. It involves two things:
 - ✚ A vocabulary of known words
 - ✚ A measure of the presence of known words
- It is called a “*bag*” of words, because any information about the order or structure of words in the document is discarded.
- First, a list of all the unique words in the corpus is made which we call vocabulary.
- Then each document of free text is turned into a vector that we can use as input or output for a machine learning model.
- The vector here is list of 0s and 1s. 0 indicates absence and 1 indicates the presence of a particular word in the vocabulary.

Tf_Idf:

- Tf-idf stands for term frequency-inverse document frequency.
- The tf-idf weight is a weight often used in information retrieval and text mining.
- The tf-idf weight is composed by two terms:
 - ✚ Normalized Term Frequency (TF), aka the number of times a word appears in a document, divided by the total number of words in that document.
 - ✚ Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

word2vector:

- Word2Vec model is used for learning vector representations of words called “word embeddings”.
- It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc.
- It can be obtained using two methods (both involving Neural Networks): Skip Gram and Common Bag Of Words (CBOW).
 - ✚ **CBOW**: This method takes the context of each word as the input and tries to predict the word corresponding to the context.
 - ✚ **SKIP GRAM**: We input the target word into the network. The model outputs C probability distributions.
- Skip Gram works well with small amount of data and is found to represent rare words well.
- On the other hand, CBOW is faster and has better representations for more frequent words.

Benchmark:

Here, I am using the KNN classifier model as my benchmark model. It got a testing accuracy of 0.665 and an F1-score of 0.524. Various other classifier models are build and their performance metrics are measured and compared with other. The one which gives better results will be considered best of all.

Methodology

Data pre-processing:

Text is a highly unstructured form of data, various types of noise are present in it and the data is not readily analyzable without any pre-processing.

Here, I am using a user-defined function to **remove** the **unwanted text patterns** from the tweets.

```
def remove_pattern (input_text,pattern):  
  
    r=re.findall(pattern , input_text)  
  
    for i in r:  
  
        input_text=re.sub(i,"",input_text)  
  
    return input_text
```

Using the above function, I removed the user handles from the tweets. They hardly give any information about the nature of the tweet.

train.head(10)				
	id	label	tweet	tidy_tweet
0	1	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	@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disappointed #getthanked	thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disappointed #getthanked
2	3	0		bihday your majesty
3	4	0	#model i love u take with u all the time in urðz!!! ðððððððððð	#model i love u take with u all the time in urðz!!! ðððððððððð
4	5	0	factsguide: society now #motivation	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. #alishowandnogo	[2/2] huge fan fare and big talking before they leave. chaos and pay disputes when they get there. #alishowandnogo
6	7	0	@user camping tomorrow @user @user @user @user @user @user @user dannyð	camping tomorrow dannyð
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 #gri	the next school year is the year for exams ð can't think about that ð #school #exams #hate #imagine #actorslife #revolutionschool #gri

After the user handles got removed, this is how the tweets look like.

And now, the punctuations, numbers and special characters which tell nothing about the tweets' nature are replaced with white spaces.

```
0]: train['tidy_tweet'] = train['tidy_tweet'].str.replace("[^a-zA-Z#]", " ")
1]: train.head(10)
```

	id	label	tweet	tidy_tweet
0	1	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	@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disappointed #getthanked	thanks for #lyft credit i can t use cause they don t offer wheelchair vans in pdx #disappointed #getthanked
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ur\$!!! \$\$\$\$\$\$/5/5/	#model i love u take with u all the time in ur
4	5	0	factsguide: society now #motivation	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	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 danny&	camping tomorrow danny
7	8	0	the next school year is the year for exams &~ can't think about that & #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	we won!!! love the land!!! #allin #cavs #champions #cleveland #clevelandcavaliers &	we won love the land #allin #cavs #champions #cleveland #clevelandcavaliers

The tweets looked like as shown in the figure after the executing the above piece of code.

Next, short words whose length is less than 3 have been removed.

removing short words

```
1]: train['tidy_tweet'] = train['tidy_tweet'].apply(lambda x: ' '.join([w for w in x.split() if len(w)>3]))
2]: train.head(10)
```

	id	label	tweet	tidy_tweet
0	1	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	@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disappointed #getthanked	thanks #lyft credit cause they offer wheelchair vans #disappointed #getthanked
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ur\$!!! \$\$\$\$\$\$/5/5/	#model love take with time
4	5	0	factsguide: society now #motivation	factsguide society #motivation
5	6	0	[2/2] huge fan fare and big talking before they leave. chaos and pay disputes when they get there. #allshowandnogo	huge fare talking before they leave chaos disputes when they there #allshowandnogo
6	7	0	@user camping tomorrow @user @user @user @user @user @user @user danny&	camping tomorrow danny
7	8	0	the next school year is the year for exams &~ can't think about that & #school #exams #hate #imagine #actorslife #revolutionschool #girl	next school year year exams think about that #school #exams #hate #imagine #actorslife #revolutionschool #girl
8	9	0	we won!!! love the land!!! #allin #cavs #champions #cleveland #clevelandcavaliers &	love land #allin #cavs #champions #cleveland #clevelandcavaliers
9	10	0	@user @user welcome here i i'm it's so #gr8 i	welcome here

Next, stop words are removed after tokenising the tweets.

removing stopwords

```
5]: from nltk.corpus import stopwords
stop_words=set(stopwords.words('english'))
for i in range(len(tokenized_tweet)):
    tokenized_tweet[i]=[w for w in tokenized_tweet[i] if w not in stop_words]
tokenized_tweet.head(10)

6]: 0          [father, dysfunctional, selfish, drags, kids, dysfunction, #run]
    1  [thanks, #lyft, credit, cause, offer, wheelchair, vans, #disappointed, #getthank]
    2                                     [bihday, majesty]
    3                                     [#model, love, take, time]
    4                                     [factsguide, society, #motivation]
    5          [huge, fare, talking, leave, chaos, disputes, #allshowandnogo]
    6                                     [camping, tomorrow, danny]
    7  [next, school, year, year, exams, think, #school, #exams, #hate, #imagine, #actorslife, #revolutionschool, #girl]
    8          [love, land, #allin, #cavs, #champions, #cleveland, #clevelandcavaliers]
    9                                     [welcome]
    Name: tidy_tweet, dtype: object
```

Then, we perform stemming on the tweets.

#stemming(normalising)

```
]: ## normalizing the tokenized tweets
from nltk.stem.porter import *
stemmer=PorterStemmer()
tokenized_tweet=tokenized_tweet.apply(lambda x:[stemmer.stem(i) for i in x])
tokenized_tweet

10: 0          [father, dysfunct, selfish, drag, kid, dysfunct, #run]
    1  [thank, #lyft, credit, caus, offer, wheelchair, van, #disapoint, #getthank]
    2                                     [bihday, majesti]
    3                                     [#model, love, take, time]
    4                                     [factsguid, societi, #motiv]
    5          [huge, fare, talk, leav, chao, disput, #allshowandnogo]
    6                                     [camp, tomorrow, danni]
    7  [next, school, year, year, exam, think, #school, #exan, #hate, #inagin, #actorslif, #revolutionschool, #girl]
    8          [love, land, #allin, #cav, #champion, #cleveland, #clevelandcavali]
    9                                     [welcom]
    10         [wireland, consum, price, index, climb, previou, #blog, #silver, #gold, #forex]
    11  [selfish, #orlando, #standwithorlando, #pulseshoot, #orlandoshoot, #biggerproblem, #selfish, #heabreak, #valu, #love]
    12          [daddi, today, day, #gettingf]
    13         [#cnn, call, #michigan, middl, school, build, wall, chant, #tcot]
```

Since, the dataset is imbalanced (which is evident from the section exploration), I made an new dataset from the previous one by taking equal number of positive and negative tweets.

reducing the dataset

```
In [30]: train_pos = train[train.label==1]
         train_neg = train[train.label==0]
```

```
In [31]: train_1 = train_pos[:500]
         train_2 = train_neg[:500]
         frames = [train_1, train_2]
         red_train = pd.concat(frames)
         print(red_train.shape)
         red_train['label'].value_counts()
```

```
(1000, 4)
```

```
Out[31]: 1    500
         0    500
         Name: label, dtype: int64
```

Implementation:

I applied 3 word vectors and 5 classifiers on the dataset.

Word vectors: bag-of-words, Tf-Idf and word2vec

Classifiers: K-nearest neighbours, Naive bayes, Logistic regression, decision trees and random forest.

First, I converted the tweets into vectors using word vectors.

Then, the dataset is split into training and testing sets of 0.8 and 0.2 proportion resp.

And then classifiers were applied on each set of vectors and metrics were evaluated.

Of all the models built and trained, Logistic regression classifier model gave high accuracy and F1 score on all the word vectors except word2vec.

Logistic regression	Accuracy	F1 score
BOW	0.815	0.802
Tf-Idf	0.835	0.8135
Word2v	0.41	0.58

In this entire process, I felt converting the text to vector form is the difficult task. But, the use of different word vectors made this task a little easier. Also, the classifiers, sometimes, take a long time to get trained.

Refinement:

I used GridSearchCv to tune the parameters of the models. Unlike usually, instead of training the classifiers with random parameters and then refining them using best ones, I trained classifiers with best parameters (in the first attempt itself) which were obtained from GridSearchCv.

While applying GridSearchCV on the models, I passed the following as parameters:

For KNN,

n_neighbors:[1,2,3,4,5,6,7,8,9,10]

For Logistic Regression,

Penalty: ['l1', 'l2']

'C': [0.0001, 0.001, 0.01, 0.1, 1, 10],

Best parameters

	KNN	Logistic regression
--	-----	---------------------

BOW	n_neighbors=1	C=0.1, penalty='l2'
Tf-Idf	n_neighbors=7	C=10, penalty='l2'
Word2vec	n_neighbors=1	C=0.0001, penalty='l2'

Results

Model Evaluation and Validation:

The results of each model are tabulated as follows: (accuracy)

	BOW	Tf-Idf	word2vec
Naive	99, 78	99, 77	50, 47
KNN	100, 66	85, 77	98, 49
Log-regression	96, 81	99, 83	52, 41
Decision tree	65, 72	65, 72	79, 45
Random forest	78, 68	78, 70	98, 51

Naive: 99, 78

Training accuracy=99%

Testing accuracy=78%

Of all the models, I chose Logistic regression because of its better performance. It got a testing accuracy of 81% with 96% training accuracy when bow is applied. And, got 83% testing accuracy and 99% training accuracy when Tf-Idf is applied.

I checked by applying different random states to find the robustness of my model, it gives same accuracy for different random state.

```
In [173]: rand=[76,5,9,34,62,19]
          for i in rand:
              log = LogisticRegression(C=10, penalty="l2",random_state=i)
              log.fit(x_train,y_train)
              pred5=log.predict(x_train)
              acc5=accuracy_score(pred5,y_train)
              print("the training accuracy with {} is:{}".format(i,acc5))
              pred6=log.predict(x_test)
              acc6=accuracy_score(pred6,y_test)
              print("the testing accuracy with {} is:{}".format(i,acc6))

the training accuracy with 76 is:1.0
the testing accuracy with 76 is:0.835
the training accuracy with 5 is:1.0
the testing accuracy with 5 is:0.835
the training accuracy with 9 is:1.0
the testing accuracy with 9 is:0.835
the training accuracy with 34 is:1.0
the testing accuracy with 34 is:0.835
the training accuracy with 62 is:1.0
the testing accuracy with 62 is:0.835
the training accuracy with 19 is:1.0
the testing accuracy with 19 is:0.835
```

This is when it got trained on BOW-applied data .

```
In [174]: rand_lst=[5,24,36,87,9,8,56,61,41]

In [180]: for i in rand_lst:
            log = LogisticRegression(C=10, penalty="l2",random_state=i)
            log.fit(x_train,y_train)
            pred5=log.predict(x_train)
            acc5=accuracy_score(pred5,y_train)
            print("the training accuracy with {} is:{}".format(i,acc5))
            pred6=log.predict(x_test)
            acc6=accuracy_score(pred6,y_test)
            print("the testing accuracy with {} is:{}".format(i,acc6))

the training accuracy with 5 is:0.9975
the testing accuracy with 5 is:0.835
the training accuracy with 24 is:0.9975
the testing accuracy with 24 is:0.835
the training accuracy with 36 is:0.9975
the testing accuracy with 36 is:0.835
the training accuracy with 87 is:0.9975
the testing accuracy with 87 is:0.835
the training accuracy with 9 is:0.9975
the testing accuracy with 9 is:0.835
the training accuracy with 8 is:0.9975
the testing accuracy with 8 is:0.835
the training accuracy with 56 is:0.9975
the testing accuracy with 56 is:0.835
the training accuracy with 61 is:0.9975
the testing accuracy with 61 is:0.835
```

This is when it got trained on Tf_Idf-applied data.

I have passed a new tweet to the model. And the results are almost same. So, I consider this model is robust to this dataset.

Justification:

- ✚ The benchmark model have an accuracy score of 66.5%, the optimized model has obtained an accuracy score of 83.5%. So, this model is good performing.
- ✚ And the Classification Report shows good results of recall, f1-score, and support.

- ✚ In Confusion matrix the number obtained between Actual Positive and Predicted Negative is very less, that means its better performing.
- ✚ There is a reasonable difference in between the unoptimized model and Optimized Model with a difference in accuracy score and f1 score of approx. 17% and 28% respectively.

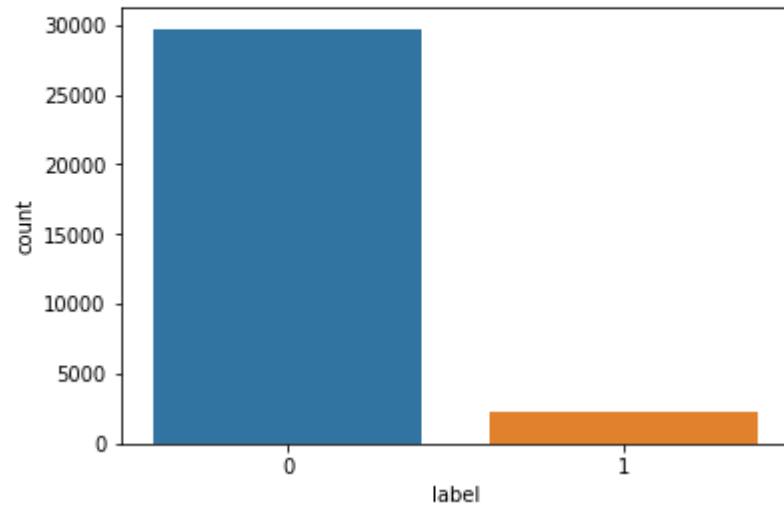
Conclusion

- ✚ After doing my project, I observed some of the interesting facts such as all features are not important to find the target value.
- ✚ We have to remove such features and process the data, after processing the data which are most important to find the target value considered as independent variables and target value is considered as dependent variable.
- ✚ In my project text is most important feature, so I take text as independent variable and target value is label whether the text is positive or negative.
- ✚ Size of dataset is an important thing while training a model. Very large datasets leads to memory errors and all the systems don't have the capacity to handle such a huge data. So I took the reduced dataset here. Also, previously, the dataset was imbalanced. The new dataset is balanced now.

Imbalanced dataset

```
In [108]: sns.countplot(x='label',data=train)
```

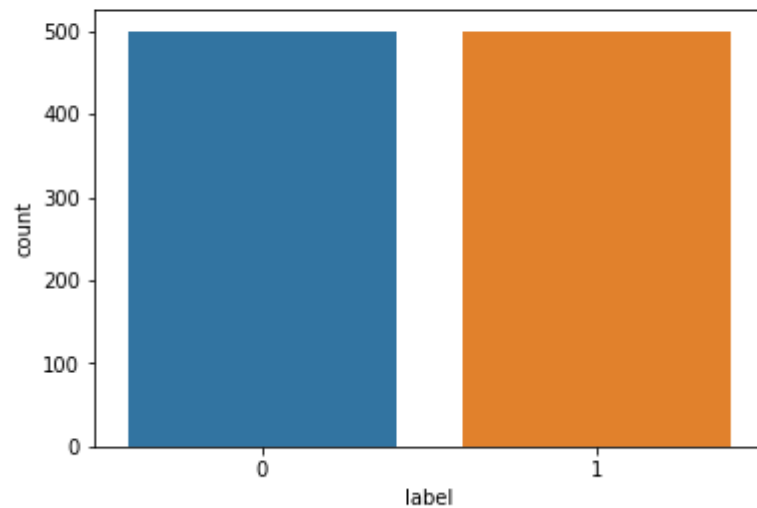
```
Out[108]: <matplotlib.axes._subplots.AxesSubplot at 0x26b3d983710>
```



Balanced dataset

```
In [156]: sns.countplot(x='label',data=red_train)
```

```
Out[156]: <matplotlib.axes._subplots.AxesSubplot at 0x2b353e1fc50>
```



Reflection:

- ✚ Our aim here is that to build a classifier which can correctly classify a tweet as either positive or negative.
- ✚ First, data cleaning and pre-processing is done.
- ✚ Word vectors are applied to convert the text to vectors
- ✚ Various machine learning algorithms are used to build models
- ✚ And finally, the metrics from each of these models are evaluated against each other and the one which gives better score is taken as the best model.

The difficult part of the project is

- How to define a problem, collect the corresponding data and engineer/select the relevant features.
- How to choose appropriate learning algorithm and refine the hyper parameters. Since the obtained model achieve a fairly good performance, I think RF model with a refined hyper parameter should be used in a general setting to solve these types of problems

Improvement:

This can be improved to classify a review into positive or negative by increasing the no of features and number of instances we can also improve the performance by applying some deep learning techniques.