Twitter Sentiment Analysis

- Preena Brar

To give a brief summary about the project, it is about building a model that can detect sentiment, so that this model can be used on small texts to identify if the text is positive or negative. Sentiment Analysis is the process of determining whether a piece of writing is positive or negative. With sentiment analysis, we can generate insights about consumers' reactions to announcements, opinions on products or brands, and even track opinion about events as they unfold.

The task is to build a model that will determine the tone (positive or negative) of the text. To do this, we will need to train the model on the existing data (train.csv). The resulting model will have to determine the class (positive or negative) of new texts (test data that were not used to build the model) with maximum accuracy. Training and test data is gathered from twitter by using Twitter API and the training data is labeled manually to train the model.

We are using Twitter data for our project because over the past few years, text mining got lot of attention, due to an exponential increase in digital text data from web pages, Google's projects and social media services such as Twitter. Twitter data constitutes a rich source that can be used for capturing information about any topic imaginable.

Getting Data from Twitter Streaming API:

For this project, we will use Twitter Streaming API to download tweets.

Authentication:

In order to fetch tweets through Twitter API, we need to register an App through an existing twitter account. We will follow these steps for the same:

- Open this link and click the button: 'Create New App'

- Fill the application details. You can leave the callback URL field empty.
- Once the app is created, you will be redirected to the app page.
- Open the 'Keys and Access Tokens' tab.
- Copy 'Consumer Key', 'Consumer Secret', 'Access token' and 'Access Token Secret'.

Next, we follow these 3 major steps in our program:

- We will be using a Python library called Tweepy to connect to Twitter Streaming API and downloading the data.
- Authorize twitter API client using the all the keys we collected while creating the App.
- Make a GET request to Twitter API to fetch tweets for a particular query.
- Parse the tweets. Classify each tweet as positive, negative or neutral.

Data fields in the training set:

ItemID - id of tweet

Sentiment – sentiment of the tweet (labeled manually)

SentimentText - text of the tweet

0 - negative

1 – positive

ItemID	Sentiment	SentimentText
1	0	is so sad for my APL friend.....
2	0	I missed the New Moon trailer...
3	1	omg its already 7:30 :O
4	0	.. Omgaga. Im sooo im gunna CRy. I've been at this dentist since 11.. I
5	0	i think mi bf is cheating on me!!! T_T

Characteristic features of Tweets:

From the perspective of Sentiment Analysis, we will consider some characteristics of twitter:

Length of a Tweet: The maximum length of a Twitter message is 140 characters. This means that we can consider a tweet to be a single sentence.

Language used: Since twitter only allows for 140 characters in a tweet, it leads the users to use more abbreviations that don't exactly follow conventional spelling or grammar rules. Also, twitter is used via a variety of media including SMS and mobile phone apps. Use of hashtags also gained popularity on Twitter and is a primary feature in any given tweet. **#twitter.**

Domain of topics: People often post about their likes and dislikes on social media. These are not all concentrated around one topic. This makes twitter a unique place to model a generic classifier platform for many brands, organizations and certain people.

Data cleaning steps:

Dealing with text on Twitter can be messy, because people would add all sorts of things while texting each other like emoji's, typos, bad grammar, usage of slang, presence of unwanted content like URLs, stop words etc. We as humans use many liberties while communicating with each other. Therefore, majority of available text data is highly unstructured and noisy in nature so to achieve better insights or to build better algorithms, it is necessary to play with clean data. There are several steps which will be implemented to clean the data as discussed below:

Decoding data: This is the process of transforming information from complex symbols to simple characters which are easy to understand. Text data may be subject to different forms of decoding like “Latin”, “UTF8” etc. Therefore, for better analysis, it is necessary to keep the complete data in standard encoding format. UTF-8 encoding is widely accepted and is recommended to use.

Escaping HTML characters: Data obtained from web usually contains a lot of html entities like < > & which gets embedded in the original data. Therefore, it is necessary to get rid of these entities. Another approach is to use appropriate packages and modules (for example HTML parser of Python), which can convert these entities to standard html tags. For example: < is converted to “<” and & is converted to “&”.

Apostrophe Lookup: To avoid any word sense disambiguation in text, it is recommended to maintain proper structure in it and to abide by the rules of context free grammar. When apostrophes are used, chances of disambiguation increases. For that we can either use preexisting dictionaries available on internet or create our own.

Removal of Stop-words: When data analysis needs to be data driven at the word level, the commonly occurring words (stop-words) should be removed. We can either create a long list of stop-words or we can use predefined language specific libraries.

Split Attached Words: We humans in the social forums generate text data, which is completely informal in nature. Most of the tweets are accompanied with multiple attached words like

RainyDay, PlayingInTheCold etc. These entities can be split into their normal forms using simple rules and regex.

Slangs lookup: Again, social media comprises of a majority of slang words. These words should be transformed into standard words to make free text. The words like luv will be converted to love, Helo to Hello. The similar approach of apostrophe look-up can be used to convert slangs to standard words.

Removal of URLs: URLs and hyperlinks in text data like comments, reviews, and tweets should be removed.

Spelling correction: We've all seen tweets with a plethora of spelling mistakes. In that regard, spelling correction is a useful pre-processing step because this also will help us in reducing multiple copies of words. For example, "Analytics" and "analytcs" will be treated as different words even if they are used in the same sense.

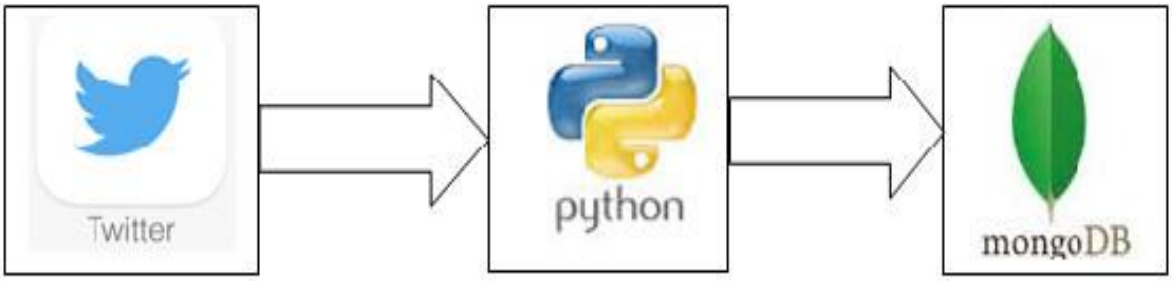
Removing Punctuation: The next step is to remove punctuation including '#' and '@', as it doesn't add any extra information while treating text data. Therefore, removing all instances of it will help us reduce the size of the training data.

Data Storage:

Twitter API output is in JSON format and we need to parse them into a .csv file and store these files. The .csv file works well, but tweets don't always make good flat .csv files, since not every

tweet contains the same fields or the same structure. Some of the data is well nested into the JSON object. Fortunately, NoSQL databases like MongoDB exist and it greatly simplifies tweet storage, search, and recall eliminating the need of a tweet parser.

MongoDB is a document-oriented database. Instead of storing your data in tables made out of individual rows, like a relational database does, it stores your data in collections made out of individual documents. These documents look like just like JSON objects using key-value pairs, but they are called BSON [since it's stored as binary]. In MongoDB, a document is a big JSON blob with no particular format or schema.



Few reasons on why we are using mongoDB for our text data storage:

- Schema less: MongoDB is document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.
- Ease of scale-out: MongoDB is easy to scale
- Uses internal memory for storing the (windowed) working set, enabling faster access of data
- Index on any attribute

- Replication & High Availability

Steps for importing the data into mongoDB:

1) Run MongoDB
2) Now open two instances of the command terminal. In the first window start the MongoDB server by navigating to MongoDB install path/bin and typing mongo.

   We got the following message which means the server is up and running:

   2018-06-18T16:15:25.282-0700 [initandlisten] waiting for connections on port 27017
3) MongoDB provides the mongoimport utility that can be used to import JSON, CSV, or TSV files into a MongoDB database. To import data, we have to open a new Terminal/Command Prompt window and enter mongoimport followed by parameters such as database name, collection name, source file name, etc.
4) Now, in the second command terminal window use the following command to import the collection into our database.

```
mongoimport –db database_name --file /file_path/collection_name.csv
```

Resulting message:

```
2018-06-18T13:34:04.904+0700    no collection specified
2018-06-18T13:34:04.905+0700    using filename 'collection_name' as collection
2018-06-18T13:34:04.911+0700    connected to: localhost
2018-06-18T13:34:04.968+0700    imported 13 documents
```

Visualization:

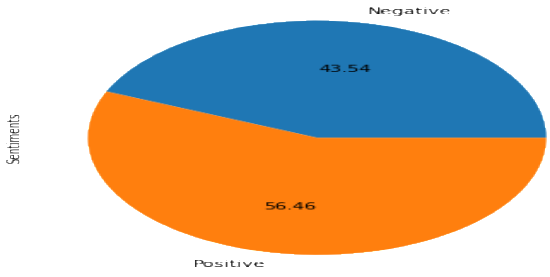
Data visualization refers to the process of converting data into a graphical or visual representation. Because our brains process visual information so efficiently, visualizing data graphically greatly speeds up the process of data analysis. Towards the end of this project, it's important to be able to present the final results in a clear, concise, and compelling manner that the audience, whom are often non-technical clients, can understand.

Matplotlib is a popular Python library that can be used to create your Data Visualizations quite easily. However, setting up the data, parameters, figures, and plotting can get quite messy and tedious to do every time we do a new project.

A **Word cloud** is a visual representation of text data. It displays a list of words, the importance of each being shown with font size or color. Word clouds can identify trends and patterns that would otherwise be unclear or difficult to see in a tabular format. Frequently used keywords stand out better in a word cloud. Common words that might be overlooked in tabular form are highlighted in larger text making them pop out when displayed in a word cloud.

Initial data discoveries through visualizations:

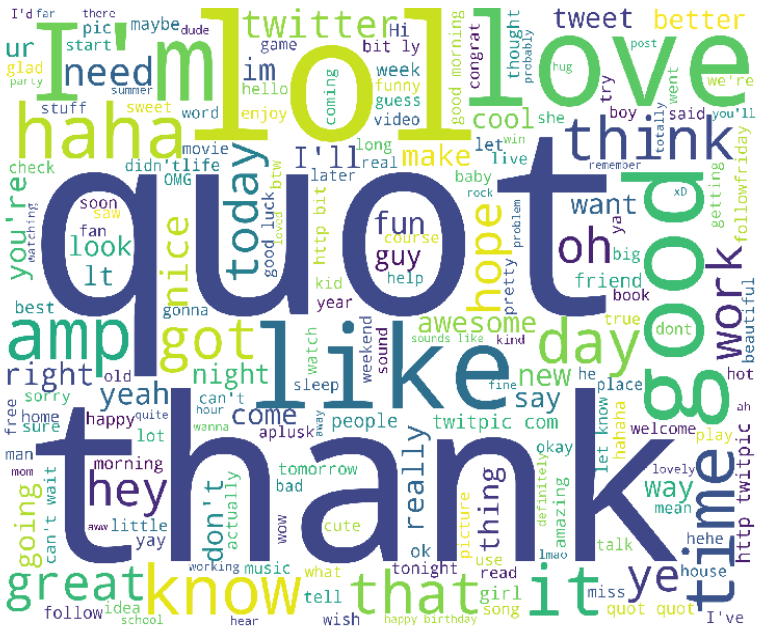
- Comparing the count of positive and negative tweets in the training set by plotting a Pie Chart using matplotlib



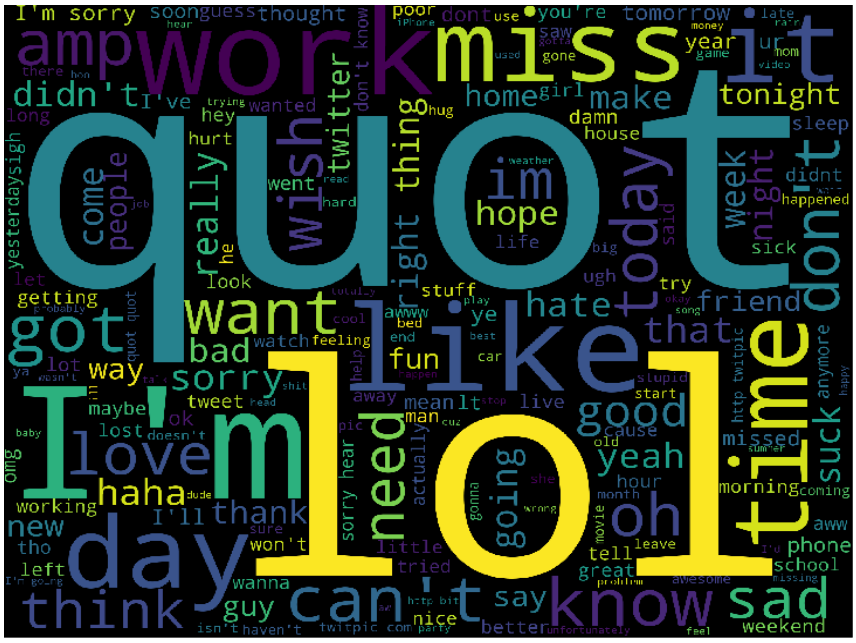
There are 44% negative reviews and 56% positive reviews in the training dataset.

- Word cloud with top frequent words in the whole training dataset

- Two different word clouds with top positive and negative words in the training dataset



Positive word cloud



Negative word cloud

The entire process of cleaning and standardization of text, making it noise-free and ready for analysis is known as text pre-processing. Previously, we discussed some text cleaning steps and a few pre-processing as well. For this assignment we will further discuss all the pre-processing steps involved and some further steps involved to achieve the results.

The next step is the feature engineering step. In this step, raw text data will be transformed into feature vectors and new features will be created using the existing dataset. We will implement the following different ideas in order to obtain relevant features from our dataset.

Lexicon Normalization - Apart from what we tried to reduce during previous assignment, there is another type of textual noise that is about the multiple representations exhibited by single word. For example – “play”, “player”, “played”, “plays” and “playing” are the different variations of the

word – “play”, Though they mean different but contextually all are similar. This step converts all the ambiguities of a word into their normalized form (also known as lemma). Normalization is a pivotal step for feature engineering with text as it converts the high dimensional features (N different features) to the low dimensional space (1 feature), which is an ideal ask for any ML model.

The lexicon normalization practice used in this project is:

- **Stemming:** Stemming is a rudimentary rule-based process of stripping the suffixes (“ing”, “ly”, “es”, “s” etc) from a word.

For example**:**

from nltk.stem.porter import SnowballStemmer

stem = SnowballStemmer()

word = "multiplying"

stem.stem(word)

Output: "multipli"

Tokenization - Tokenizing raw text data is an important pre-processing step for many NLP methods. As explained on wikipedia, tokenization is “the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens.” In the context of actually working through an NLP analysis, this usually translates to converting a string to a list or array as below:

For example:

import nltk

mystring2 = " is so sad for my APL friend.*"*

nltk.word_tokenize(mystring2)

[‘is’, ‘so’, ‘sad’*, ‘for’, ‘my’, 'APL', 'friend’]*

N-Grams - The problem of language detection is that human language (words) have structure. For example, in English, it's very common for the letter 'u' to follow the letter 'q,' while this is not the case in transliterated Arabic. n-grams work by capturing this structure. Thus, certain combinations of letters are more likely in some languages than others. This is the basis of n-gram classification. N-grams are the combination of multiple words used together. Ngrams with N=1 are called unigrams. Similarly, bigrams (N=2), trigrams (N=3) and so on can also be used.

For example: *TextBlob(train['tweet'][0]).ngrams(2)*

Output: [['is','so'], ['so','sad'], ['sad','for'], ['for','my'], ['my','APL'], ['APL', 'friend']]

Model Building

The final step in the text classification framework is to train a classifier using the features created in the previous step. There are many different choices of machine learning models which can be used to train a final model. We can implement following different classifiers for this purpose

1. Naive Bayes Classifier
2. Support Vector Machine
3. Convolutional Neural Network

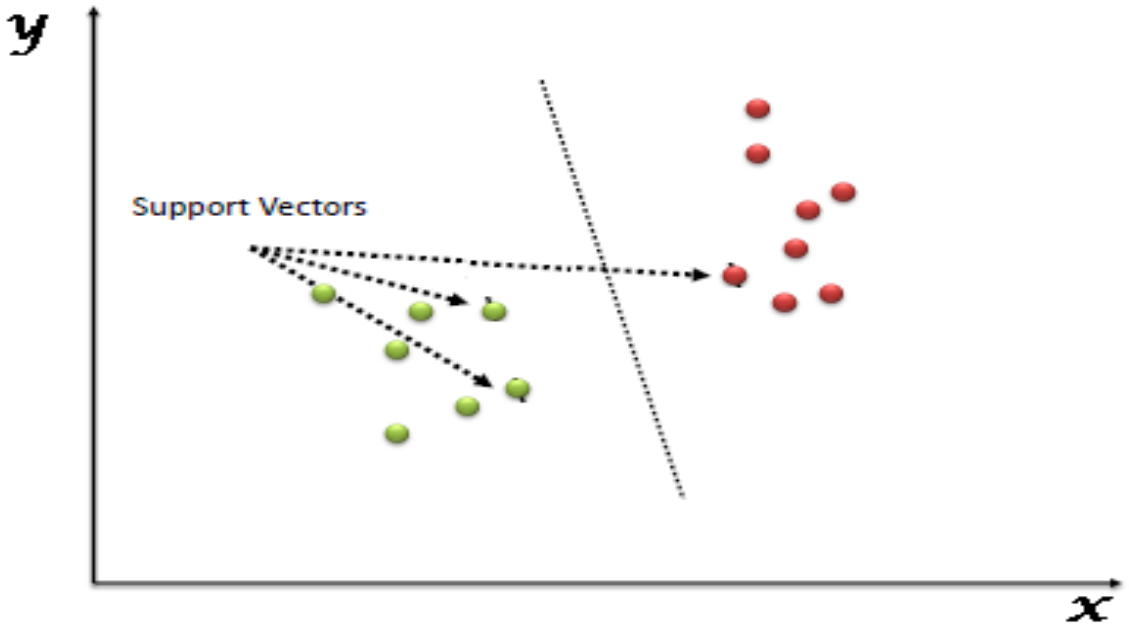
For this project we implemented each of the above-mentioned classifier algorithms. But Support Vector Machine gave us the best results with accuracy nearly equal to 82%.

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, for this project we used it as a classification algorithm. In this algorithm, we plot each data item as a point in n-dimensional space (where n is 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 very well.

In Python, scikit-learn is a widely used library for implementing machine learning algorithms, SVM is also available in scikit-learn library.

Sample:

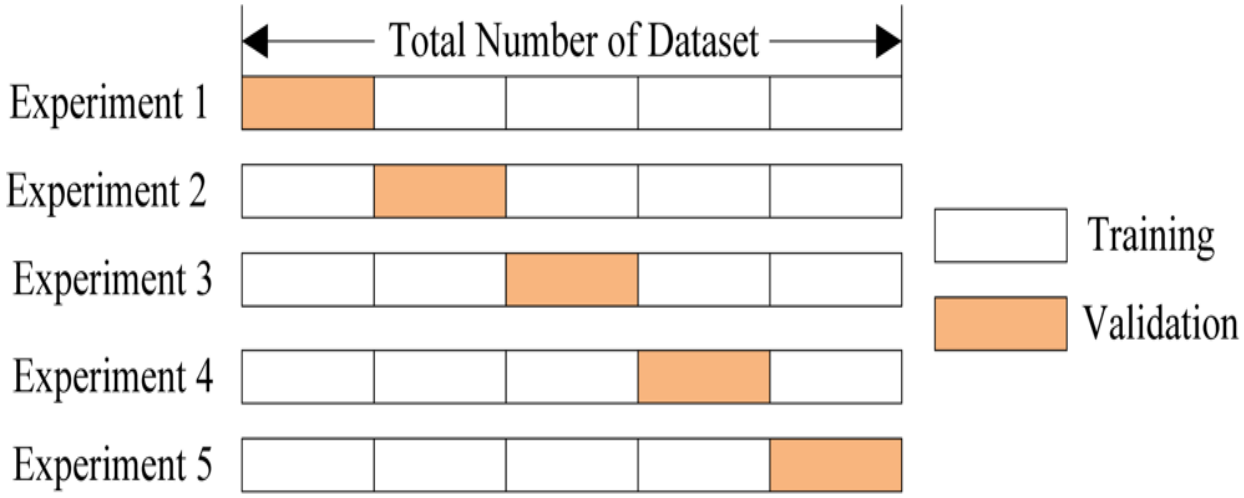
```
from sklearn.svm import SVC
model = SVC(probability=True, kernel="linear", class_weight="balanced")
```



We used **Cross-validation** and **GridSearchCV** to find good hyperparameters for our SVM model.

Cross-validation: Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it. In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples.

In cross-validation, we run our modeling process on different subsets of the data to get multiple measures of model quality. For example, we could have 5 folds or say experiments. We divide the data into 5 pieces, each being 20% of the full dataset.



Cross-validation gives a more accurate measure of model quality, which is especially important if you are making a lot of modeling decisions. However, it can take more time to run, because it estimates models once for each fold. So, it is doing more total work.

GridSearchCV: Grid search is an approach to parameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid. Best C value

for SVM which we got using GridSearchCV is “0.1”.

After building the model, firstly we trained the model with the train data and then tested with the test data. To verify how well this model works, we need a second data set, the test set. We use the model we learned from the training data and see how well it predicts the variable in question for the training set. If it has low accuracy here, it is likely that the model is flawed. One typical reason where a model does very well on the training set but poorly on the test set is what we call overfitting. Basically, it means that the model learned a very specific way of predicting the training data.

The results after testing our model that we got are as follows:

Accuracy: 0.8251729786709477

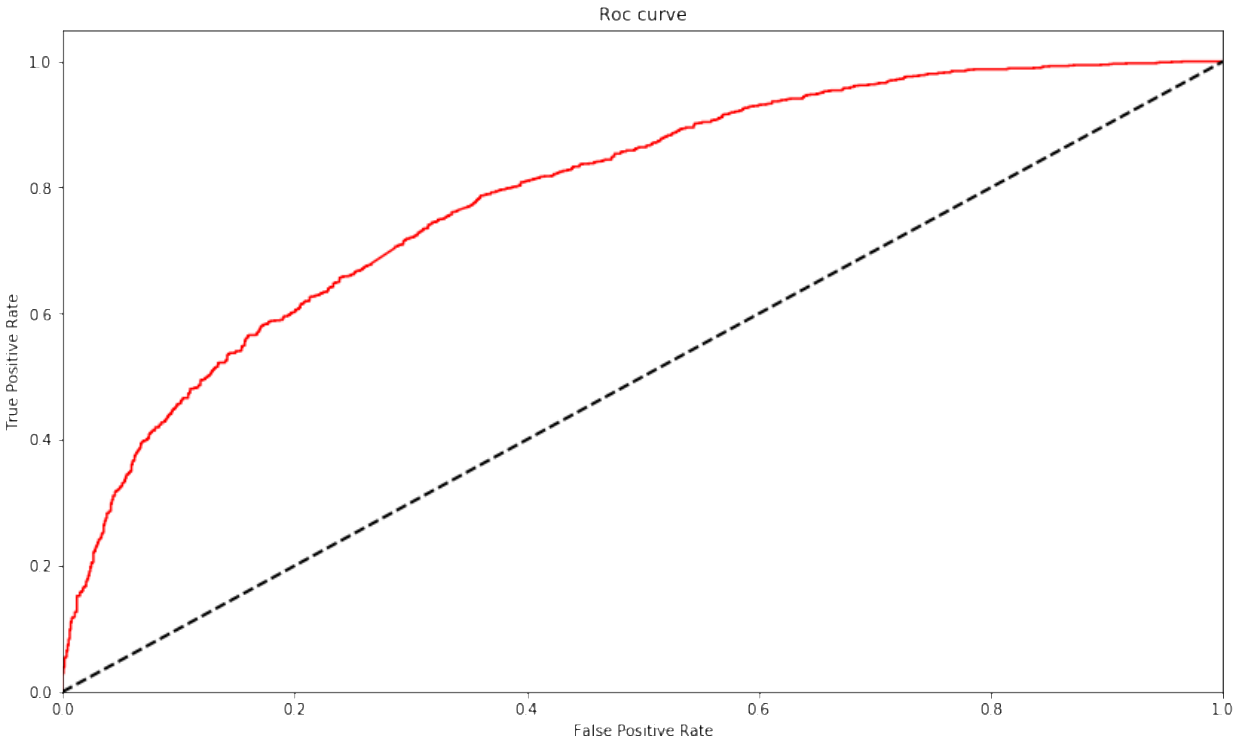
F1 score: 0.7855546001719691

Precision: 0.7631139325091881

Recall: 0.8093550673281361

Next, we plotted the Plotting False Positive Rate vs True Positive Rate by plotting the ROC curve. ROC curves typically feature true positive rate on the Y axis, and false positive rate on the X axis. This means that the top left corner of the plot is the “ideal” point - a false positive rate of zero, and a true positive rate of one. This is not very realistic, but it does mean that a larger area under the curve (AUC) is usually better.

The “steepness” of ROC curves is also important, since it is ideal to maximize the true positive rate while minimizing the false positive rate.



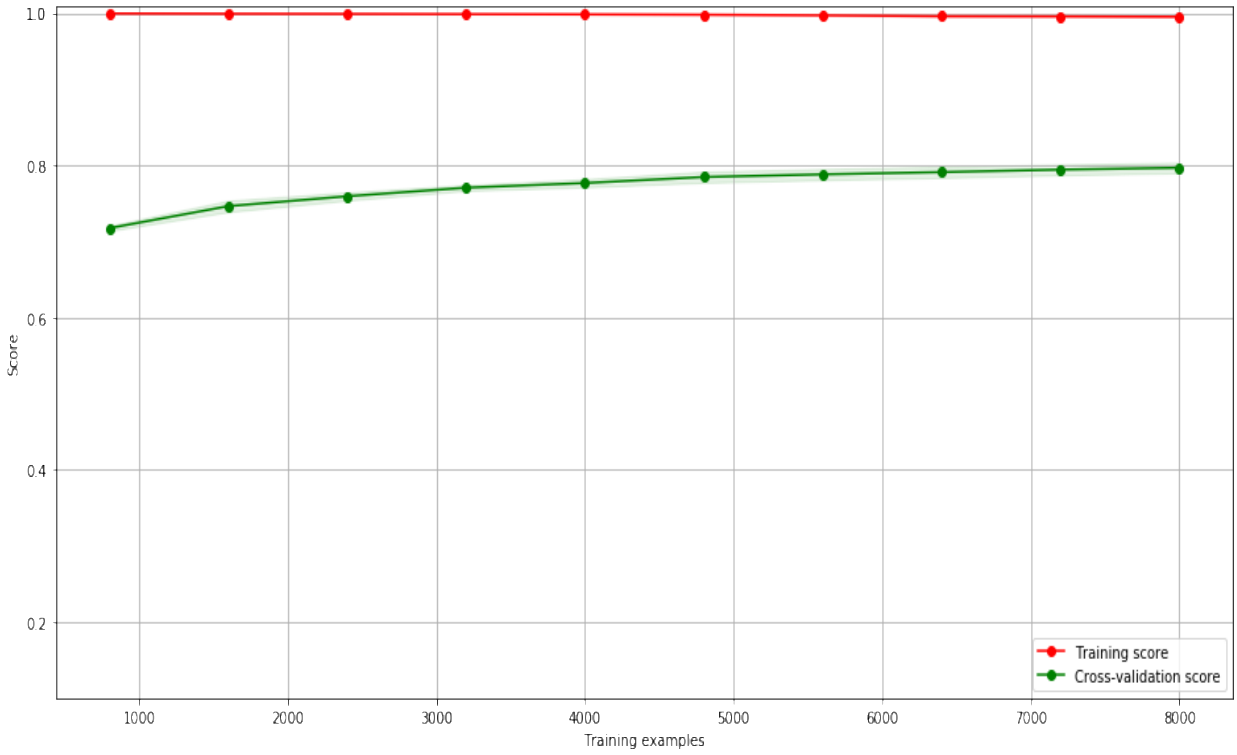
For a perfect classifier the ROC (receiver operating characteristic curve) curve will go straight up the Y axis and then along the X axis. A classifier with no power will sit on the diagonal, whilst most classifiers fall somewhere in between.

Next, we checked if the model has some bias or variance problem by plotting its learning curve. Let's say we have some data and split it into a training set and validation set. We take one single instance from the training set and use it to estimate a model. Then we measure the model's error on the validation set and on that single training instance. The error on the training instance will be 0, since it's quite easy to perfectly fit a single data point. The error on the validation set, however, will be very large. That's because the model is built around a single instance, and it almost certainly won't be able to generalize accurately on data that hasn't seen before.

Now let's say that instead of one training instance, we take ten and repeat the error measurements. Then we take fifty, one hundred, five hundred, until we use our entire training set. The error scores will vary more or less as we change the training set.

We thus have two error scores to monitor: one for the validation set, and one for the training sets. If we plot the evolution of the two error scores as training sets change, we end up with two curves. These are called *learning curves*.

Learning curves give us an opportunity to diagnose bias and variance in supervised learning models. We'll see how that's possible in what follows.



It looks like there isn't a big bias or variance problem, but it is clear that our model would work a little better with more data. if we can get more labeled data the model performance will increase.

Conclusion:

Sentiment analysis is smart enough to identify social media mentions of any brand and pick up comments and posts that could carry actionable insight on what customers really want.

Previously digital marketers only had the options of using structured surveys and questionnaires to elicit useful feedback from customers on social media. With sentiment analysis, these feedback campaigns can be easily monitored to collect all user-generated content that could offer insight. Because the percentage of users that actually respond to direct queries by brands is diminishing. Therefore, sentiment analysis is bound to be the replacement for social media surveys in the near future.

Brand needs to know in order to succeed on social is to understand what worked in the past and do more of it. Sentiment analysis can make that happen by helping brands understand how their social media reputation has panned out. Need not to say that Sentiment Analysis is already proving its worth in market today and it is going to be the future of the marketers.