CAPSTONE 3 PROJECT REPORT

TABLE OF CONTENTS

1. PROBLEM STATEMENT

Twitter has become an essential channel for customer service. In fact, a growing number of companies have specific teams in charge of delivering customer support via this social media platform. Prompt replies are key since 60% of the customers that complain on social media expect a response within one hour.

But how can you evaluate the performance of your customer support on Twitter? Twitter sentiment analysis allows you to track and analyze all the interactions between your brand and your customers. This can be very useful to analyze customer satisfaction based on the type of feedback you receive.

For companies it is crucial to find out whether the customers are satisfied with their services and products or not. And they spent ten thousands of dollars to find out their customers' experience. And people use twitter a lot to express their opinions about the service they get or the product they use by mentioning the company's name in the tweet. So providing an algorithm that finds out whether the products or the service for the companies will help a lot to solve this problem. So I will do this for DoorDash tweets and find out the customers' sentiment (polarity) about DoorDash's services.

2. DATA WRANGLING

2.1. Data Set

The dataset used in this project is scraped from Twitter by using Twint API. The original data had a shape of (1421799, 36). However, since I am analysing the English text, I took only the tweets that were written in English language and I dropped all the columns except the “date” and “time”. I used these columns for EDA after that I dropped those columns as well.

2.2. Data Descriptions

‘id'	: this is the tweet id,
'conversation_id'	: conversation id unique for each conversation,
'created_at'	: data, time and time zone given in string format,
'date'	: date in string format,
'time'	: time in string format,
'timezone'	: time zone in string format,
'user_id'	: twitter user’s id,
'username'	: twitter user’s username,
'name'	: displayed name of the user,
'place'	: displayed place of the user,
'tweet'	: text of the tweet,
'language'	: language of the tweet,
'mentions'	: mentioned if there is any,
'urls'	: links in the tweet (if there is any),
'photos'	: photos in the tweet (if there is any),
'replies_count'	: count of the replies,
'retweets_count'	: count of retweets,
'likes_count'	: count of likes,
'hashtags'	: hashtags in the tweet,
'cashtags'	: cashtags in the tweet,
'link'	: links in the tweet,
'retweet'	: retweeted or not (false or true),
'quote_url'	: mostly nan’s,
'video'	: videos in the tweet,

'thumbnail' : mostly nan’s,
'near' : about location but mostly nan’s,
'geo' : geography of the user but mostly nan’s,
'source' : mostly nan’s,
'user_rt_id' : user retweet id,
'user_rt' : user retweeted or not,
'retweet_id' : id of the retweet,
'reply_to' : username that was replied to,
'retweet_date' : date of the retweet,
'translate' : whether the tweet translated,
'trans_src' : source language of the tweet translated to,
'trans_dest' : destination language of the tweet translated

2.3. Data Cleaning

I am analysing the English text, so I took only the tweets that were written in English language and I dropped all the columns except the “date” and “time”. I used these columns for EDA after that I dropped those columns as well.

2.3.1. Data Type Correction

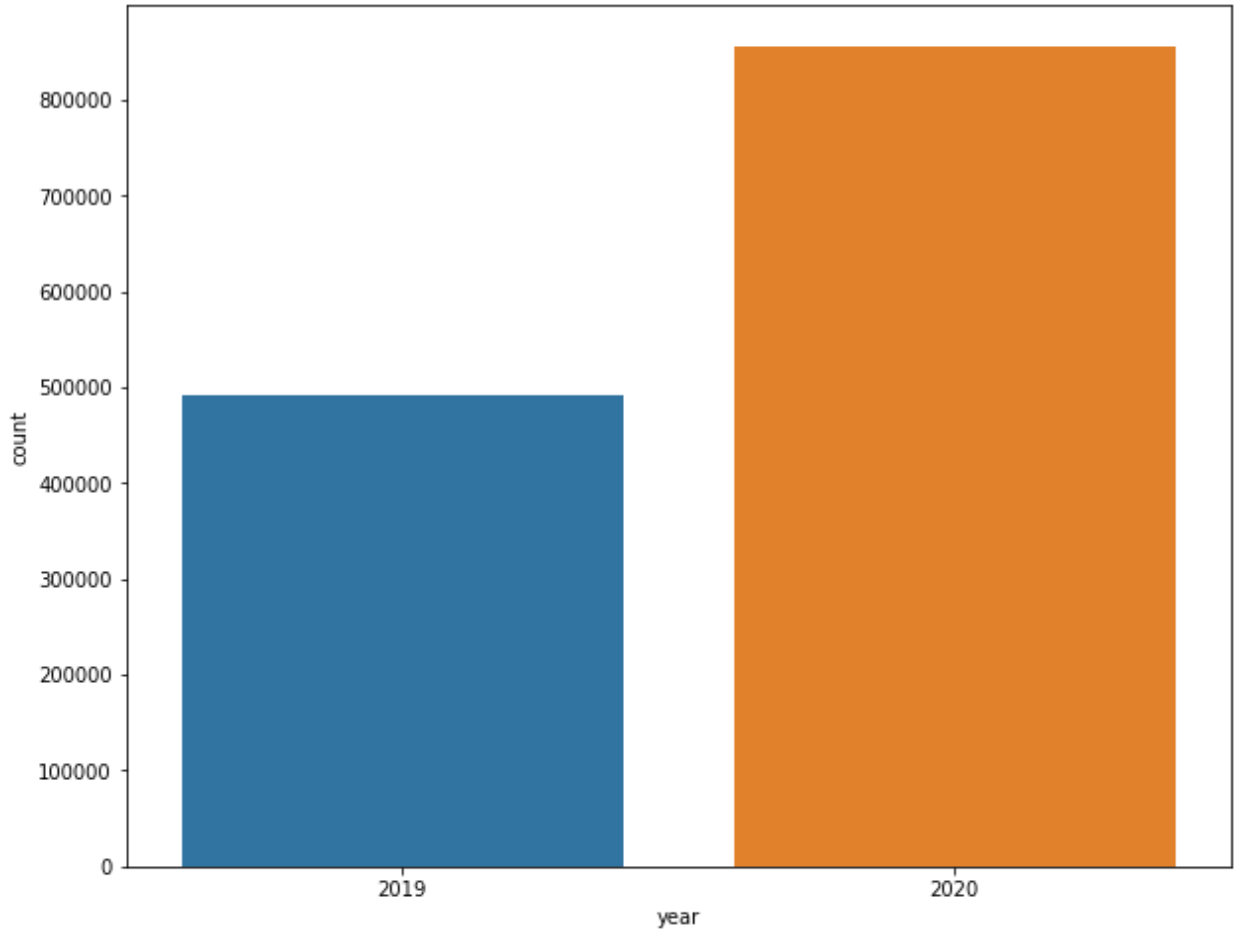
created_at, date and time were given as strings so I changed them to date time objects. I checked if there are any duplicates. There were 12 duplicates. So I removed them.

2.3.2. NaN Imputation

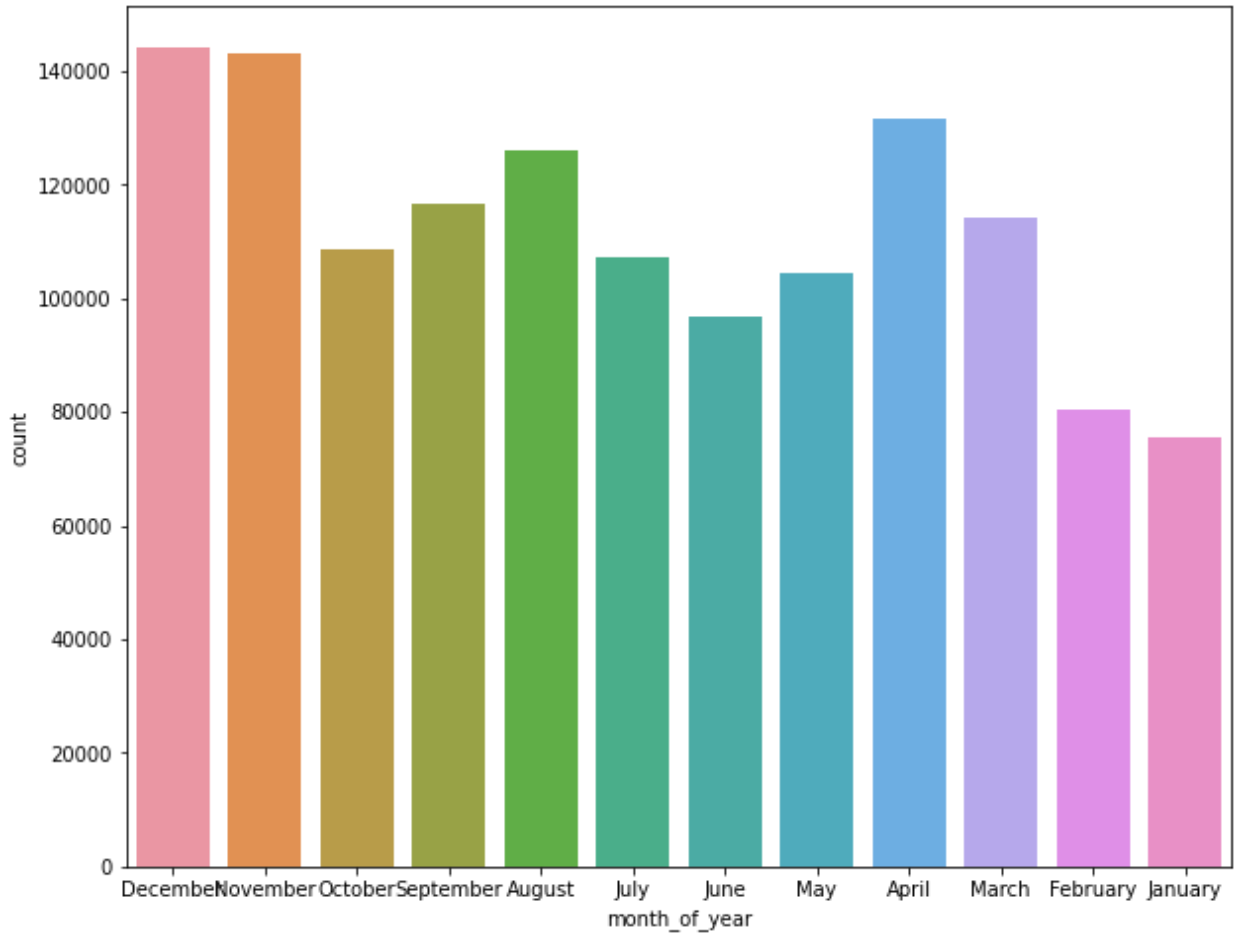
At the beginning there were a lot of columns that had missing values, however after dropping the columns I don't need there were only 6 rows that had NaN’s. Since I know that only 6 rows will not affect the data I dropped those observations.

3. EXPLORATORY DATA ANALYSIS

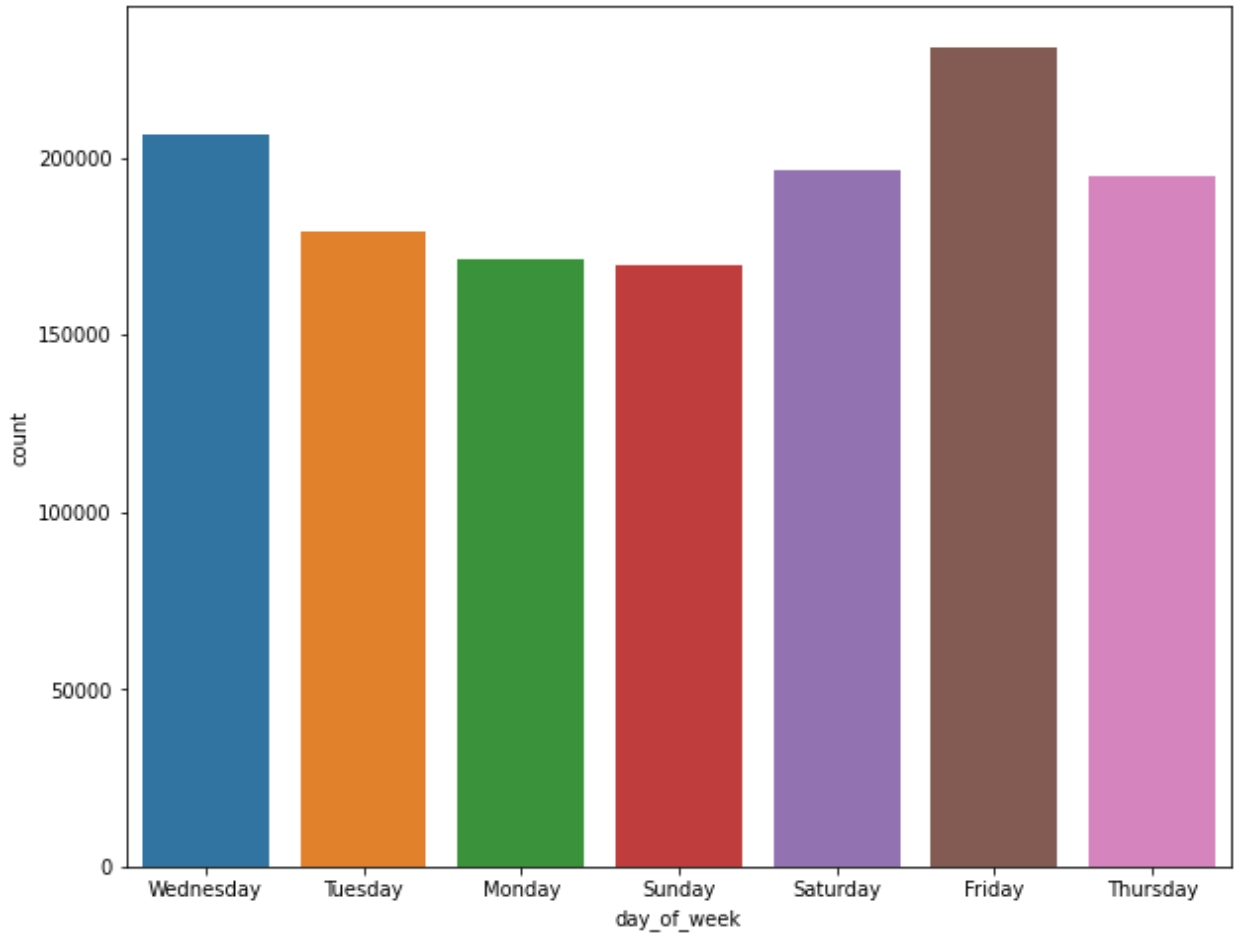
I wanted to have a look at the year, date and time distribution of the tweets. Since DoorDash is a food delivery company I was expecting more tweets during meal times. Following are my findings. I wanted to scrape only 2019 and 2020 tweets. But in my data there 731 tweets from 2018 as well so I dropped them since it was not covering the whole year.



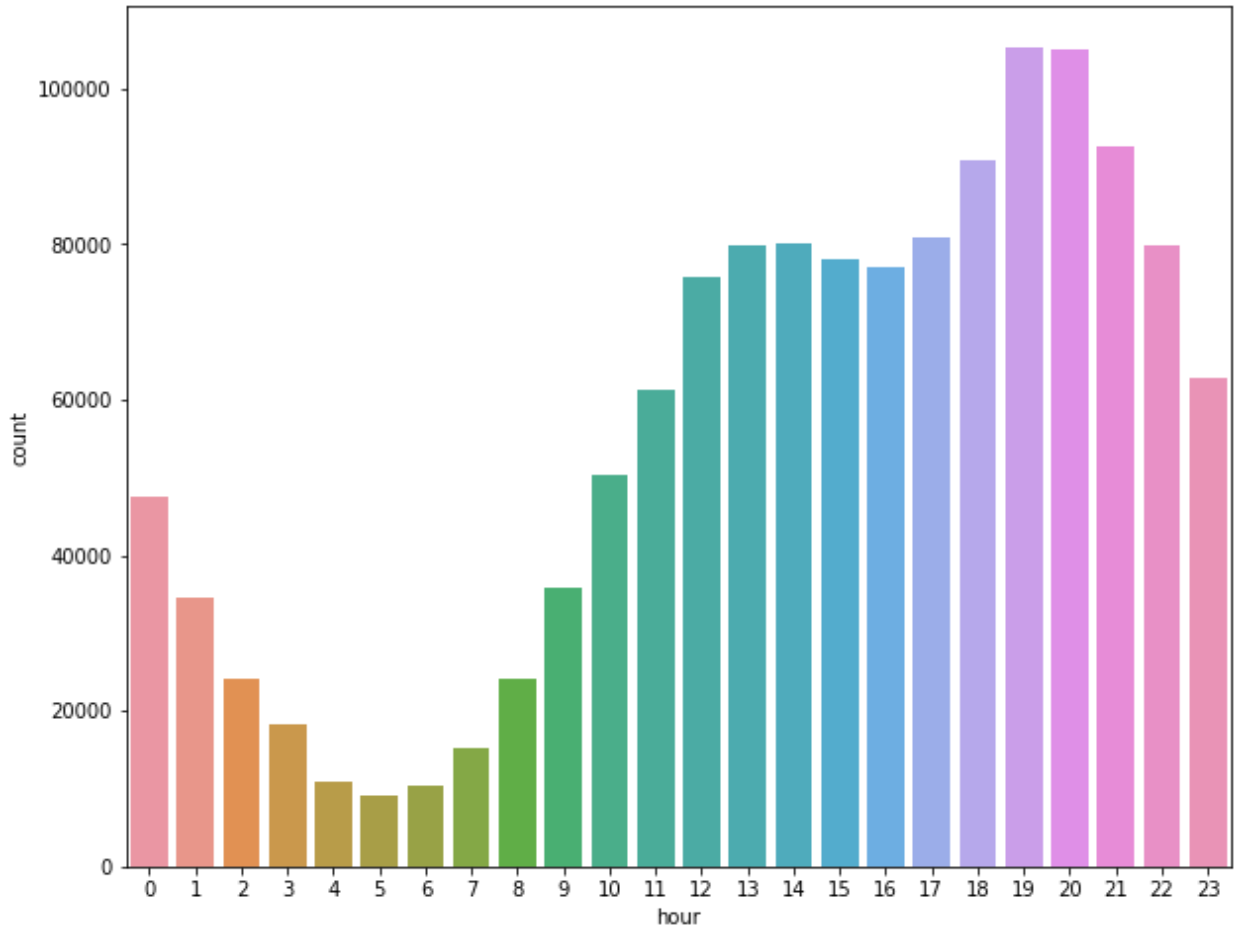
In 2020 there were almost twice the number of tweets than in 2019. That shows that as DoorDash expands its service to more places, more people use it and hence more people tweet about it.



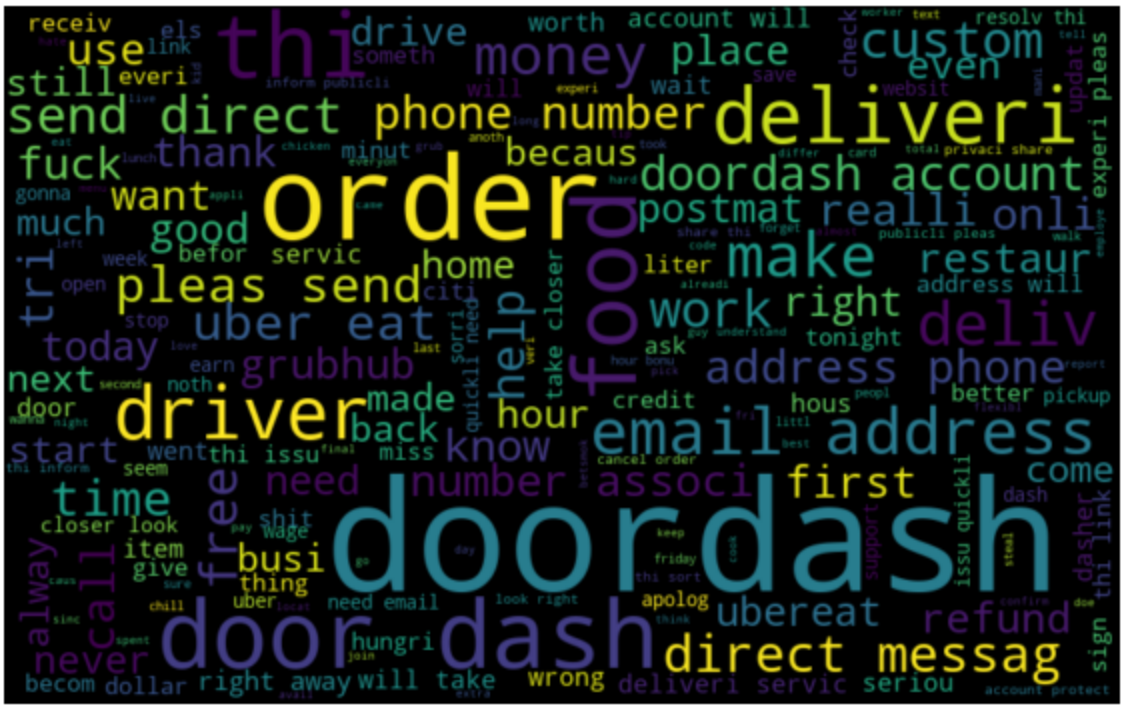
As for distribution with respect to months, we can see that during holidays people tend to order more on DoorDash. We can see a peak in December and November and April, these are the times of Thanksgiving, Christmas and Easter where people get together and share meals. January and February are the least amount. Maybe after spending some money just before January, people tend to cut down on their food spendings.



For the day of the week, Friday leads the most amount of tweets. It might be because people want to reward themselves with food and not want to spend time preparing and spend some time with their families after a tiring week. Strange enough Wednesday takes 2nd place. This might be due to people wanting to give themselves a reward in the middle of the week to be able to end the week with a high note.



When I looked at the hours of the day. In the morning there are not many tweets, however as we get to noon, the number of tweets increases a lot and at around 7:00-8:00 pm it peaks. This might be because people don't order food in the morning and they start ordering when it is lunch time and it peaks during the dinner time.



After I tokenized the tweets, I have also made a word cloud of the words in the tweets. As you can see, words “doordash”, and “order” catch eye at first look.

4. PREPROCESSING

After I was done with the EDA, I dropped all the columns except the text of the tweet, “tweet”. Then by using SentimentIntensityAnalyzer from VaderSentiment i got the sentiment scores for the tweets. And put them in a data frame. Vader sentiment gives the sentiment in 4 columns “Negativity Score”, “Neutrality Score”, “Positivity Score”, and “Compound Score”. I just needed the compound score. So I later dropped other columns. Then I concatenated the original data frame with their sentiment scores. Since it would take very long to apply all of the process to the whole data set. I just took a small sample of it to see how the model works.

4.1. Removing Twitter Handles (@user)

Then I first removed the twitter handles “@user” in the text since it will not add meaning to the text.

4.2. Removing Punctuations, Numbers, and Special Characters

Punctuations, numbers and special characters do not help much. It is better to remove them from the text just as we removed the twitter handles. Here I replaced everything except characters and hashtags with spaces. There were a lot of emojis and punctuations removed in this process.

4.3. Removing Short Words

We have to be a little careful here in selecting the length of the words which we want to remove. So, I have decided to remove all the words having length 3 or less. For example, terms like “hmm”, “oh” are of very little use. It is better to get rid of them.

4.4. Tokenization

I have tokenized all the cleaned tweets in our dataset. Tokens are individual terms or words, and tokenization is the process of splitting a string of text into tokens.

4.5. Stemming

I stemmed the tokenized words. Stemming is a rule-based process of stripping the suffixes (“ing”, “ly”, “es”, “s” etc) from a word. For example, For example – “play”, “player”, “played”, “plays” and “playing” are the different variations of the word – “play”.

5. EXTRACTING FEATURES FROM CLEANED TWEETS

To analyze a preprocessed data, it needs to be converted into features. Depending upon the usage, text features can be constructed

using assorted techniques – Bag-of-Words, TF-IDF, and Word Embeddings. In this project, I used only Bag-of-Words and TF-IDF.

5.1. Bag of Words

Bag-of-Words is a method to represent text into numerical features. It shows the frequency of each word that appears in the document.

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

5.2. 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 penalizing 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.

6. MODEL SELECTION

Before I applied the models, I have created a “label” column by using the compound scores. So any tweet that has a compound score of less than or equal to -0.05 is labeled as “-1” (Negative), any tweet that has a compound score between -0.05 and 0.05 is labeled as “0” (Neutral) and any tweet with a compound score of greater than or equal to 0.05 labeled as “1” (Positive).

Firstly, I used Logistic Regression with Bag of Words Features. Secondly, I tried Logistic Regression with TF-IDF Features. Thirdly, I tried Random Forest with Bag of Words Features and Lastly. I tried Random Forest with TF-IDF Features. And the metrics are given in the below table.

Model	Parameter	Precision	Recall	F1	Accuracy
Logistic R. with BoW	default	0.56	0.54	0.54	0.54
Logistic R. with TF-IDF	default	0.63	0.52	0.55	0.52
Random Forest with BoW	default	0.57	0.53	0.54	0.53
Random Forest with TF-IDF	default	0.56	0.51	0.52	0.51

The models performed very close to each other.

7. FUTURE WORK

I started trying BERT modeling with transformers. BERT gives very good results with text analysis by using Neural Nets. However, when I ran the BERT model, the kernel stopped. After checking why it was stopped, I found out that it was due to insufficient memory since BERT requires a lot of memory to run. Due to the computation limitation of my machine, I couldn't try BERT. So this work can be done with a stronger machine or in the cloud.