Analysis of customer support response on Twitter

Apoorv Awasthi, Bala Akhil Rajdeep Battula
Indiana University-Purdue University, Indianapolis, IN 46202, USA
aawasth@iu.edu, bbattula@iu.edu

1. Introduction

One of the most important selling points for a company is customer support. If a company looks after the needs and problems faced by its customers, they gain a loyal consumer base. Customers also feel connected to the company if the company offers them support in case of any problems. According to the sprout social index: "40% of consumers expect brands to respond within the first hour of reaching out on social media, while 79% expect a response in the first 24 hours."[1]

Twitter has become one of the best platforms to contact customer support because it facilitates users to convey their problems and issues in brief so that the companies can understand and acknowledge the problem effectively. The messages of the customers contacting the company are open to the public so that the company is persuaded to give a quick response[2]. The quick response of the company will give its customers a sense of satisfaction and will contribute to increasing the brand value of the company.

In this project, we analyzed the customer's tweets to the companies and the company's responses. We found a 'customer support on twitter' dataset which contained the text of the tweets of the customers as well as the response of the companies along with the response time. We tried to understand the pattern behind the tweets to get appropriate response time by applying machine learning classification and regression algorithms. We did sentiment analysis of all the tweets to quantify the text of the tweet to make it easier for interpretation, visualization, and modeling.

2. Problem statement

Contacting customer support on Twitter is tough due to the bulk load of requests. Companies are unable to respond to everyone and selectively respond to some of the tweets. If the company responds, the response is usually very late for most of the customers. This severely impacts customer satisfaction which in turn is not good for the image of the company. We plan on aiding the customers by recommending them the type of tweet based on our analysis to get a quick response from customer support.

3. Data description

The dataset is taken from Kaggle. https://www.kaggle.com/datasets/thoughtvector/customer-support-on-twitter?resource=download

This dataset contains information about the customer and the company tweets, the time of tweet along with the author id, tweet id, and two more columns indicating if the given tweet is in response to some tweet or if there is some tweet that is a response to the given tweet. A screenshot of the dataset can be found below.

```
tweetdf1.show()
```

tweet_id	author_id	inbound	created_at	text	response_tweet_id	in_response_to_tweet_id
1	sprintcare	False	Tue Oct 31 22:10:...	@115712 I underst...	2	3
2	115712	True	Tue Oct 31 22:11:...	@sprintcare and h...	null	1
3	115712	True	Tue Oct 31 22:08:...	@sprintcare I hav...	1	4
4	sprintcare	False	Tue Oct 31 21:54:...	@115712 Please se...	3	5
5	115712	True	Tue Oct 31 21:49:...	@sprintcare I did.	4	6
6	sprintcare	False	Tue Oct 31 21:46:...	@115712 Can you p...	5,7	8
8	115712	True	Tue Oct 31 21:45:...	@sprintcare is th...	9,6,10	null
11	sprintcare	False	Tue Oct 31 22:10:...	@115713 This is s...	null	12
12	115713	True	Tue Oct 31 22:04:...	@sprintcare You g...	11,13,14	15
15	sprintcare	False	Tue Oct 31 20:03:...	@115713 We unders...	12	16
16	115713	True	Tue Oct 31 20:00:...	@sprintcare Since...	15	17

Fig. 1. Original Dataset

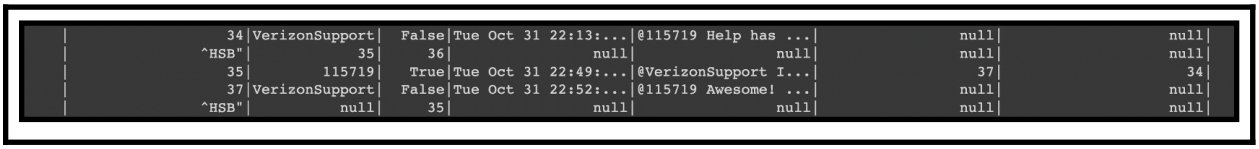
Let us explain all the attributes one by one.

- tweet_id: This column indicates the unique identification number for a particular tweet.
- author_id: This column indicates the unique identification number for a particular user. If the user is a company, then the name of the company is the author_id.
- inbound: This column indicates whether the tweet is directed towards the company or not. So, all the tweets from customers will be marked as 'True' because they are directed towards some company.
- created_at: This column indicates the time of the creation of the tweet.
- text: This column contains the actual text of the tweet.
- response_tweet_id: This column contains the tweet_id of the tweet which is a response to the given tweet.
- in_response_to_tweet_id: This column contains the tweet id of the tweet to which the given tweet is a response. [3]

4. Methodology

4.1 Data cleaning

The first step that we did was the data cleaning step. Our dataset fortunately did not contain any null values. But there were few rows in the dataset which contained non-numeric tweet_id. Evidence of this is shown below:



34	VerizonSupport	False	Tue Oct 31 22:13:...	@115719 Help has ...	null	null
^HSB"	35	36	null	null	null	null
35	115719	True	Tue Oct 31 22:49:...	@VerizonSupport I...	37	34
37	VerizonSupport	False	Tue Oct 31 22:52:...	@115719 Awesome! ...	null	null
^HSB"	null	35	null	null	null	null

Fig. 2. Few rows containing non-numeric tweet_id

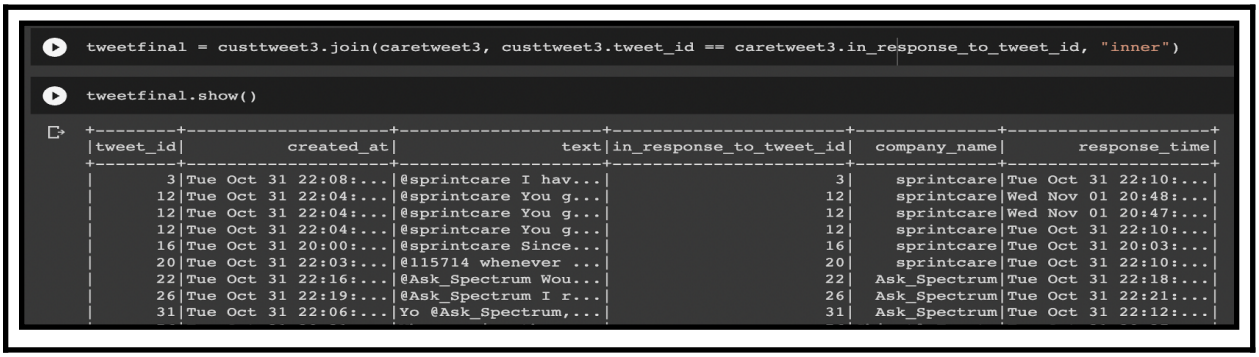
As can be seen in the first column, there are two rows with non-numeric tweet_id. These rows along with many other similar rows were filtered out from the dataset.

4.2 Data transformation

The second step was to transform our dataset in such a way that it can be used for analysis. We recognized that our dataset had both the company tweets and the customer tweets together. We felt the need to split our dataset into two parts - one which contained only the company tweets and the other which contained the customer tweets. We did that by using the author_id column. We figured out that all the non-numeric author_id values were the company tweets. All other numeric author_id values were in rows that contained customer tweets. Based on this logic we removed all the non-numeric author_id to get the customer tweets and then subtracted this dataset from the original dataset to get the company tweets. Moreover, we only selected a few columns from both the datasets.

From the customer tweet dataset, we selected 'tweet_id', 'created_at', and the 'text' columns whereas from the company tweet dataset, we selected 'in_response_to_tweet_id', 'company_name', and the 'created_at' which was renamed to 'response_time'.

In the next step, we inner joined both the datasets on the columns 'tweet_id' and 'in_response_to_tweet_id' to get the final dataset as shown below:



```
tweetfinal = custtweet3.join(caretweet3, custtweet3.tweet_id == caretweet3.in_response_to_tweet_id, "inner")
```

```
tweetfinal.show()
```

tweet_id	created_at	text	in_response_to_tweet_id	company_name	response_time
3	Tue Oct 31 22:08:...	@sprintcare I hav...	3	sprintcare	Tue Oct 31 22:10:...
12	Tue Oct 31 22:04:...	@sprintcare You g...	12	sprintcare	Wed Nov 01 20:48:...
12	Tue Oct 31 22:04:...	@sprintcare You g...	12	sprintcare	Wed Nov 01 20:47:...
12	Tue Oct 31 22:04:...	@sprintcare You g...	12	sprintcare	Tue Oct 31 22:10:...
16	Tue Oct 31 20:00:...	@sprintcare Since...	16	sprintcare	Tue Oct 31 20:03:...
20	Tue Oct 31 22:03:...	@115714 whenever ...	20	sprintcare	Tue Oct 31 22:10:...
22	Tue Oct 31 22:16:...	@Ask_Spectrum Wou...	22	Ask_Spectrum	Tue Oct 31 22:18:...
26	Tue Oct 31 22:19:...	@Ask_Spectrum I r...	26	Ask_Spectrum	Tue Oct 31 22:21:...
31	Tue Oct 31 22:06:...	Yo @Ask_Spectrum,...	31	Ask_Spectrum	Tue Oct 31 22:12:...

Fig. 3. Dataset after inner joining

Now, this dataset contains the customer 'tweet_id', the time of the creation of the tweet (created_at), the text of the tweet (text), the company to which the tweet was addressed (company_name), and the time of response by the company (response_time).

4.3 Feature engineering

The next step was to create features that will be used for machine learning. For the first feature, we decided to calculate the time of response taken by the company by subtracting the time of the creation of a customer tweet from the time of response of the company tweet. But both these columns (created_at and response_time) were in string format. We first converted them to timestamps and subtracted them to get the new variable which we named 'response_dur' which contained the time (in minutes) taken by the company to give the response.

For the second feature, we went on to calculate the sentiment score of the 'text' column using the 'Vader sentiment' library. Vader sentiment library has a 'SentimentIntensityAnalyzer()' function which gives the positive, negative, neutral, and the compound score of the text. Using this, we got our second feature 'compound' which is the net sentiment score of the given text.

Now we have two features: Sentiment score ('compound') which is the independent variable and the response time ('response_dur') which is the target or response or dependent variable.

4.4 Statistical inference and visualization of the independent and response variable

Now we have both our variables. But before going directly into machine learning, we first decided to understand the distribution of both variables. We first check the summary of both variables.

```
tweetfinal5.describe(['compound']).show()

+-------+--------------------+
|summary|            compound|
+-------+--------------------+
|  count|             1220336|
|   mean|0.022374618301841922|
| stddev|  0.4399329705796217|
|    min|             -0.9992|
|    max|              0.9998|
+-------+--------------------+
```

Fig. 4. Summary of sentiment scores

```
tweetfinal5.describe(['response_dur']).show()

+-------+------------------+
|summary|      response_dur|
+-------+------------------+
|  count|           1220336|
|   mean|288.15627909034794|
| stddev| 5289.141652670688|
|    min|       -2052878.85|
|    max|2609746.8333333335|
+-------+------------------+
```

Fig. 5. Summary of response time

We found that sentiment scores range from -1 to +1 and have a mean of about 0.022, whereas response time has a mean of about 288 minutes.

Next, we did the normality testing of both variables. For that, we converted the data frame to pandas data frame and then did the normality testing using the normaltest() function present in the stats package of the scipy library.

```
[70] from scipy import stats
     stats.normaltest(compound)

     NormaltestResult(statistic=46578.00313170772, pvalue=0.0)

[72] from scipy import stats
     stats.normaltest(tweetpandas["response_dur"])

     NormaltestResult(statistic=6317896.3538639825, pvalue=0.0)
```

Fig. 6. Normality testing of both variables

We find that in both cases, the p-value is very close to zero. Hence we can reject the null hypothesis of normality and conclude that both variables are not normally distributed.

Below are the density plots of both the variables:

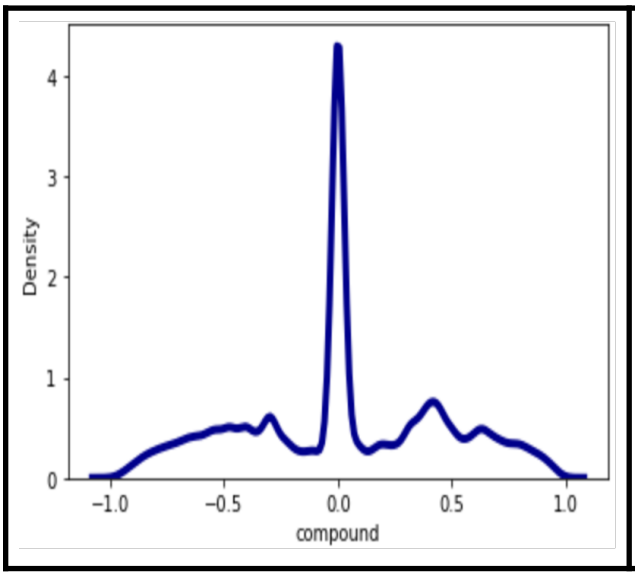


Fig. 7. Density plot of sentiment scores

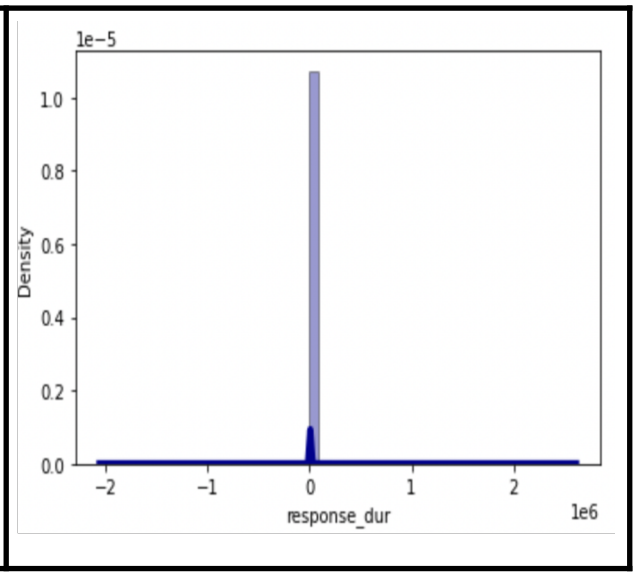


Fig. 8 Density plot of response time

Both the plots also show that both variables are not normally distributed.

4.5 Machine learning models

4.5.1 Linear regression model

As both the input variable (sentiment score) and the response variable (response time) are continuous, we went ahead with the linear regression model. We first did the vector assembling to get input variables as a vector and then split the dataset into test (30%) and train (70%). Then we fitted the linear regression model to the training dataset and got the coefficient value as 23.57 and intercept value as 286.60

For model evaluation, we calculated the R-squared and the Root mean squared error value and got the following results:

```
trainingSummary2 = M2.summary
print("RMSE: %f" % trainingSummary2.rootMeanSquaredError)
print("r2: %f" % trainingSummary2.r2)
```

```
RMSE: 5606.067696
r2: 0.000003
```

Fig. 9. Evaluation of linear regression model

We got the r-squared value as 0.000003 which is not good and the RMSE value is 5606.06 which means that on average, our predictions were off by 5606.06 minutes. Both the evaluation parameters show that the linear regression model is not a good model to predict the response time. Moreover as discussed in the previous section, as both the

variables (input and output) were not normally distributed, the linear regression model is not a good choice, and hence it is giving bad results.

4.5.2 Classification models

As the linear regression model gave bad results, we went ahead with the classification models. For classification models to work, we needed to divide our response variable into classes. So, we divided the response time variable into two classes - Class 0 for all values of response time less than 60 minutes and Class 1 for all values of response time equal to or greater than 60 minutes. We named this new column 'label'. Now we had to predict the label using the input variable which is the sentiment score.

To evaluate our models, our main evaluator was 'Area under ROC curve' which was used to compare the models to choose which one among them performed the best. After finding the best model, we used other evaluators on that model such as 'F1 score', 'Accuracy', 'Weighted Recall', and 'Weighted Precision'.

We used four classification models on our dataset namely - Logistic regression, Random forest, Gradient boosting classifier, and Support vector machines.

We first did the vector assembling to get our input variable converted to vector and then split the dataset into test (40%) and train (60%). Then we fitted all the four models to the training dataset and made predictions on the test dataset. Then we used 'Area under ROC curve' on all four models to evaluate their performance and compare the models. The results are shown below:

```
[ ] print(evaluator.evaluate(predictionslr))

    0.5118854829381144
```

Fig. 10. Logistic Regression

```
print(evaluator.evaluate(predictionsrf))

0.5212261422787738
```

Fig. 11. Random Forest

```
[ ] print(evaluator.evaluate(predictionsgbt))

    0.533877964141122
```

Fig. 12. Gradient boosting classifier

```
[ ] print(evaluator.evaluate(predictionslsvc))

    0.5
```

Fig. 13. Support Vector Machines

From these results, we see that the gradient boosting classifier is the best performing model with the highest 'Area under ROC' values. Now, if we see the count of prediction values on the test dataset, we see the following results:

```
[ ] predictionsgbt3.groupBy("prediction").count().show()

    +----------+------+
    |prediction| count|
    +----------+------+
    |       0.0|488128|
    +----------+------+
```

Fig. 14. Count of prediction values on the test dataset

This is a problem because our model is unable to make even one prediction of Class 1. This is because we had a class imbalance in our original dataset and that is why the model is not getting enough training data to find the underlying pattern in class 1. The class imbalance is shown below:

```
[ ] tweetfinal5.groupby("label").count().show()

+-----+------+
|label| count|
+-----+------+
|    1|401668|
|    0|818668|
+-----+------+
```

Fig. 15. Count of prediction values on the test dataset

Now to avoid this class imbalance, we formed the classes based on the median value of the response time variable. So, finding the median value:

```
tweetfinal5.approxQuantile("response_dur", [0.5], 0.25)

[21.716666666666665]
```

Fig. 16. Finding the median value of the response time variable

We find that the median value is 21.716 minutes. Now we divide the data into two classes - Class 0 for all values of response time less than the median value and Class 1 for all values of response time equal to or greater than the median value. We again did the vector assembling, splitting into test and train, fitting all four models on the training dataset, and making predictions on the test dataset. Then again we used 'Area under ROC curve' to evaluate our models and found the following results:

```
[ ] print(evaluator.evaluate(predictionslr2))

    0.5124859783131245
```

Fig. 17. Logistic Regression

```
print(evaluator.evaluate(predictionsrf2))

0.5200443352385021
```

Fig. 18. Random Forest

```
[ ] print(evaluator.evaluate(predictionsgbt2))

    0.5266545590513327
```

Fig. 19. Gradient boosting classifier

```
[ ] print(evaluator.evaluate(predictionslsvc2))

    0.5
```

Fig. 20. Support Vector Machines

We again found that the gradient boosting classifier is the best performing model among all four models. Now checking the count of the prediction values on the test dataset, we found the following results:

```
[ ] predictionsgbt4.groupBy("prediction").count().show()

+----------+------+
|prediction| count|
+----------+------+
|       0.0|283388|
|       1.0|204740|
+----------+------+
```

Fig. 21. Counting the prediction values on the test dataset

The model performs better on this dataset as it can make predictions on both classes. After this, we evaluated this model further using other evaluators:

```
[ ] #accuracy
    print(evaluator2.evaluate(predictionsgbt2))

    0.5219638242894057

    #f1
    print(evaluator3.evaluate(predictionsgbt2))

    0.4936412676703965

[ ] #Weighted precision
    print(evaluator4.evaluate(predictionsgbt2))

    0.5274723135548249

[ ] #Weighted recall
    print(evaluator5.evaluate(predictionsgbt2))

    0.5219638242894057
```

Fig. 22. Evaluating the best model using F1 score, accuracy, weighted precision, and weighted recall

We found that all the evaluators except the 'F1 score' were giving the value of 0.52. This shows that since this value is far from one, this model is not the best but it is better than the linear regression model.

5. Results

5.1 Understanding the best performing model

Now we tried to understand the results shown by our model. Based on the predictions made by our model on the test dataset, we decided to divide the dataset into two groups - 'Tweets getting early response' (prediction value = 0) and 'Tweets getting late response' (prediction value = 1). Then we decided to check whether the sentiment value of tweets in both groups is different or not. For this, we first did normality testing on both groups:

```
from scipy import stats
stats.normaltest(prediction1)
```

```
NormaltestResult(statistic=1359811.415295219, pvalue=0.0)
```

```
stats.normaltest(prediction0)
```

```
NormaltestResult(statistic=27486.219519633258, pvalue=0.0)
```

Fig. 23. Checking the normality of both the groups

As P-value for both the groups was very close to zero, we can reject the null hypothesis and conclude that the sentiment scores in both the groups were not normally distributed. Now, as these groups were independent with non-normal distribution, we decided to compare both the groups using the 'mannwhitneyu test'. The results of this test are shown below:

```
#mannwhitneyu test as both groups are not normally distributed and independent
stats.mannwhitneyu(prediction0,prediction1)
```

```
MannwhitneyuResult(statistic=20117265838.0, pvalue=0.0)
```

Fig. 24. Comparing both the groups using the mannwhitneyu test

As we see from the result, P-value is very close to zero. Therefore, we can reject the null hypothesis and conclude that both the groups are different from each other.

Then we tried to understand this difference by checking the summary and density plots of both the groups.

```
prediction1.describe()
```

```
count    218656.000000
mean         -0.131262
std           0.492977
min          -0.998700
25%          -0.571900
50%          -0.128000
75%           0.352700
max           0.648800
Name: compound, dtype: float64
```

Fig. 25. Sentiment scores of late response tweets

```
prediction0.describe()
```

```
count    269472.000000
mean          0.147963
std           0.344386
min          -0.359500
25%           0.000000
50%           0.000000
75%           0.440400
max           0.999400
Name: compound, dtype: float64
```

Fig. 26. Sentiment scores of early response tweets

From the summary, we see that mean of sentiment scores of late response tweets is negative whereas the mean of sentiment scores of early response tweets is positive.

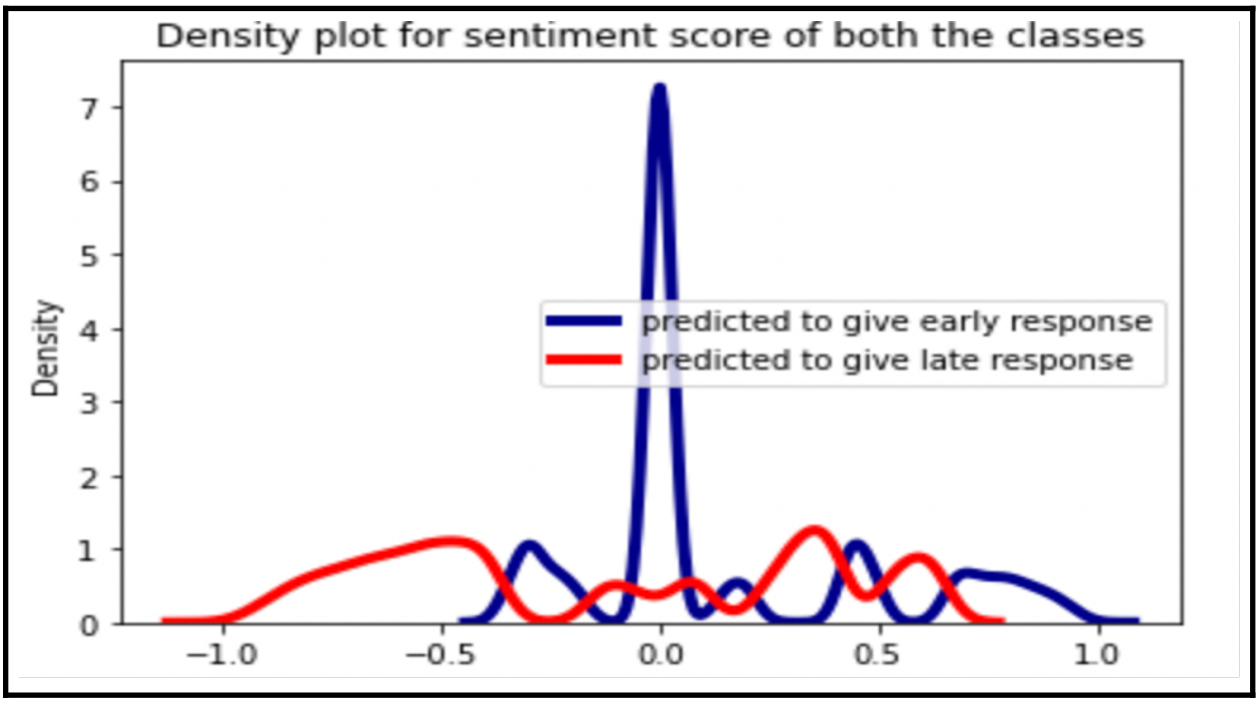


Fig. 27. Density plot of both the groups

From the density plot, we can see that tweets giving early response are concentrated towards the middle region (neutral sentiment) whereas the tweets predicted to give late response have high sentiment values (negative as well as positive) and is pretty spread out.

5.2 Results provided by the model

From our analysis, we found the following results:

•Tweets with negative sentiments provide delayed response
•Tweets with positive sentiments provide early response
•Tweets with low sentiments or neutral sentiments will get a quicker response
•Tweets with high sentiments, either positive or negative will get a delayed response

5.3 Novelty

Based on the results of our project, we have a clear guideline for the type of tweet that a customer has to put in order to get a prompt response from the customer support team. According to our results, a customer tweet should have a low value of positive sentiment in order to get a prompt response. This recommendation of the type of tweet to get a prompt response is the novel contribution of the project.

6. Conclusion

As mentioned earlier, Twitter is one of the best platforms to ask for customer support. Our analysis will aid customers on how exactly to raise the request to the company to get an early response. As more and more customers get prompt responses from the companies, it will benefit the companies as well by improving their brand image.

7. Future Work

For future work, we are looking to include more input features like the length of the tweet. We plan on using ‘string indexer’ and ‘one hot encoder’ to use the text column as a separate feature along with the sentiment score. We are also looking to analyze the response of each company separately and find out the optimum sentiment score and length of a tweet to get a quick response from a particular company.

8. Limitations

One of the limitations we recognized was that our dataset only contained those tweets from the customers which were responded to by the customer support. There are many tweets which are not even addressed by the companies which were missing from the dataset. It would have been interesting if we would have those tweets as well because it would have increased the scope of our project and we would be able to analyze those tweets as well which are not getting response at all.

9. References

1) *Why you need to speed up your social media response time (and how)*. Sprout Social. (2021, June 28). Retrieved April 28, 2022, from https://sproutsocial.com/insights/social-media-response-time/

2) Bernazzani, S. (2020, April 15). *Customer service tweets from 10 brands doing Twitter Support right*. HubSpot Blog. Retrieved April 28, 2022, from https://blog.hubspot.com/service/customer-service-tweets

3) Vector, T. (2017, December 3). *Customer support on Twitter*. Kaggle. Retrieved April 28, 2022, from https://www.kaggle.com/datasets/thoughtvector/customer-support-on-twitter?resource=download

10. Appendix 1

Both Apoorv Awasthi and Bala Akhil Rajdeep Battula contributed equally to this project. Akhil did a lot of work to clean the dataset, doing data transformation and feature engineering. Apoorv handled the ML models, statistical testing, and understanding of the results. Both the students helped out each other and remained in contact throughout the project. When one of the students got stuck because of some problem, the other member chipped in to troubleshoot the problem and get everything working. Report writing was equally divided as well and both students read the document multiple times to avoid errors.