MSBA 6420 Predictive Analytics

Predicting US Airline Sentiment

MSBA Class of 2017

Arpit Sidana
Sandeep Chitta
Russell Nour
Rohit Nadgiri

Table of Contents

Business Problem

Twitter has approximately 313 million active users worldwide and approximately one billion unique visits every month to sites with embedded tweets. In addition to this, we have also observed high professional usage of such social platforms. Given this high popularity of twitter, it becomes an important platform to make or break the reputation of a company through customer engagement. In fact, it has been noted by industry experts that "airline Twitter feeds are the digital equivalent of an airport customer service counter."[1]

The business problem is to classify the sentiment (positive, negative, or neutral) of airline customer tweets for six major U.S. airlines. This will enable us to examine what is being said online about the airlines and to identify potential problems brought forward by customers in their respective airline experiences.

The value added by this exercise resides in the power to classify tweet sentiment in a relatively fast and automated fashion (as opposed to a manual approach), to identify dissatisfied customers early on (including reasons why they provided negative feedback), and ultimately to improve the overall airline experience and reduce customer churn. In addition, marketers can use the information gleaned from tweets to engage with their customers, manage their brand perception online, and for marketing research and competitive intelligence.

[1] *The Turbulent World of Airline Twitter Accounts*, CNN, August 21, 2015
http://www.cnn.com/2015/08/21/travel/airline-twitter-personalities/

Since tweets are notorious for going viral and causing negative PR for companies, early identification can help address customer concerns and thus deal with potentially deleterious situations in an orderly fashion.

Once we have developed a multinomial classification model, we can use our model to identify tweet sentiment almost in real time and address negative customer feedback instantly. Customers expect an appropriate and timely response when they tweet about a negative experience. For example, the American Airlines Twitter account has been criticized for not picking up on a customer's sarcasm. Similarly, American Airlines also experienced public backlash when a popular music artist tweeted his millions of follows about a bad experience with the airline.[2] United and Virgin America have been criticized for taking too long to respond to customers (as well as Southwest, for not even responding at all). It is clear how damaging a negative tweet or late response can potentially be for airlines.

Data

At a broader level, the summary of our dataset for our first model of classifying the sentiment of the tweet is as below,

Number of instances: 14,640

Number of features: 20

Target classes: 3

Type of variables: categorical, numerical, and ordinal.

[2] http://www.cnn.com/2015/08/21/travel/airline-twitter-personalities/

The Data for our project has been retrieved from kaggle competition held for analyzing US airline sentiment. Our dataset consists of 14,640 tweets and attributes like Tweet, airline name, sentiment associated with airline, tweet location, tweet creation time and few more.

Each instance in our dataset represents one tweet.

The aforementioned data was for the first model that we build i.e. to classify each tweet as “positive”, “negative” or “neutral” making it a classification problem. In our dataset, we had the above classes classified manually which made up our target variable.

Moreover, each negatively classified tweet in our dataset had a “negativereason” associated with it. This formed the basis of our second classification model where the target variable was “negativereason” and the target classes were: “Bad Flight”, “Can't Tell”, “Late Flight”, “Customer Service Issue”, ”Flight Booking Problems”, “Lost Luggage”, “Flight Attendant Complaints”, “Cancelled Flight”, “Damaged Luggage” and ”longlines”.

For our second model, the summary of our dataset is as below,

Number of instances: 9,178

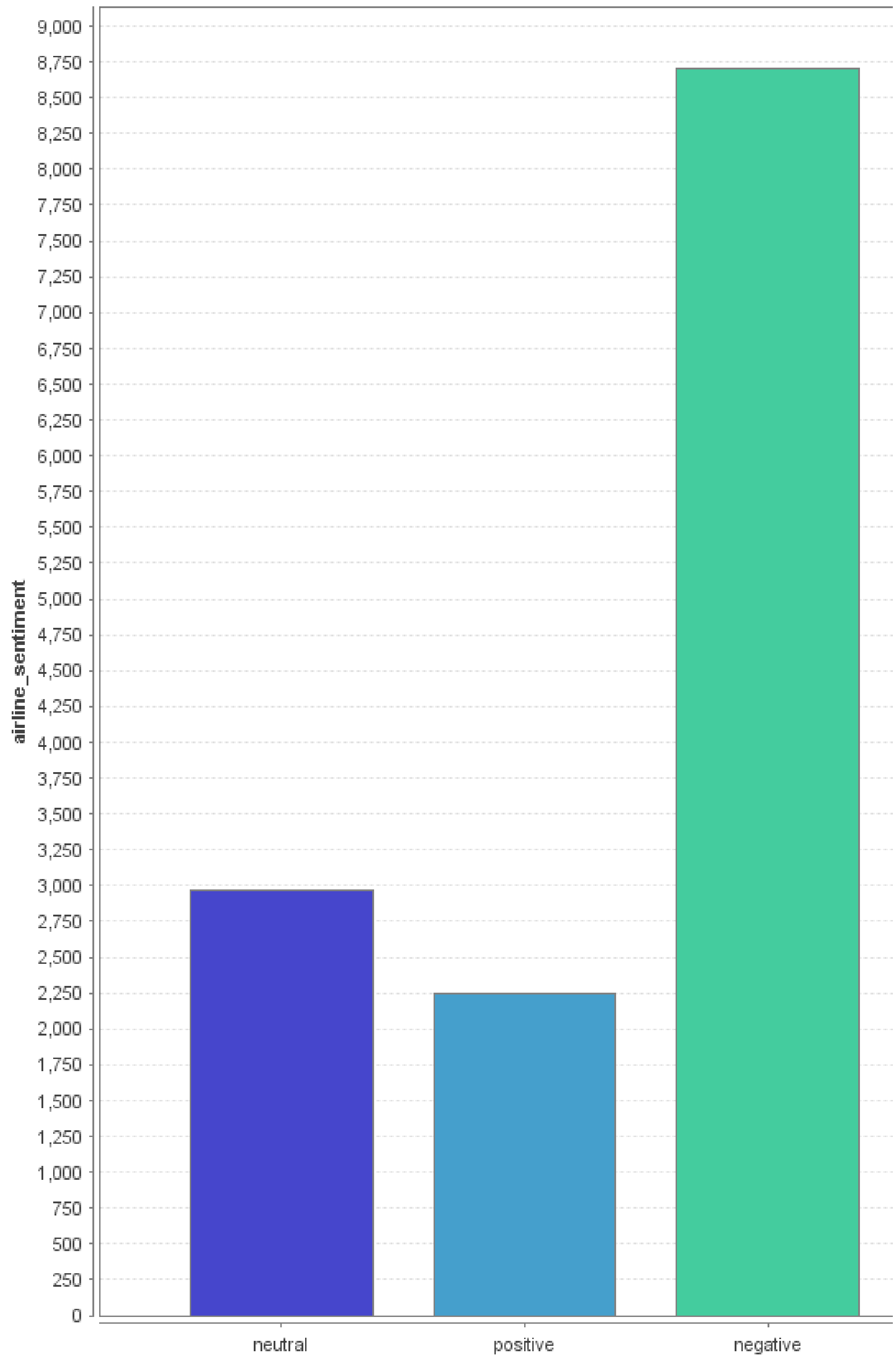
Number of features: 20

Target classes: 10

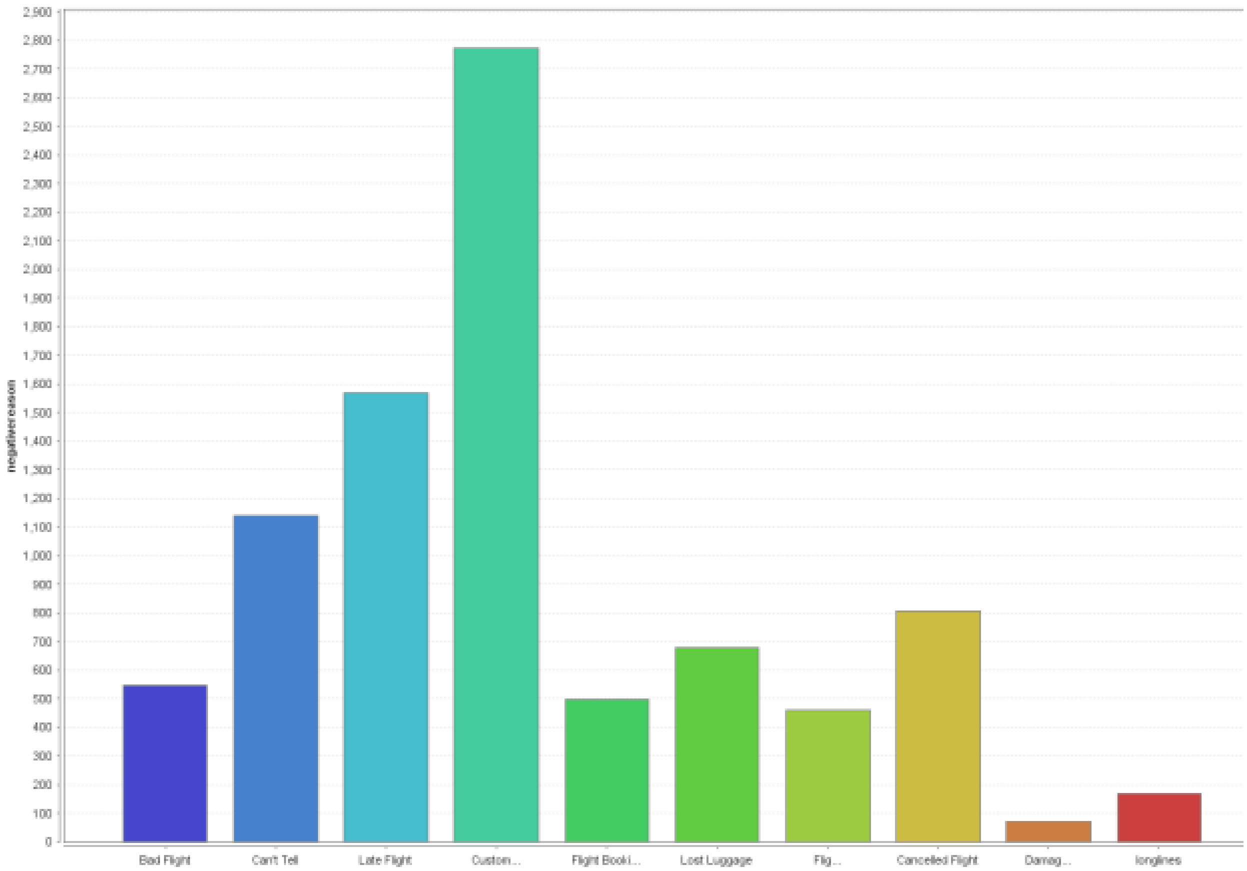
Type of variables: categorical, numerical, and ordinal.

For each of our models, we looked at the distribution of classes of target variable.

For our first model (sentiment):

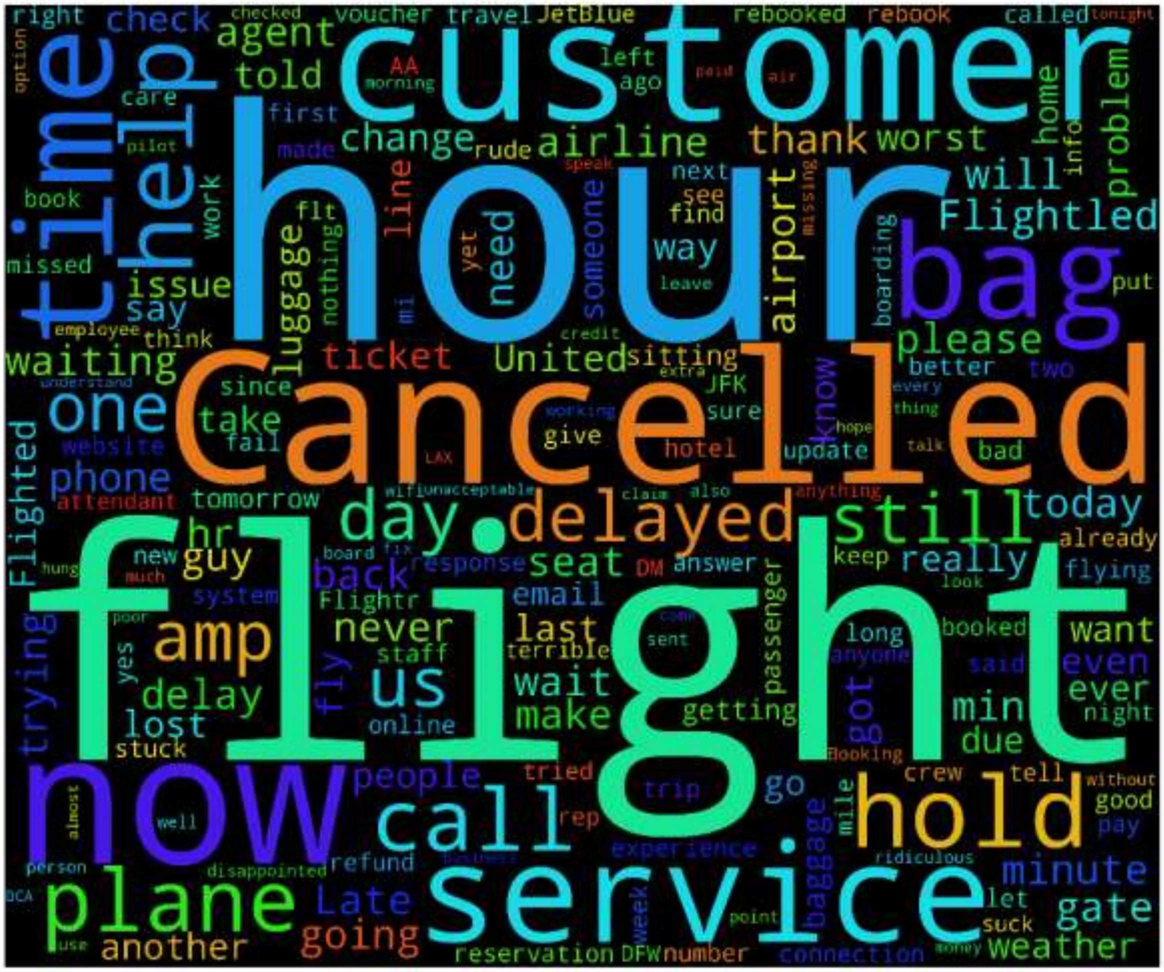


For our second model (negativereason):

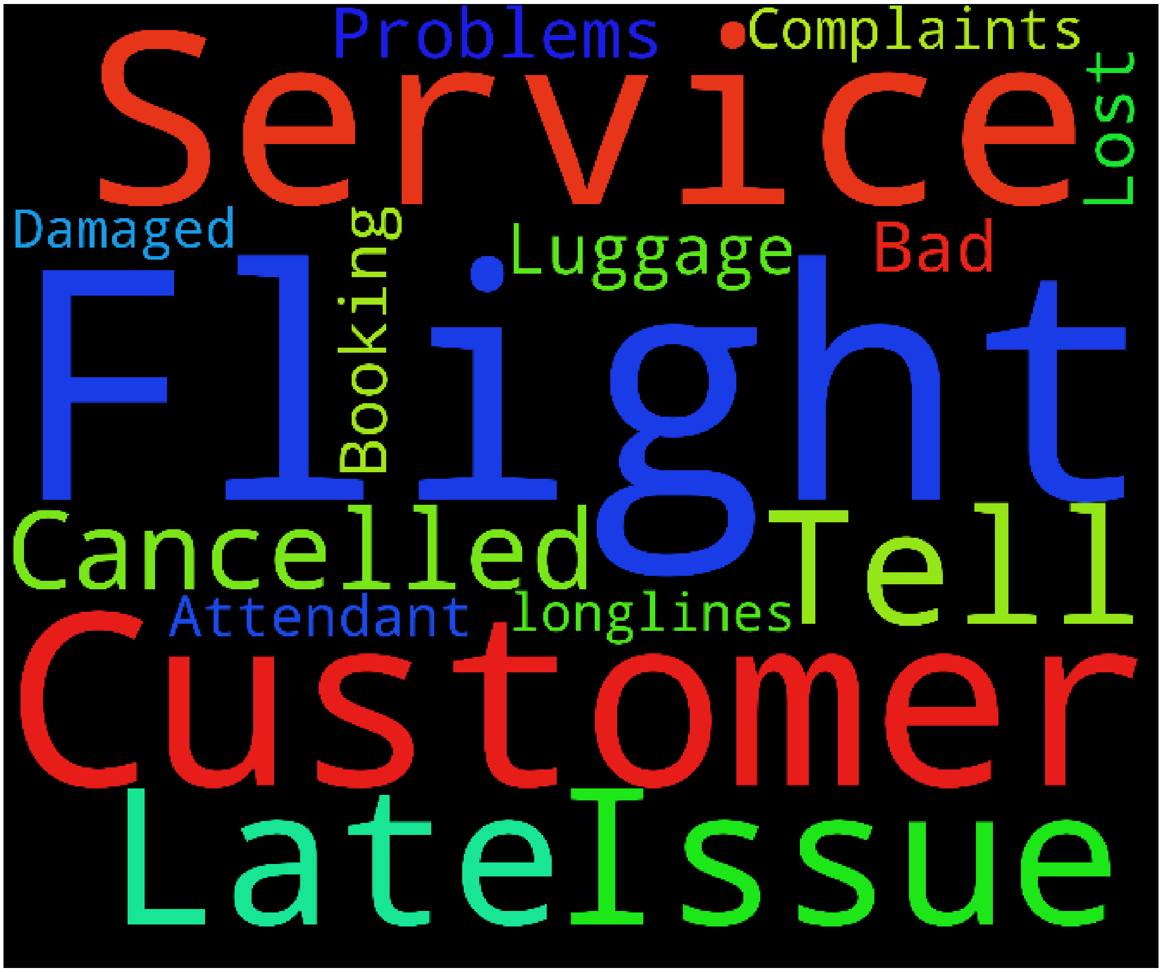


For each of our models, we also looked at wordclouds ("text" (negative tweets) for our first model and "negative reason" classes as per our second model) to gain an overall understanding of the data before we delved into modelling

First model wordcloud:



Second model wordcloud:



Data preparation

As a part of preparing the data for implementing the model we went through the following steps,

1. Imported the relevant Python libraries/packages such as sklearn (classifiers and ensembles) and matplotlib. Note that for our project we also installed external packages such as nltk, seaborn, mlxtend and vaderSentiment and imported their respective libraries.

2. An initial exploration of missing data (as a percentage of total dataset):

```
df.isnull().sum(axis=0)/len(df)*100
```

```
tweet_id                         0.000000
airline_sentiment                0.000000
airline_sentiment_confidence     0.000000
negativereason                  37.308743
negativereason_confidence       28.128415
airline                          0.000000
airline_sentiment_gold          99.726776
name                             0.000000
negativereason_gold             99.781421
retweet_count                    0.000000
text                             0.000000
tweet_coord                     93.039617
tweet_created                    0.000000
tweet_location                  32.329235
user_timezone                   32.923497
```

Based on model deployment objectives, we intended to make a generalized model that could be applied only on the based of the "text" (i.e. the tweet) feature and hence built our model entirely around text and missing values in other features was not a concern.

3. For our first model, we encoded the target variable as "positive":0,"neutral":1,"negative":2 and for our second model we encoded the target variables as "Bad Flight":1, "Can't Tell":2 ,"Late Flight":3, "Customer Service Issue":4, "Flight Booking Problems":5, "Lost Luggage":6, "Flight Attendant Complaints":7, "Cancelled Flight":8, "Damaged Luggage":9, "longlines":10.

4. Below is the distribution of tweets sentiments across six different airlines.

airline_sentiment	0	1	2
airline			
American	336	463	1960
Delta	544	723	955
Southwest	570	664	1186
US Airways	269	381	2263
United	492	697	2633
Virgin America	152	171	181

For every tweet in our dataset we have a column called 'airline' which represent the particular airline customer is referring to. The dataset has six unique airline carriers and we created this column into six dummy binary variables ("American", "Delta", "Southwest", "US Airways", "United", "Virgin America") each cell having a value 0 or 1.

Doing so have the columns "airline_sentiment", "airline_sentiment_confidence", "American", "Delta", "Southwest", "US Airways", "United", "Virgin America", "retweet_count" and "text" for our first model of classifying the sentiments of the tweets. Similarly our second model, they were "negativereason", "negativereason_confidence", "American", "Delta", "Southwest", "US Airways", "United", "Virgin America", "retweet_count" and "text".

Feature engineering and Feature extraction:

We carried out some feature engineering that would help us add more relevant attributes from our existing dataset. We believe that this engineered features would make our model better classify the tweets.

The engineered features have been elucidated below:

Dictionary Words: Using the publicly available dictionary of positive and negative words, we attempted to capture tweet sentiment by creating “Positive_Word_Flag” and “Negative_Word_Flag” as features based on total count of positive and negative words respectively in a tweet in a bag of words fashion.

Sarcasm: After iterating through initial models and analyzing the misclassified tweets manually, we observed that sarcastic tweets tend to be associated with a common strong positive word (among great, thanks, thank and congrats) along with negative words. We then created a “Sarcasm” feature to flag a tweet as sarcastic if it contained one of the aforementioned positive words along with it having at least one negative word count (based on “Negative_Word_Flag”).

Word Sentiment (VaderSentiment): Based on publically available word sentiment scores, we used VaderSentiment to analyze each tweet and based on the words for the tweet, come up with the features “Vader_compound”, “Vader_pos”, “Vader_neg” and “Vader_neu” to capture tweet sentiment scores. The “Vader_compound” captures the emotional intensity of the tweet. Its value ranges from -1 to 1. Words like ‘SERVICE WAS GREAT!!!’ with exclamation marks or with capital words have greater emotional intensity. Similarly, "Vader_pos", "Vader_neg" and "Vader_neu" captures positive, negative and neutral emotional intensity.

UpperCase: We presumed that continuous upper case letters would function as proxy for extreme tone in text (Example: shouting) and therefore developed a new feature that evaluated the uppercase letters. Here we specified the length of continuity to be at least two to capture words such as "NO" and other longer words.

Special Characters: We created a new feature that captures special characters (in continuity of at least length 2) such as !! or ??? that could potentially indicate extreme tone. Based on observations of misclassified tweets in initial model iterations, such extremes tended to be associated more with negative than positive extremes.

Emojis: We also observed emojis in our tweets and in an effort to leverage emojis as a feature, we built ~200 features of emojis based on common emojis used in tweets. Hence each emoji (such as 😟) was counted in each tweet and the respective count was captured in a feature for which the feature name was the emoji itself (😟 in this case). We handled around 183 different emojis(😋~ i am there in the list too).

After adding the above mentioned features, we pre-processed the text data to incorporate the following:

- **Tokenization**: To split the text of each tweet and then combine them into a sequence of tokens (words)
- **Lemmatization**: "(press, 2009) Since stemming is a crude way of treating tokens, we carried out lemmatization that considers vocabulary and includes morphological analysis of words (that aims at removing inflectional endings only)"
- **Stop Words**: To filter stopwords like "a", "an", "the" etc. that do not contribute much to the sentiment and are not potential meaningful features
- **Punctuation removal**: To remove punctuations such as ";", ":", and "," used in the tweets
- **Lower case conversion**: To convert all tokens (words) to lowercase in order to avoid any discrepancies in the tweet text. So that when finding the word frequency both 'happy' and 'HAPPY' are treated the same.
- **Removing most frequent and less frequent words:** To remove words that appear more than 50% of the times in the tweets or less than 2 times (absolute)
- **Removal of alphanumeric and numeric words:** To treat and remove cases in tokens such as "05AM" and "10"
- **Keeping words of length > 1**: To capture only words that have character length of words greater than 1

- **N-gram (bigram) generation**: To generate bi-grams as features using the available tweet text, to develop multiple combinations of bi-grams to add myriad features to our dataset. However, we note that it also increases the sparsity of our dataset.

Evaluation

In the context of our business problem, correctly classifying a negative sentiment tweet is equally important as precision in which we correctly classify them as negative sentiment tweets. Since a business organization has limited customer representatives, we want to improve both precision and recall. This is because misclassifying a neutral or positive tweet as negative will require business to invest in resources that would not yield a good ROI and misclassifying a negative tweet can have serious ramifications. Thus f-measure of negative sentiment tweets is the right metric to evaluate the models; the model with a higher f-measure of negative sentiment tweets would be preferred.

We believe that accuracy is not a correct measure of performance for our model as we could land up with a model that could be highly accurate but misclassifies the negative tweet, which can be expensive from a business standpoint.

Currently in the realm of supervised machine learning, there are very powerful classification algorithms like Decision Trees, Naive Bayes, Linear classifiers like Logistic, SVM with stochastic gradient descent learnings. Choosing an algorithm of these many available algorithms is done by evaluating the mean cross-validation (more accurate estimate of the performance) f-measure of the negative sentiment tweets. We take a standard number of cross validation as 10 and find the mean of f-measure of all these folds to find the algorithm that gives best metric. Other metrics that we considered are below:

1. As a typical use case of the model that we deploy should be able to classify the tweets with their sentiment in pretty much real time, we need to have a model which is simple and computationally inexpensive.
2. On an average, around 6000 tweets are being tweeted every second. This is tremendous amount of feed. So, we need to build a model that can leverage these information, once the tweets are classified, these tweets should be subsequently used in training our model, to better the existing model. So, choosing a model which requires less time to build and an incremental learner is much preferred, leveraging the new tweeted tweets continuously.

The above additional metrics should also be considered while choosing a model for production as it is expected that the business will see gains in performance over time, more meaningful customer engagement, priorities on what to improve based on classified reasons for negative tweets, a reduction of negative tweets potentially going viral, and a more positive brand image through appropriate and timely responses.

Modeling

Initially, we started modelling (for our first model) in Rapidminer. However, we noted that our feature engineering functionality was limited to RapidMiner's offerings thereby allowing only basic text pre-processing.

Our Initial attempt was to develop a basic model using Rapidminer and the same has been shown below,

Given the nature of data, we had to convert the same from Nominal to Text format before preprocessing. The Pre-processing step included procedures like Stemming (Porter), Tokenization, filtering token words by character length and transform cases.

After developing this basic model, we used multiple trial and error methods and have developed the below log of our attempts that captures different f-measures.

We note that the evaluation metric here is the f-measure of class 2 ('negative') since we assume that misclassification cost of class 2 ('negative') is significantly higher as against those for class 0 ('positive') and class 1 ('neutral') since any tweet that is not classified as negative which is actually negative can cause a PR disaster for an airline as it goes viral.

For our modelling process (for first model) in RapidMiner we observe all results on K-fold cross validation where K=10.

RapidMiner Model's summary:

		in-class: negative		
Description	**Accuracy**	**Precision**	**Recall**	**F-Measure = 2*(R*P)/(R+P)**
Naive Bayes, TF-IDF, Stem(Porter)	42.45%	82.79%	40.70%	54.57%
Random Forest, TFIDF, Stem (Porter), no missing	62.51%	62.51%	100.00%	76.93%
Naive Bayes, Term Frequency, Stem(Porter), no missing	42.51%	82.79%	40.70%	54.57%
Naive Bayes, Term Occurrences, Stem(Porter), no missing	42.48%	82.79%	40.70%	54.57%
Naive Bayes, Binary Term Occurrences, Stem(Porter), no missing	42.48%	82.79%	40.70%	54.57%
Naive Bayes, TF-IDF, Stem(Porter), no missing, Costs 25:50:1	47.69%	79.96%	52.43%	63.33%
Naive Bayes, TF-IDF, Stem(Porter), no missing, Costs 20:1	47.70%	79.93%	52.42%	63.32%
W-Naive Bayes Multinomial, TF-IDF, Stem(Porter), no missing, Costs		62.51%	100.00%	76.93%
SVM, TFIDF, Stem (Porter), no missing	65.00%	67.90%	93.01%	78.50%

kNN (k=4), weighted vote, TFIDF, Stem (Porter), no missing	58.18%	72.28%	75.15%	73.69%
SVM, Weight by IG, Select by weights,TFIDF, Stem (Porter), no missing	67.32%	68.37%	98.33%	80.66%
SVM, TF-IDF, just airline sentiment and text attributes, Costs	69.57%	74.13%	90.89%	81.66%
Logistic(SVM), TFIDF, Stem (Porter), no missing	63.98%	68.33%	91.77%	78.33%
LibSVM+Poly by Binom, TFIDF, Stem (Porter), no missing	62.51%	62.51%	100.00%	76.93%
SVM(Linear)+Poly by Binom, Weight by IG, Select by weights,TFIDF, Stem (Porter), no missing	64.70%	67.60%	93.92%	78.62%

For our first model, we then proceeded with modelling in Python (iPython notebook) to facilitate complex feature engineering and a variety of classification modelling techniques.

In iPython notebook, we split the data initially to retain only relevant features (as explained in Data Understanding) and proceeded to measure f-score of class 2 ('negative') as explained earlier.

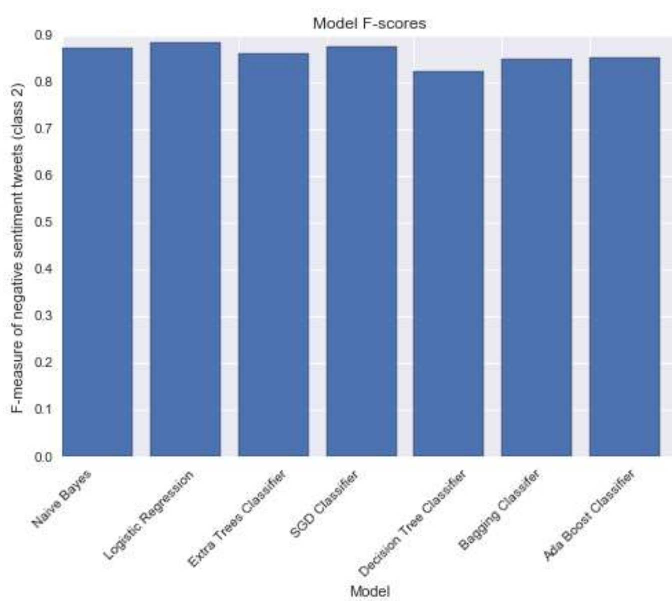
We have many classification algorithms. Some of them which we tested are mentioned below-

1. Naive Bayes
2. Logistic Regression
3. Extra Tree Classifier
4. Linear Support Vector Machines (SGD Classifier)
5. Decision Tree
6. Bagging Classifier
7. AdaBoost Classifier

Choosing the best algorithm of the many above algorithms:

We used cross validation of 10 folds and used the mean f-measure of negative sentiment of all the folds as a metric to gauge the performance of the above algorithms. The advantage of using this approach is to choose an algorithm that is best for all the data together.

Below is the plot for the performance of all the algorithms we tested for our first model(classification of tweets sentiment).



Below we have the confusion matrix for Logistic Regression and Naive Bayes model and summary statistics when they are tested on 20% testing data(80 % training dataset). Training and Testing datasets are split using stratified sampling

```
CONFUSION MATRIX
```

PREDICTED	0	1	2
ACTUAL			
0	334	54	71
1	65	347	168
2	41	134	1714

```
None
Accuracy of model is 0.8179644809
Precision of Positive class('negative sentiment') is 87.7624167947
Recall of Positive class('negative sentiment') is 90.7358390683
The F-measure of the Positive class is 89.2243623113
```

Below we have the confusion matrix for Naive Bayes model and summary statistics.

```
CONFUSION MATRIX
```

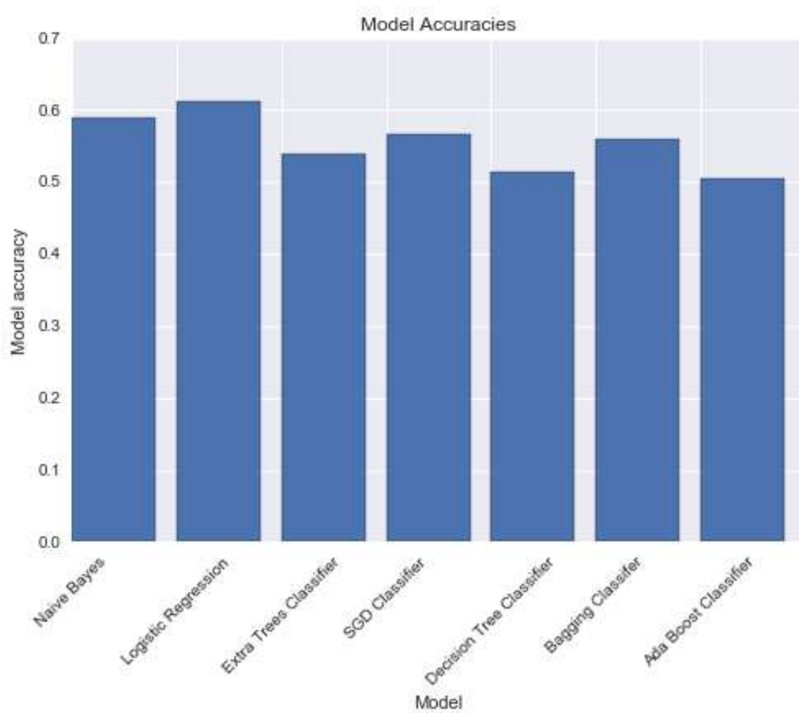
PREDICTED	0	1	2
ACTUAL			
0	318	42	99
1	42	280	258
2	37	87	1765

```
None
Accuracy of model is 0.8070355191
Precision of Positive class('negative sentiment') is 83.1762488219
Recall of Positive class('negative sentiment') is 93.4356802541
The F-measure of the Positive class is 88.0079780603
```

From the confusion matrix of logistic and naive bayes, we observe that negative sentiment F-measure of Logistic is slightly better than that of Naive Bayes. Despite F-measure of Logistic being better than Naive Bayes, we select Naive Bayes over Logistic Regression as a final model for the production, the reason being that the Naive Bayes is computationally inexpensive and it is an incremental learner. So, choosing this model will give swift predictions as classifying the tweets instantly is of paramount in our business context.

For our second model, we followed a procedure similar to that for the first model in terms of data preparation, pre-processing, feature engineering, and same modelling techniques. The metric on which we gauge the performance for our second model is accurate as all the negative reasons while classifying are equally important.

We plot models based on accuracy for second model (10-fold cross validation):



Here too we choose Naive Bayes over Logistic regression as the accuracy difference between Naive Bayes and Logistic regression is not substantial and more over if a negative sentiment tweet is wrongly classified with a different negative reason, internally the customer support system is well equipped to handle this situation and these can be swiftly transferred to right customer representatives.

The classification matrix for the Naive Bayes model (for the second model):

PREDICTED	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
ACTUAL										
0	11	11	39	37	1	1	3	0	0	0
1	4	81	44	108	0	2	2	3	0	0
2	1	6	265	44	1	2	0	6	0	0
3	0	28	34	524	3	3	2	13	0	0
4	1	5	8	71	11	0	0	5	0	0
5	0	10	27	52	0	54	1	0	0	0
6	1	8	20	50	0	2	5	1	0	0
7	1	8	18	27	0	2	0	121	0	0
8	0	3	1	4	0	3	0	0	0	0
9	1	1	17	16	0	1	0	1	0	0

```
None
Accuracy of model is 0.5838779956
```

Apart from the advantages mentioned above for Naive Bayes, below points cover the advantages and disadvantages of all the algorithms that we tested in much more detail

Pros and cons of different models

Advantages of Naive Bayes:

1. Performance and scale matter in many real world problems. Naive bayes is often "good enough" in a lot of real world applications as it is efficient in terms of both storage space and computation time
2. It is an incremental learner, so with the arrival of more and more tweets, the model can be continuously improved

Disadvantages of Naive Bayes:

1. Non-accurate class probability estimation. But in our case of classification, this is not going to be any issue as when we want to know, which particular sentiment does a tweet fall under

Advantages of Logistic Regression:

1. “Low variance
2. Provides probabilities for outcomes. Thus, using domain expertise threshold can be varied to improve classification accuracy
3. Works well with diagonal (feature) decision boundaries

Disadvantages of Logistic Regression:

1. High bias

Advantages of Decision Trees:

1. Easy to interpret visually when the trees only contain several levels
2. Good for tweet data as it can easily handle qualitative (categorical) features
3. Works well with decision boundaries parallel to the feature axis

Disadvantages of Decision Trees:

1. Prone to overfitting
2. Issues with diagonal decision boundaries"[3]

"Advantages of AdaBoost Classifier:

1. No Prior knowledge (of weak entity) is required
2. Simple, quick and easy to implement

Disadvantages of AdaBoost Classifier:

1. Sensitive to Outliers
2. Vulnerable to uniform noise"[4]

Advantages of Linear (SVM) Stochastic Gradient Descent:

1. Efficiency
2. Ease of implementation

Disadvantages of Linear (SVM) Stochastic Gradient Descent:

1. SGD requires a number of hyperparameters such as the regularization parameter and the number of iterations.
2. SGD is sensitive to feature scaling"[5]

[3] https://github.com/ctufts/Cheat_Sheets/wiki/Classification-Model-Pros-and-Cons
[4] http://math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Eric-Boosting304FinalRpdf.pdf

Deployment

Use Case

At any given interval of time, the use case of the model developed would be to classify the sentiment of the tweets tweeted by airline customers and classify the reason for their negative sentiment. After classification, the negative sentiment tweets are assigned to our customer support team and our customers are addressed accordingly. This model once deployed with big data technologies like Apache Spark streaming, Kafka that supports real time analytics would be easily scaled up to support the real life problem with customizable batch interval depending upon latency requirements.

Issues/risks associated with the model

There are some inherent risks that are associated not only with this particular Naive Bayes model that we developed but also for any text classification model as it is challenging to capture the contextual meaning of all the tweets. As capturing human emotion/reason behind the tweets of maximum 160 characters can be done by humans accurately and these accuracies cannot be matched with existing machine learning algorithms. Below are the additional issues that needs to be addressed by the business before deploying the model to production.

1. How do we capture the tweets that are specific to our context (customers of airlines)? We cannot process all the tweets happening in the twitter as that would be plethora of a tweets and practically it's not feasible to classify all these tweets. We would classify only the tweets

[5] http://scikit-learn.org/stable/modules/sgd.html

that contain the tags of the airlines with proper '@' format (@AmericanAirlines etc.). In doing so, there is good chance that if the customers doesn't tag the airlines exclusively with '@' in their tweets, then these tweets are not being processed by our model and have a potential chance of becoming viral and bring huge losses to the airlines

2. The firm should be aware of the possibility that a model won't be able to correctly classify every single tweet accurately. A negative sentiment tweets might be classified as positive/ neutral and vice versa, especially when sarcasm is involved since it is notoriously difficult for a machine to detect. Over reliance on the model might lead to misclassifications slipping through the cracks. It might be prudent to manually double check a few of the tweets every once in awhile as a quality control to ensure the model is still performing (this could also serve to potentially catch a few that slipped through the cracks). It may also be prudent to retrain the model on new historical data in order to ensure that the model does not become outdated and it keeps up with changes in the business and in how customers tweet.

Ethical Concerns

There are very few ethical concerns associated with the use case of the model. Users tweet publicly, especially if they use an airline's twitter handle or a related hashtag, and therefore have no expectation of privacy - tweets are not considered private communications.

However, one potential ethical concern is customers using the Twitter communication channel to "game the system" by fabricating complaints in order to receive perks from airlines. Similarly, it is possible to create a perception of unfairness depending on the response to different Twitter users and among those customers who do not use Twitter for their complaints. Although this model is exciting, it is important to treat all customers complaints equally, no matter which communication channel they come through.