David Lattimer

DSC 680

7/22/2021

Disneyland Reviews

Overview:

With websites allowing customers to leave reviews and rate products, trips and more, there is far more data to be found as long as we know how to use it. If we can take these text reviews and stories/testimonials from people that have been kind enough to leave us this information, can we use them to learn what people are having problems with? Can we use the text from their reviews to predict the rating they are going to give? And what else can we leverage from these text reviews to help make decisions and learn things that would otherwise be ignored? If we can take reviews of text as they are coming in, then change the things that people aren't enjoying, can we learn this quickly so that we can make a change and make future customers happy?

With this dataset, I will be looking at a batch of reviews from TripAdvisor left for three different Disneyland theme parks; Disneyland in California, Hong Kong Disneyland and Disneyland Paris. I will be trying to build a model that takes the reviews that are left to try to predict the rating (1-5 stars) that was given with the review. By taking several approaches, such as sentiment of the review, or using the tf-idf (which finds the most important words in the reviews at predicting the rating), we can find the most accurate way for us to predict the scores that will be given and can use to predict future text reviews, or to predict future text reviews that may not have a rating tied to them. Outside of the model, we can explore the dataset to figure out who is leaving these reviews, why they might be leaving them and how we can make meaningful changes as quickly as possible and fix any problems before it affects too many guest's experiences.

Defining the Problem:

There is an increasing amount of unruly and hard to use data that may be more important to use than the data we are able to collect in person. If we can find a way to use this data and make decisions based on the reviews people are leaving, then we can make their experiences better and leverage the data into both new customers and loyal guests that want to keep coming back for more. There are tons of ways to look into these reviews and some will be better than others in terms of usefulness and what we can glean from them. If we can find a way to utilize these reviews, we can answer many problems and learn a lot all from the same set of reviews. Do we want to know about the rides people are talking about and focusing on? We can do that. Do we want to focus on lower rating reviews to figure out the sorts of problems people are having? We can do that as well. Do we want to look at the high rated reviews so that we can continue to do the things that make people the most happy? We can do that as well. Should we focus on park specific issues and figure out how we can perform better from each park? Or should we look at the dataset as a whole with all three parks and figure out if there are any underlying issues that encompass the theme parks division of the company as a whole? Do we want to separate all the reviews into their individual ratings and try to predict those or would it be better to throw all the reviews under 4-stars into a “problematic” category and treat them all as the same in terms of wanting to improve those experiences specifically? Because there is so much information to be found in a group of 42,000 reviews, there needs to be a focus on what is important and what questions you want to answer with what can be found.

This project will focus specifically on building a model that can reasonably predict the ratings from the text reviews, and then will do some initial exploration into the data that is involved in the dataset. More specific problems and projects can easily be created based on

what is needed to be learned and what decisions need to be made, but starting with a model to predict those ratings will always be useful and there are many ways we can leverage the data in creative and useful ways for whatever is needed at the time. This also allows us to use future reviews in order to learn more of the problems people have and could possibly apply this model across multiple websites and classify these reviews even if there isn't a rating system on the website. By using reviews to learn about what people want, how we can fix it and how we can improve the parks as a whole, we can constantly be checking for issues people have and get in front of the issues as quickly as possible. By adapting this model into being able to find these problems and focus on the specific areas that are being talked about in the reviews, we can learn a lot more valuable information from this data than simple number records and data that we are collecting separately in the parks.

Data Understanding:

This dataset was fairly simple with few issues. The initial dataset has the following variables contained within it: Review_ID, Rating, Year_Month, Reviewer_Location, Review_Text and Disneyland_Branch. Review ID is the unique ID given to each review, which is not helpful for us in any way besides locating duplicate reviews that we need to remove from the data. The rating that was given with each review will be the target variable which we are trying to predict and what we will use to compare our model to see if we can make it accurate. The Year_Month variable just tells us when the review was written. This helps us see when the reviews were written and can allow us to phase out some of the concerns people had for issues that we were able to fix or that no longer are applicable because the reviews are too far in the past. This also allows us to examine more recent reviews so that we can see how they respond to changes that we make or to see new problems arise.

Next we have the reviewer location which I feel has some usefulness, but is a lot harder to implement. When brainstorming, I thought it would be interesting to see if the distance from the location to the park they are reviewing led to lower or higher scores because of the expectations that needed to be met. This was problematic because the country that it is written from doesn't pinpoint the reviewer very well (we could have two reviewers, one in New York, the other in Anaheim, and both would appear as the same "United States"), and also there is an implicit bias in the reviews on TripAdvisor based on which countries use the website and write the reviews. Because the website is taking reviews in English and is entirely in English, using the website, and especially writing reviews about the trip may not be helpful for a lot of people. For example, we have over 9,600 reviews left on TripAdvisor for Hong Kong Disneyland, but only 554 of the reviewers are from Hong Kong. This is in contrast to the 19,400 reviews for California Disneyland and the 14,500 reviewers that are in the United States. This is to say, we have more people from English speaking countries leaving reviews across all the parks than we have people in Hong Kong reviewing their own home park. That's not to say these are useless or anything, we are just missing the opinions of people visiting by only taking data from TripAdvisor. This could also allow us to find places where people in China/Hong Kong are leaving reviews and get a better picture of some of the problems in Hong Kong Disneyland. Unfortunately, the location is too vague to find distances to the parks to figure out the distance from the reviewer to the park to see if people from closer or further had interesting trends.

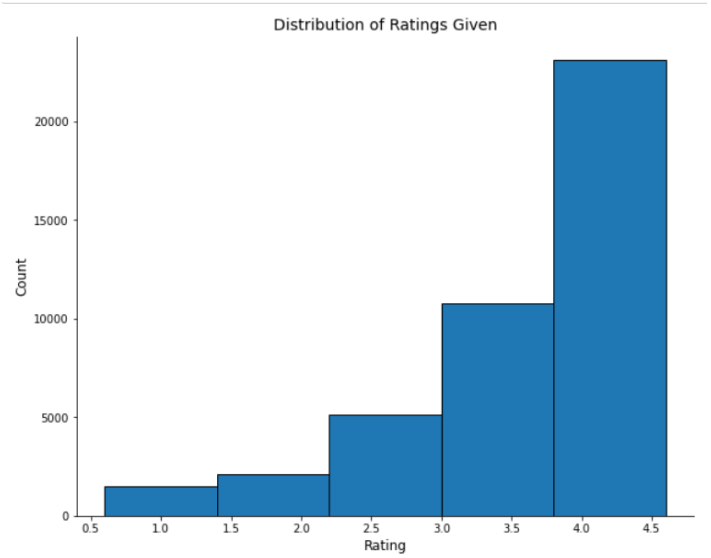
The next variable we have is the review itself, which obviously is what we will use to predict the rating that is given. This is the most important aspect of predicting the rating and the only one, with some variation, that we will be using to measure the rating we are expecting and compare it to our target variable. The last variable is which Disney park the review is about, whether it is Disneyland in California, Hong Kong Disneyland or Disneyland

Paris. This is helpful to try to find if our model handles a certain park well or if there is one that is harder to pinpoint how they review.

Data Preparation:

There was not a ton of preparation needed for this dataset. We needed to remove some duplicate reviews, which were fairly easy to remove, and then deal with missing dates. I decided not to completely remove them from the data since the only variables we needed from the modelling were the review and the rating given, so it was fine to leave the dates as missing. For EDA and plotting any variables over time, we removed those missing dates and were not able to use them, but that was only something to look out for when doing anything with dates and stuff like counts of posts over time.

As for creating variables, I ran the text reviews through TextBlob to measure the polarity of the sentiment, and then used this to estimate the rating that was given. By using a similar distribution of the reviews, which leaned heavily toward 5-star reviews, as shown here. This makes it a lot easier for our models to predict the 5-star reviews, because we have so many of them, but hard to predict our lower rated reviews because we have so few 1- and 2-star reviews.



The text review column was also prepared to run through a Term Frequency — Inverse Document Frequency (tf-idf) method, which takes the most important words that appear within the reviews, assigns an importance in how they determine the ratings from the training set and use them to predict the rating of the test set. First, we clean the reviews so

that punctuation is removed, and we remove stop words that add no real importance and are way too common to offer us any insight (such as “the”, “and” or “was”). Once we have the reviews clean of this, we take the most impactful words using the tf-idf method and can use them to learn how impactful they are, how their occurrence correlates to each rating and use them to predict what the rating is based on the review. Using the tf-idf, I used both 1-2 n-grams, which will look at both the importance of each individual word, as well as combinations of two words to help us predict the rating given.

Modelling:

There were several modelling approaches I used to try to predict the rating and see which method was the most effective in predicting the ratings that were given. The first method used was fairly straight forward, and although ended up being decently accurate, required information we may not have in future models.

For this model, I ran the reviews through a sentiment analysis library and measured the polarity of each review on a scale from -1 to 1. From here, I used the same distribution from the final ratings that I had, so that I could find the cutoffs of sentiment that followed this same pattern. Without doing this, it would have been impossible for the sentiment to match the way the ratings were actually distributed because there were a large number of 5-star reviews and very few 1-star reviews. Because of this we could build a model to help us predict these reviews, but it would be hard to apply to future incoming reviews because the rating disparity might change at some point. With this method, I was able to find the cutoffs to end up with both an actual and predicted distribution of ratings that matched, although that doesn’t mean that the predictions were always right. With this method, I was able to find an accuracy of 51%, which isn’t great but gives us a benchmark.

This also allowed me to do some adapting of the model so that instead of splitting it by each ranking, I could set it up so that 4-5 star reviews were on their own and all other

reviews were separated. This would be more of a “passing” review vs “failing” review, and it would allow us to either focus on the things people are talking about in the bad reviews, while getting insight on what is making people the happiest. By segmenting the ratings into these passing/non-passing classifications, we are able to predict the category it falls into with an accuracy of 81%. A lot better, but also could be due to a couple factors we will discuss later.

The next couple of models that I built used a tf-idf, which takes the most important words within the reviews in order to find how they typically contribute to the rating that is in the training set. Words used for one-star reviews are typically a lot different than the words people are using in five-star reviews, and if we can find words such as “horrible” or “awful”, we would predict these are most likely in low-starred reviews. By finding these important words, then scanning the reviews for how often they appear, we can run a model to identify and estimate the rating tied to a review. Again, if we were to split the ratings into two groups, we would be far more accurate and be able to identify passing/non-passing reviews, but for now we will focus on how close we can get to the exact rating number given to us, especially with a baseline of 51% we got from looking at sentiment.

The first model that uses the tf-idf vectorizer is a simple logistic regression model. This takes each of the words that we have determined to be important and assigned a value to each of these words, sometimes positive and sometimes negative. When the model takes a review, any time a word appears, it ups the count in the column for that word. From there, it adds up the occurrence of each word multiplied by the multiplier for the weight of that word in the model, and then gives us a number that gets sorted into a rating between 1-5. Words that are typically negative will cause negative numbers in our model, and positive words will produce positive numbers, where in the end each review gives us a number that correlates to

a rating. Running the model, I got an accuracy of 62% which outperforms the sentiment analysis and gives relatively accurate results.

The next model that I used was the multinomial Naïve Bayes classifier, which uses the same technique, but instead of having a weight for the review based on the words and then using rounding to classify the prediction into the 1-5 rating system, would just place the words into most likely to be in each of the 5 star ratings and then estimating based on that. Now unfortunately, this model went a little rogue and decided to… well, take a look. Our model has predicted essentially all of the ratings to be 5-star, and a couple 4-star. As we discussed earlier, most of our ratings are 5-star ratings and the model has decided that instead of guessing, if we just say everything is 5-star, we still have a decent performing model. Sadly, this is not something we want to do and provides us with very little insight that we didn't already know, and isn't needed since a human could have spotted that problem from the start. Sadly this performs better than my modified sentiment analysis, but still, not what we want.

	precision	recall	f1-score	support
1	0.00	0.00	0.00	729
2	0.00	0.00	0.00	1070
3	0.00	0.00	0.00	2564
4	0.40	0.00	0.00	5419
5	0.54	1.00	0.70	11540
accuracy			0.54	21322
macro avg	0.19	0.20	0.14	21322
weighted avg	0.39	0.54	0.38	21322

The next model is the decision tree model, which typically separates data based on strict guidelines which split the reviews based on criteria that it notices. This would include a lot of decisions that branch in many directions, especially with the vectorizer of the words producing essentially a dataframe with thousands of columns with 0s and 1s for each review. This means there are many places for the data to be split and sort the ratings based on what is found, although building just one tree may be the downfall here. Using this model we got an

accuracy of 47%, the worst so far. But this has the potential to be expanded upon and made better, which we will delve into.

The last model used is the random forest classifier. Basically this makes many decision trees, and then based on running the review through each tree will decide which rating it is most likely to be a part of. Random forests should always outperform decision trees because it finds different ways to segment the data and creates a way to account for errors and shortcomings that one will run into. This model results in an accuracy of 55%, which is much better than just one, but still not close to our first model of just a logistic regression model.

Conclusion:

By using reviews to try to predict the rating, there are tons of ways we can use the information, explore the problems and fix things before more people have similar experiences. By identifying the rating, we can use this to only pull reviews that would classify as negative, use them to find what people are saying and come up with ways to fix the issues. This dataset allows us to locate parks found in individual parks or the company as a whole, to use the dates provided to see what more recent problems are and overall just solve a ton of problems by automatically finding the reviews we want to find.

By running these reviews through several models, the most accurate to predicting the rating was a logistic regression model surprisingly, and we would expect around 61% of the reviews to be predicted correctly, and would perform even better if we split the reviews into a pass/fail type system for the ratings. Although these models would constantly be changing and don't really provide any actionable decisions to be made, we can still use the data in a ton of ways and find just about anything we are interested in, and then make choices from there.

One of the big drawbacks to this dataset was how lopsided the ratings were distributed. We had more 5-star reviews than we had all the rest of the reviews combined.

This seems to be a problem with using most review data, that people usually leave reviews on good experiences and don't on bad ones. This is not the place for philosophical reasonings, but having this disparity in the data makes it easier for our models to identify 5-star reviews and increasingly more difficult as we move down the scale. Although there are ways to work around this, such as selecting equal numbers of each rating, this will always be an issue and we will always suffer from this problem when trying to glean insight from reviews.