Analyzing Disneyland Reviews

Alfred Prah, Umang Chaudhry

Table of Contents

Introduction/Motivation **2**

Problem **2**

Methods **2**

Exploratory analysis 3

Modeling 5

Data Cleaning and preparation 5

Binary Classification 5

Results: 5

Multi-Label Classification 6

Results 6

Exploring wrongly-classified reviews 7

Conclusion **7**

References **8**

Introduction/Motivation

The dataset that we selected for our final project is the Disneyland Reviews dataset that is available on Kaggle. The dataset includes over 40,000 reviews of three different Disneyland branches that were posted by visitors on Tripadvisor between 2010 and 2019. The branches included in the dataset are Disneyland California, Disneyland Paris and Disneyland Hong Kong. Information in the data includes the rating, the text of the review, the year and month in which the reviewer visited the park, the reviewer's country of origins and the location of the park that they were reviewing. There are now over 5 different Disneyland parks across the world. Each of these parks is unique and offers different experiences to visitors. In addition to different themes, rides and shows at these parks, there are a lot more factors that can impact visitors' experiences when at these parks. This dataset provides an opportunity to examine visitors' experiences at three different parks and determine which park offers the best experience for visitors.

Problem

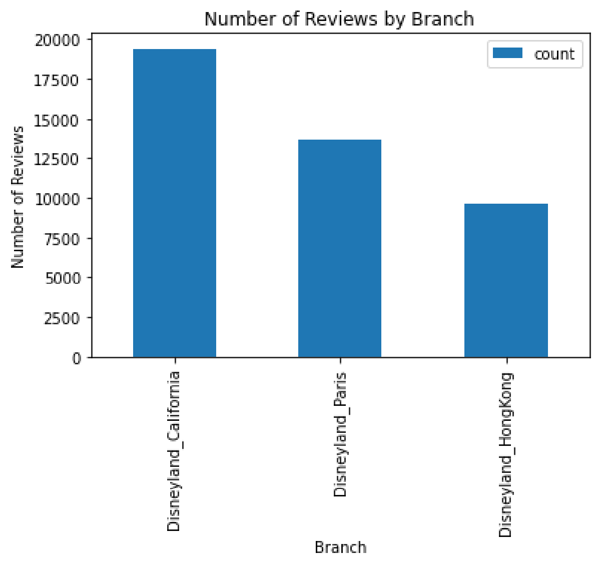
Part of our motivation for choosing this particular dataset was our interest in Natural Language Processing methods. This is a field of study that is extremely powerful that allows us to extract valuable information from text data. An analysis of the text of the reviews along with information about the users and the rating they gave each park will enable us to better understand what makes for a good experience at each of these parks. We aim to develop a model that will predict the rating that a user may give based on their review. THis will help determine the characteristics of the reviews that make for good and bad ratings. Additionally, we will use conventional approaches to perform exploratory analysis on the data to determine on average which park offers the best experiences on average, and determine the best time to visit these parks.

Methods

We performed all of our analysis in python using Google Colaboratory. Analysis was done using a combination of Apache Spark, Pandas and Matplotlib packages. Our analysis was done in two steps. To first get an understanding of the average experience at each of the parks, we spent some time performing exploratory analysis of the data available to us. Once we drew some conclusions based on our exploratory analysis, we worked on our modeling approach to develop a predictive model that would determine the rating based off a review's text.

Exploratory analysis

The first thing we examined was ratings and reviews across parks. Graphs for the same can be found in figures 1 and 2 below. We noted that Disneyland California had more reviews than either of the other two parks by a very large margin. In fact, The number of reviews for Hong Kong was less than half of those for California as can be seen in figure 1. Disneyland California also had the highest rating of all three parks at 4.4 followed by Hong Kong at 4.2 and Paris at 3.96.



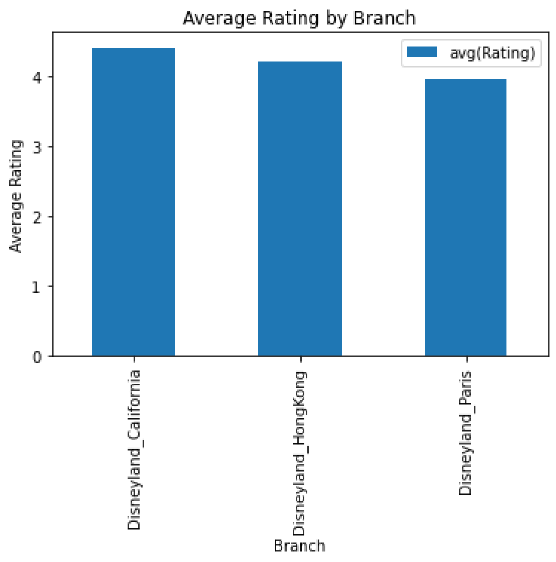


Figure 1: Number of Reviews by Branch

Figure 2: Average Rating by Branch

The next thing we examined was the average ratings and the number of visits (we used the number of reviews as a proxy for this metric) by Month to better understand the distribution of ratings across the year, and also to get an idea of when would be the best time to visit each of these parks. The results of this analysis can be seen in figures 3 and 4 below. We again found that California had better ratings throughout the year than any other branch, and that Paris was consistently ranked the last. We also noted that both Hong Kong and Paris saw a sharp decline in their ratings for the month of August. The ratings did however go back up to their average over the subsequent months. California had the most stable ratings, but in general saw a decline in the winter months from October through December. When examining the number of visits per month, we found that California had the most number of visits throughout the year, followed by Paris and then Hong Kong. What was particularly interesting was that despite having the lowest rating in August, Paris had the highest number of visitors in that month, reaching the same number of visits as the California location which was followed by a steep decline in the subsequent months.

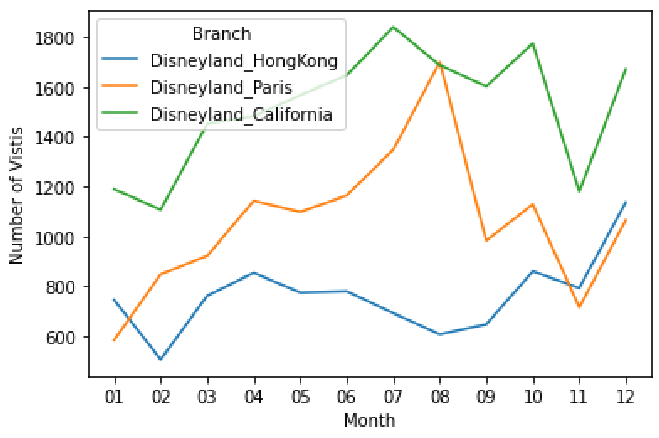


Figure 3: Number of Visits by Branch per Month

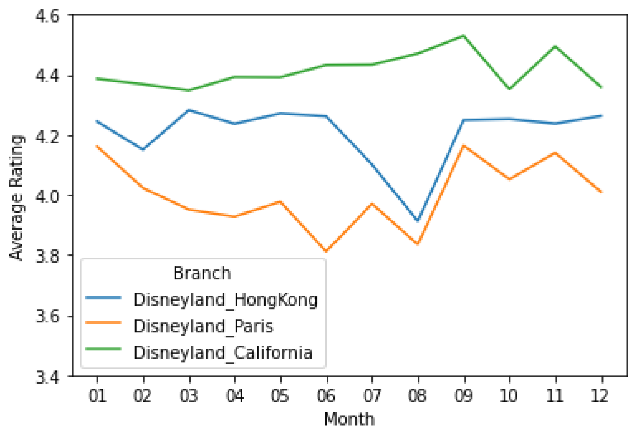


Figure 4: Average Rating by Branch per Month

Based on our analysis so far, we determined that Disneyland California is by far the best park to visit out of the three. It consistently has the highest ratings. We believe that the best time to visit this park would be in November. This month has one of the highest ratings across the year and also has the lowest number of visitors, making for a great experience with shorter lines. Next on our list would be Disneyland Hong Kong. The best time to visit this park would be early in the year, as the ratings are fairly good around this time, and the number of visitors is the lowest. Finally, the third and last option would be Disneyland Paris. We would recommend visiting this park in September as the rating is the highest and the number of visitors is the lowest. We would strongly recommend to not visit this location in August as the ratings are the lowest and the number of visitors increases sharply to meet that of the California location.

The next part of our analysis included examining the reviews themselves. To first get an idea of the types of reviews, we analyzed the length of reviews by park and by rating. The graphics for this can be found in figures 5 and 6 below. We generally found that people had the most to say when visitors rated a park poorly. We also noticed that rating lengths for the Paris location were the highest regardless of the rating.

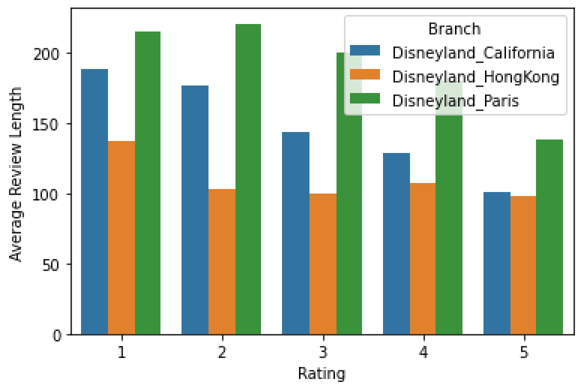


Figure 5: Review length by Branch and Rating

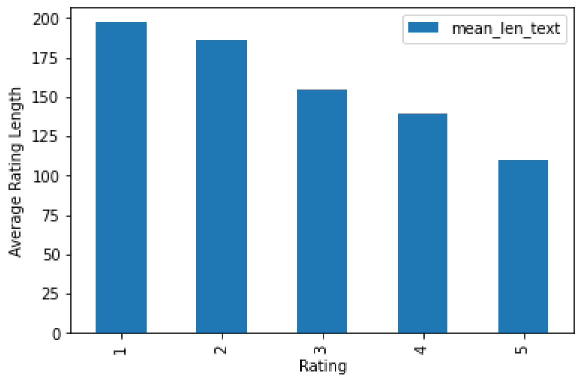


Figure 6: Review length by Rating

Modeling

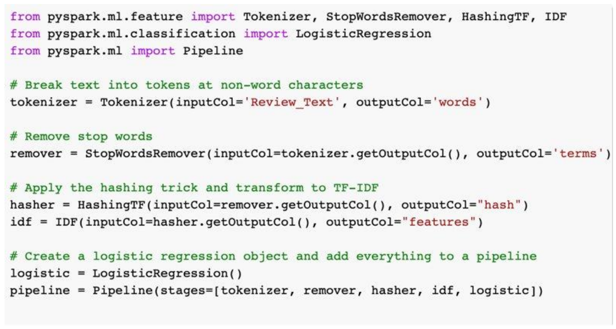
Data Cleaning and preparation

We invested a lot of time in cleaning the Review_Taxt column of our dataset. Some of the data cleaning techniques used were

- Removing punctuations and numbers (regexp_replace)
- Removing stop words (StopWordsRemover)
- Applying the Hashing Trick (HashingTF)
- Converting hashed symbols to TF-IDF (IDF)
- Splitting data into 2: training set and test set (tf_idf.randomSplit)
- Creating a pipeline to automate the process

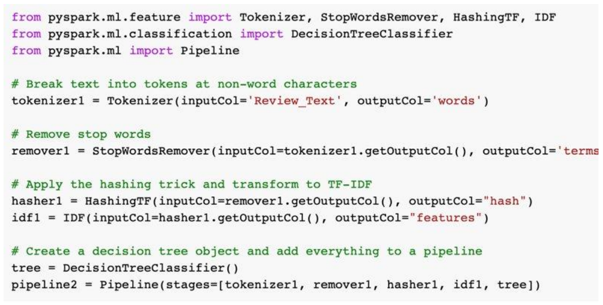
Binary Classification

We created a Binary classification model to determine whether a review is good or bad. To do this, we had to make the following assumption: a Rating of 3+ is good and a rating of 1 or 2 is bad. This allowed us to set a threshold for Ratings, where a good rating = 1, and a bad rating = 0. The next step was to create a pipeline to enable us to train a model and make predictions with ease.



```python
from pyspark.ml.feature import Tokenizer, StopWordsRemover, HashingTF, IDF
from pyspark.ml.classification import LogisticRegression
from pyspark.ml import Pipeline

# Break text into tokens at non-word characters
tokenizer = Tokenizer(inputCol='Review_Text', outputCol='words')

# Remove stop words
remover = StopWordsRemover(inputCol=tokenizer.getOutputCol(), outputCol='terms')

# Apply the hashing trick and transform to TF-IDF
hasher = HashingTF(inputCol=remover.getOutputCol(), outputCol="hash")
idf = IDF(inputCol=hasher.getOutputCol(), outputCol="features")

# Create a logistic regression object and add everything to a pipeline
logistic = LogisticRegression()
pipeline = Pipeline(stages=[tokenizer, remover, hasher, idf, logistic])
```

Logistic Regression pipeline



```python
from pyspark.ml.feature import Tokenizer, StopWordsRemover, HashingTF, IDF
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml import Pipeline

# Break text into tokens at non-word characters
tokenizer1 = Tokenizer(inputCol='Review_Text', outputCol='words')

# Remove stop words
remover1 = StopWordsRemover(inputCol=tokenizer1.getOutputCol(), outputCol='terms

# Apply the hashing trick and transform to TF-IDF
hasher1 = HashingTF(inputCol=remover1.getOutputCol(), outputCol="hash")
idf1 = IDF(inputCol=hasher1.getOutputCol(), outputCol="features")

# Create a decision tree object and add everything to a pipeline
tree = DecisionTreeClassifier()
pipeline2 = Pipeline(stages=[tokenizer1, remover1, hasher1, idf1, tree])
```

Decision Tree pipeline

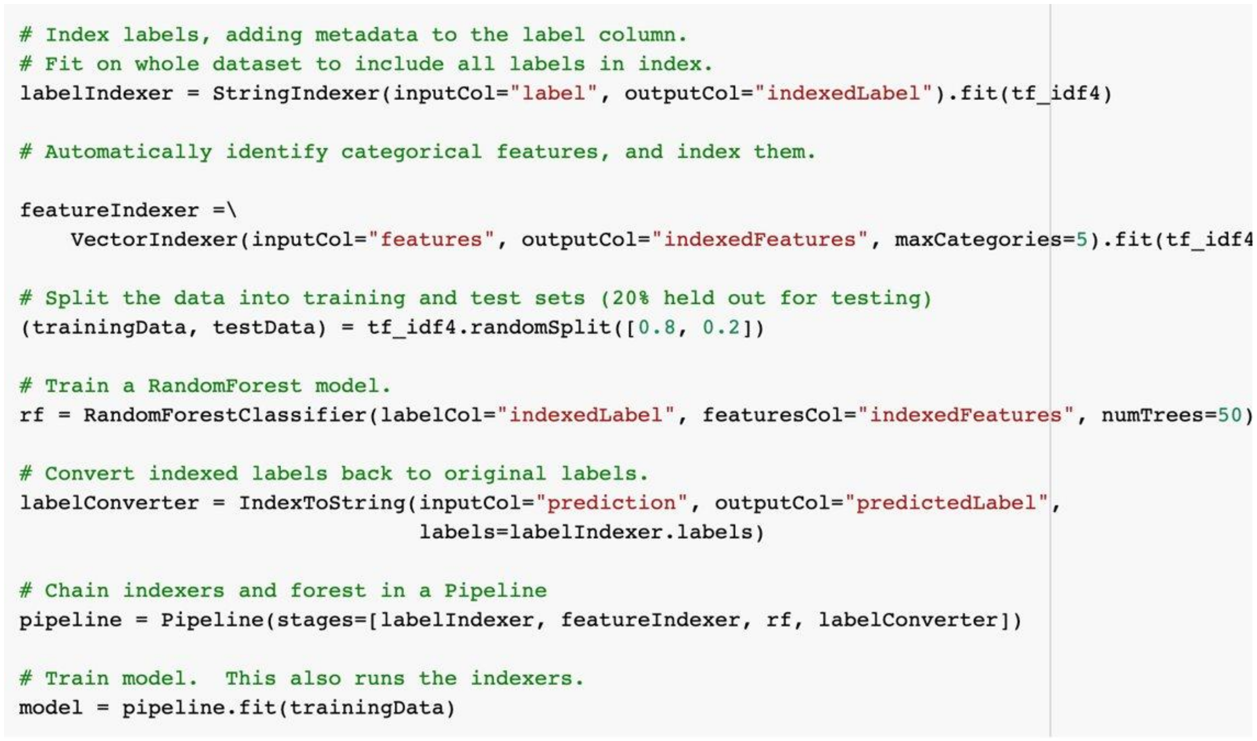
Results:

	Logistic Regression	**Decision Tree**
Implementation Time	< 1 minute	> 20 minutes
Precision	0.94	0.92
Accuracy	0.80	0.91

Note: Removing punctuations and numbers improved Logistic Regression's accuracy to 92%

Multi-Label Classification

The goal for multi-label classification was to predict the exact Rating of a visitor's Review There were 5 possibilities: 1,2,3,4 or 5. Similar to the pipelines created for Logistic Regression and Decision Tree pipelines, we created a pipeline for each of the models used for multi-label classification. We implemented Logistic Regression, Decision Tree and Random Forest models. The best-performing one was the Logistic Regression model.



```
# Index labels, adding metadata to the label column.
# Fit on whole dataset to include all labels in index.
labelIndexer = StringIndexer(inputCol="label", outputCol="indexedLabel").fit(tf_idf4)

# Automatically identify categorical features, and index them.

featureIndexer =\
    VectorIndexer(inputCol="features", outputCol="indexedFeatures", maxCategories=5).fit(tf_idf4

# Split the data into training and test sets (20% held out for testing)
(trainingData, testData) = tf_idf4.randomSplit([0.8, 0.2])

# Train a RandomForest model.
rf = RandomForestClassifier(labelCol="indexedLabel", featuresCol="indexedFeatures", numTrees=50)

# Convert indexed labels back to original labels.
labelConverter = IndexToString(inputCol="prediction", outputCol="predictedLabel",
                               labels=labelIndexer.labels)

# Chain indexers and forest in a Pipeline
pipeline = Pipeline(stages=[labelIndexer, featureIndexer, rf, labelConverter])

# Train model.  This also runs the indexers.
model = pipeline.fit(trainingData)
```

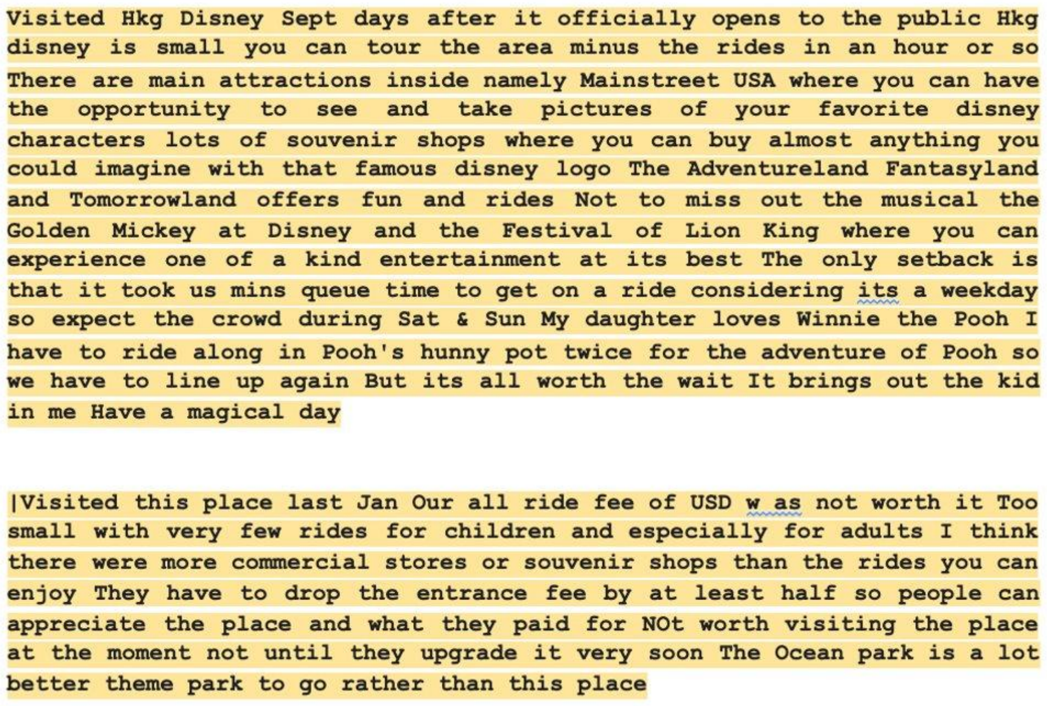
Random Forest Pipeline

Results

Model	Accuracy
Decision Tree	54%
Random Forest	55%
Logistic Regression	60%

Exploring wrongly-classified reviews

We went a step further to investigate possible reasons why our model would wrongly classify a review.



```
Visited Hkg Disney Sept days after it officially opens to the public Hkg
disney is small you can tour the area minus the rides in an hour or so
There are main attractions inside namely Mainstreet USA where you can have
the opportunity to see and take pictures of your favorite disney
characters lots of souvenir shops where you can buy almost anything you
could imagine with that famous disney logo The Adventureland Fantasyland
and Tomorrowland offers fun and rides Not to miss out the musical the
Golden Mickey at Disney and the Festival of Lion King where you can
experience one of a kind entertainment at its best The only setback is
that it took us mins queue time to get on a ride considering its a weekday
so expect the crowd during Sat & Sun My daughter loves Winnie the Pooh I
have to ride along in Pooh's hunny pot twice for the adventure of Pooh so
we have to line up again But its all worth the wait It brings out the kid
in me Have a magical day

|Visited this place last Jan Our all ride fee of USD w as not worth it Too
small with very few rides for children and especially for adults I think
there were more commercial stores or souvenir shops than the rides you can
enjoy They have to drop the entrance fee by at least half so people can
appreciate the place and what they paid for NOt worth visiting the place
at the moment not until they upgrade it very soon The Ocean park is a lot
better theme park to go rather than this place
```

Exploring wrong reviews from our dataset

Preliminary analysis showed the following:

- The model has a harder time making predictions on reviews with longer text.
- Some eviews expressed mixed feelings, with a rating that could be either good, bad or somewhere inbetween . This generally made it hard for a model relying on tokenized text as its input variable/feature to make predictions.

Conclusion

In summary, we learned that Sentiment Analysis is a great way to measure how customers feel about their experiences with a product/service. Using NLP and tools like PySpark, businesses can capitalize on text data (Reviews) to ultimately improve on their product/service offerings.

References

- PySpark documentation¶. (n.d.). Retrieved April 27, 2021, from https://spark.apache.org/docs/latest/api/python/index.html

- Classification and regression. (n.d.). Retrieved April 27, 2021, from https://spark.apache.org/docs/latest/ml-classification-regression.html

- Ml pipelines. (n.d.). Retrieved April 27, 2021, from https://spark.apache.org/docs/latest/ml-pipeline.html