# CSE 158 Assignment 2
# Zachary Perry PID: A16963488

**Topic:** Predicting Disneyland Ratings From Textual Visitor Reviews
**Source:** Kaggle dataset which can be downloaded at this link
**Exploratory Analysis:**

- There are 42,656 total reviews
- Review dates range from March-2010 to May-2019
- Mean rating across all reviews is 4.2177 (out of 5)
- Mean rating by Disneyland branch: There is certainly a noteworthy difference in mean values between Disneyland California and Paris.
    - California: 4.405
    - Hong Kong: 4.204
    - Paris: 3.960
- United States, United Kingdom, and Australia citizens make up about ⅔ of the reviews. I also find it interesting that even though Hong Kong is the location of one of the parks, only 554 (about 1%) of the reviewers are from there.

**Predictive Task:** Luckily for me, this dataset came with both textual reviews and numerical ratings (from 1 to 5) for each review. So I plan to create a model that attempts to predict a visitor's rating from their textual review. I am planning to attempt a couple different strategies learned in class, namely Bag-of-Words(BoW) and Term Frequency-Inverse Document Frequency(TF-IDF). As a quick summary, Bag-of-Words:

- Represents the text as a matrix where word occurrences are counted
- All words are treated equally, neither importance nor rarity are considered
- Usually less accurate but quicker than TF-IDF

Term Frequency-Inverse Document Frequency:

- Weighs words by their frequencies in each document and rarity across full corpus of text
- Highlights words that are more important/have more meaning to the tone of the piece of text
- Usually more accurate but slower than BoW

## Exploratory Data Analysis

```
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Review_ID          42656 non-null  int64
 1   Rating             42656 non-null  int64
 2   Year_Month         40043 non-null  datetime64[ns]
 3   Reviewer_Location  42656 non-null  object
 4   Review_Text        42656 non-null  object
 5   Branch             42656 non-null  object
dtypes: datetime64[ns](1), int64(2), object(3)
memory usage: 2.0+ MB
```

```
Rows in the dataset: 42656
```

```
Mean rating across whole dataset: 4.217695048762191
```

| Branch | mean | size |
|---|---|---|
| Disneyland_California | 4.405339 | 19406 |
| Disneyland_HongKong | 4.204158 | 9620 |
| Disneyland_Paris | 3.960088 | 13630 |

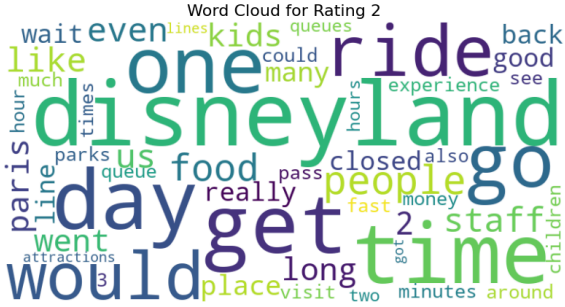
```
Dataset date range:
2010-03-01 00:00:00 2019-05-01 00:00:00
```

```
Top 10 countries with most reviews:
Reviewer_Location
United States     14551
United Kingdom     9751
Australia          4679
Canada             2235
India              1511
Philippines        1070
Singapore          1037
New Zealand         756
Malaysia            588
Hong Kong           554
dtype: int64
```

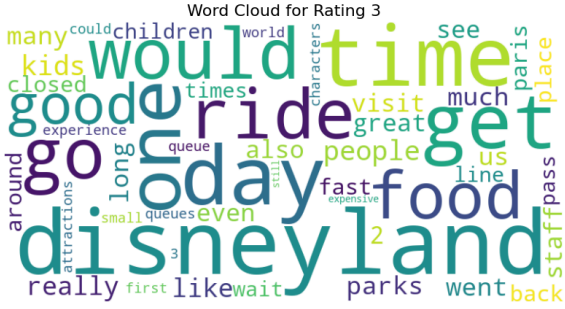
Word clouds of the most common words for each rating using python WordCloud library:

Word Cloud for Rating 1


Word Cloud for Rating 2




Word Cloud for Rating 3




Word Cloud for Rating 4


Word Cloud for Rating 5


**Method of Evaluation:**

I will be using Mean Square Error(MSE) as the main method of evaluation. MSE is a simple and effective method which penalizes predictions more severely the larger the mistake is. To avoid minority ratings from being drowned out, I will also take the MSE of each individual rating (1 to 5) to see how my model is performing for each level. I am expecting the ratings with the fewest number of reviews and the ratings furthest from the mean to have less accurate predictions just by the nature of these kinds of models.

**Baseline:**

To set a baseline to evaluate my model, I will use a trivial function that simply predicts the mean global rating for every review. This way, we can ensure our future models perform better and gauge their results in comparison. The overall MSE of the baseline predictions compared to actual ratings is 1.13, and obviously the ratings get less and less accurate the farther away from the global mean we are. So the 1 and 2 star ratings have a very poor MSE.

```
Baseline Test (Predict the mean value only)...
```

```
Overall Baseline Mean Squared Error : 1.1307315214083435
```

```
Rating 1: MSE = 10.353561426828717, Count = 1499
Rating 2: MSE = 4.918171329304335, Count = 2127
Rating 3: MSE = 1.4827812317799542, Count = 5109
Rating 4: MSE = 0.047391134255572585, Count = 10775
Rating 5: MSE = 0.6120010367311914, Count = 23146
```

**First Model - Bag-of-Words:**

The first model I want to try is the BoW model because of its simplicity. I split the dataset 80/20, with 80% of the data being used for training the model, and 20% being used for testing. I then created a CountVectorizer with 1000 max features and fit the LinearRegression model. I experimented with a couple other max_features values but 1000 seemed to give

the best accuracy. Here are the results of this first model:

```
Trying Bag-of-Words...
```

```
Mean Squared Error: 0.6887657494643821
```

```
Rating 1: MSE = 4.716602669321153, Count = 292
Rating 2: MSE = 2.174064783322758, Count = 428
Rating 3: MSE = 0.9528732869968892, Count = 1035
Rating 4: MSE = 0.32053511173969024, Count = 2200
Rating 5: MSE = 0.41018146599455774, Count = 4577
```

The overall MSE here as well as for the individual ratings is quite a bit more accurate than the baseline model. Though the 1 and 2 star reviews are still inaccurate. Next I would like to try removing stopwords from the model and see if that improves our accuracy on the test set.

```
Trying BoW with no stopwords...
```

```
Overall Mean Squared Error: 0.7034679963005178
Rating 1: MSE = 4.890715762192961, Count = 292
Rating 2: MSE = 2.2415404734559305, Count = 428
Rating 3: MSE = 0.9790210992120268, Count = 1035
Rating 4: MSE = 0.32196816451362925, Count = 2200
Rating 5: MSE = 0.41356867372123723, Count = 4577
```

Interestingly, removing stopwords actually decreased our overall accuracy on the test set. This implies that the number of stopwords in the textual review may be correlated in some way with the numerical rating. So we will keep the stopwords in the model for now. The biggest issue so far is that ratings 1 and 2 are performing pretty poorly, so let us try and improve on these ratings with another model.

**Second Model - TF-IDF:**

Using the same 80/20 dataset split, I hope to increase our performance on the lower ratings with a TF-IDF model. Since TF-IDF gives words more weight based on their rarity as well as their popularity, I am hoping this model can pick up better on any of the less common negative tone words that could indicate a potential bad rating. I used a TfidfVectorizer for this and fit the LinearRegression model with these results:

```
Trying TF-IDF...
```

```
Mean Squared Error: 0.5688591456042207
```

```
Rating 1: MSE = 3.149134132392354, Count = 292
Rating 2: MSE = 1.3058111358480293, Count = 428
Rating 3: MSE = 0.742838404516198, Count = 1035
Rating 4: MSE = 0.3693206815820283, Count = 2200
Rating 5: MSE = 0.3919005132923224, Count = 4577
```

After playing around with the max_features value and training the model several times, I found that using around 3000 features for this model gave the best results. As you can see, the TF-IDF model is significantly more accurate overall and in most of the rating categories except for Rating 4. Just to attempt to increase our performance a little more, I want to test this same model without any stopwords. Though I am prepared for a similar result to before. Here are the results:

```
Trying TF-IDF without stopwords...
```

```
Mean Squared Error: 0.5801349534000354
```

```
Rating 1: MSE = 3.2721636484476746, Count = 292
Rating 2: MSE = 1.3774626851036482, Count = 428
Rating 3: MSE = 0.7680493535031906, Count = 1035
Rating 4: MSE = 0.36680125513574324, Count = 2200
Rating 5: MSE = 0.3938806566885693, Count = 4577
```

Similar to before in the BoW model, the accuracy slightly decreased when I attempted removing stopwords. Therefore, the best model I was able to find was a TF-IDF model keeping stopwords and using 3000 max features. This gave us a Mean Square Error of 0.580 overall which represents an average deviation of 0.76 stars from the actual rating. Compared to the baseline model which had an overall MSE of 1.131, which represents an average deviation of 1.06 from the actual rating, the improvement is quite significant. I think the main difference maker is the fact that TF-IDF gives a greater weight to rarer words. In this dataset in

particular, most of the ratings are 4 or 5 stars, so the words that appear in the lower rating reviews (negatively toned words) were given more emphasis in this model compared to BoW. Improving any more past this point is tricky, because the more weight you begin to put on the reviews with lower ratings in training, the worse the accuracy becomes on the large number of 4 and 5 star reviews. Since there are so many of these higher rated reviews, any small deviation in the performance for them can drastically worsen the overall MSE. With this in mind, I would like to look at some Natural Language Processing (NLP) research and see if I can find any good papers related to my situation.

**Related Research/Literature:**

Attempting to predict numerical ratings from textual reviews has been a focus in NLP because of its wide application to shopping, social media, and customer feedback analysis. Here is a link to an article in which researchers were presented with a similar problem: predicting numerical star ratings from short-text reviews of video games. It is important to note that instead of typical continuous regression models like linear regression which I used, they only used classification models. They tried a variety of different models, as well as some "state-of-the-art" models as shown in this chart:

**Table 2.** Different models for predicting the rating of review texts.

| Model | Accuracy |
|---|---|
| Naïve Bayes | 48% |
| SVM | 50% |
| Logistic Regression | 70% |
| BERT | 23% |
| TextCNN | 56% |
| Word2Vec | 73% |
| Doc2Vec | 79% |

According to the chart, Word2Vec and Doc2Vec, which are neural network approaches, performed the best on their dataset. Another thing I noticed was that BERT (Bidirectional Encoder Representations from Transformers), which typically performs quite well, actually performed the worst in this study just barely being more accurate than a random guesser. Here is a link to another article which attempts to find the best model to predict vegan/vegetarian restaurant ratings from textual reviews. In summary, the researchers find good results with Random Forest and Support Vector Machines. They also explore neural networks which they say show more potential for more nuanced tasks like sentiment analysis. They found the best success with a model that integrated both text-based features with structured data like the demographics of the reviewers.

**Table 4.** Results of sentiment prediction on an original dataset for models with TF-IDF.

| ML Model | Accuracy | F1 Score | | Precision | | Recall | |
|---|---|---|---|---|---|---|---|
| | | Positive Class | Negative Class | Positive Class | Negative Class | Positive Class | Negative Class |
| Neural Network | 0.95 | 0.98 | 0.30 | 0.96 | 0.61 | 0.99 | 0.20 |
| Random Forest | 0.95 | 0.98 | 0.04 | 0.95 | 1.00 | 1.00 | 0.02 |
| SVM | 0.96 | 0.98 | 0.46 | 0.97 | 0.67 | 0.99 | 0.36 |
| NB | 0.95 | 0.97 | 0.04 | 0.95 | 0.88 | 1.00 | 0.02 |
| BERT * | 0.96 | 0.98 | 0.55 | 0.97 | 0.61 | 0.98 | 0.50 |

Interestingly with this dataset is that BERT actually performed pretty well in contrast to the other article.

**Takeaways and Conclusion:**

Overall I think it was fun to look at some professional work with the subject matter. Simple models like Naive Bayes, SVM, and BoW are generally pretty lightweight and accurate, though there is certainly room for improvement. Models like BERT, Word2Vec, and Doc2Vec typically perform better, though not in all scenarios. Of course with that increased performance also comes increased time and space complexity when training and running the models. And to get the best possible accuracy for a specific task, utilizing as much information as possible, including demographic features if available, all factored into the model tend to give the best results. It is fair to conclude that there is not one certain model that works

best for every scenario. There are tradeoffs to each model: some overfit to the training data, some underfit, some are extremely accurate but also take forever to run. In practice it is best to try a variety of models and features and use as much data as you have available to customize your model for predicting your data.

For my dataset in particular, I probably could have incorporated the location of the reviewer and the location of the Disney park as features in the model. However these would increase the complexity of the model and open up the possibility of overfitting to the seen data. I think a simple TF-IDF Linear Regression model performs pretty well given the limited amount of features I had in the dataset. As stated before, the biggest issue I ran into when trying different models was that my dataset was quite skewed in that most of the reviews were 4 and 5 stars. If I were to do this again and attempt to optimize the accuracy as much as possible, I might try a classification model that uses a sentiment analysis calculation to guide the prediction rather than BoW and TF-IDF. But the simplicity and quicker runtime of these models are certainly a positive to using them.