Using Statistical Methods to Build the Optimal Disney Day: An Analysis of Ride Wait Times at Walt Disney World

Kelly Petrino
Adviser: Professor David Dobkin

Abstract

This paper outlines the exploration, development, and evaluation of a machine learning program that uses previous wait time data on a subset of rides at the Walt Disney World Resort to predict the time-optimal path through the park on a given day based on the conditions - weather, season, etc. The program allows a user to input a set of conditions and a specific park within the resort and predicts the shortest path based on past data. In addition to this prediction functionality, I offer an analysis of trends within the data by evaluating the performance of the program. This analysis aims to uncover the underlying causes of long wait times while also helping to create a more magical experience for potential users by optimizing their time in the parks.

1. Introduction

1.1. Motivation

In looking for a publicly available data set, I wanted to find something that I was truly interested in. Growing up, I was an avid fan of all things Disney and visited the Walt Disney World (WDW) resort in Orlando, FL a number of times. So, I was admittedly very excited when I stumbled upon a fleshed out data set of attraction wait times along with accompanying metadata. While my family and I may think that we have the inner workings of the ideal daily schedule down pat after years of practice, I was sure that there was data to support trends that we never could have predicated on a given day. Perhaps for selfish reasons then was I so excited to take on this project and discover new things about how guests in the parks operate. Beyond that, however, I think that my findings could potentially help lots of families visiting Disney World plan their vacation to maximize the number of rides they can go on while also minimizing the amount of time on line.

If you've ever visited any of the Disney theme parks, then you've witnessed first hand the chaos that often ensues as guests criss-cross the pathways looking for their next activity. In the Magic Kingdom of WDW alone, there are six different themed lands with 41 available attractions to choose from. Thus, it is easy to see how planning could be overwhelming to a first-time visitor, especially when taking into consideration Fastpasses, which allow guests to cut the line at their assigned time (each guest is allowed to choose three per day). As technology has advanced in the parks, it has gotten easier to assess options as Disney constantly updates the official wait times for the attractions on the Walt Disney World App. However, it can still be confusing figuring out where to head next as crowds swarm to the lowest posted times, lengthening the lines.

Needless to say, there is a huge demand for resources to help plan a day in the parks at WDW. After a simple Google search of "Disney trip planning," I came across seven different blogs dedicated to providing tips and tricks to planning your Disney vacation with varying levels of personalization for the user - the DisneyTouristBlog offers dozens of articles outlining the must do's of the park, MouseSavers provides links to additional information for guests on a budget, TouringPlans gives the option to purchase a custom day plan. Some form of "unoffical guide" available for purchase is also a staple of each of the services, with new editions coming out yearly. Disney itself has even stepped up to compete with these long-revered guides, offering guests the ability to plan out their days up to 60 days in advance of arriving by booking Fastpasses and making dining reservations. For many, this advance planning provides a feeling of security, ensuring that they and their families will make the most of their time and experience everything they're excited about. With 58 million visitors in a typical year, a sizeable portion will take advantage of one or more of these resources - Fastpasses for new or popular attractions will run out of availability within hours of being released and travel experts continue to release new guides year after year, a sign that they must be profitable. TouringPlans.com even boasts 140,000 paying customers, with millions more visiting the site over the span of 9 years according to founder Len Testa [7].

While so many resources already exist online to help WDW guests plan their trips by giving tips and listing things to do, only one of the resources I came across used actual statistical methods to

make their suggestions, and they all noticeably fall short of providing a clear, succinct interpretation of what causes the wait times to fluctuate. There are so many different factors that could causes shifts in the crowd flow, ranging from weather patterns to special events to school holidays and more. While it is easy to think about this metadata and makes unbased assumptions about which factors attract or deter guests, there is not always proof to support them. Uncovering these underlying causes of crowd behavior could be applied beyond the scope of WDW and would likely be useful in planning trips to other amusement parks, such as Universal Studios or Six Flags, as well.

1.2. Goal

Thus, there are some noticeable gaps in the resources desired by potential WDW guests who want to optimize their time in the parks. My goal is to use machine learning and previous ride wait time data in order to build a predictive program that can take in the conditions of any given day and output the optimal ride path for that day in a single park. From evaluating this predictive program, I hope to gain a better understanding of the forces that drive Disney guests to visit each park and make the decisions that they do, whether it be the weather, season, or special events. Not only will this information be interesting for me on a personal level, I think it will provide valuable insight into amusement park consumption in general and allow for any reader to take the findings into consideration when planning their next trip (post COVID, of course).

1.3. Overview

As stated above, it is clear that there is an existing demand for resources to help plan Disney vacations. Thus, I aim to build a predictive program to output the optimal path through a park on a given day using previous wait time data. I will also analyze the results of the program to uncover the unseen driving forces of long wait times in WDW.

In this paper, I will first detail the existing resources already available to Disney guests and how their work influenced my own design and implementation decisions, helping to shape my problem statement and inform the questions I should be asking the data. I will then outline my approach to the project, highlighting the novel components of my design plan and defending my design choices.

Next, I will go into the details of my implementation of the project including python packages used and the inner workings of my algorithms, including sample outputs to help visualize the results. My implementation section will focus on the cleaning of the data, the predictive algorithm, and the accuracy score definition. After, I will explain my evaluation plan and the scripts written to compute accuracies on the validation set data. It will also discuss the trends uncovered during the evaluation stage. Lastly, I will give my final remarks on the project regarding its design, implementation, performance, and limitations. I also road map potential future work to expand on the project and help it continue to reach its goals of predicting optimal paths through the parks while unveiling the causes of wait times.

2. Problem Background

2.1. Related Work

The inspiration for this project and most notable example of my end goal product is the WDW Personalized Touring Plans offered by TouringPlans.com [5]. These custom plans are a part of a paid service to the website. They take into account a number of different user preferences when computing the optimal plan, making sure to prioritize the user's desired experiences. Figure 1 depicts the steps that users take to purchase and utilize one of these custom plans. As seen in the steps, there is a lot of freedom of choice for the users as the optimal day looks different for everyone. TouringPlans.com has the luxury of available data for Disney elements beyond a limited subset of rides, making their predictions and planning guides much more detailed and adaptive to the user than the prediction program I aim to write. The plan you pay for is all encompassing, designating times to ride attractions, use Fastpasses, get a snack, or watch a show. In addition to the original outline, the plan is constantly updated in case of ride down times or changes to your schedule. This feature to provide custom tour plans in WDW also exists for Disneyland in CA and Universal Orlando. These additional personalized touring plans are also offered by TouringPlans.com and include the same steps as listed in Figure 1. The algorithm and development of the program will be detailed in 2.2. Despite every effort to do so, I could not find another tour planning service from

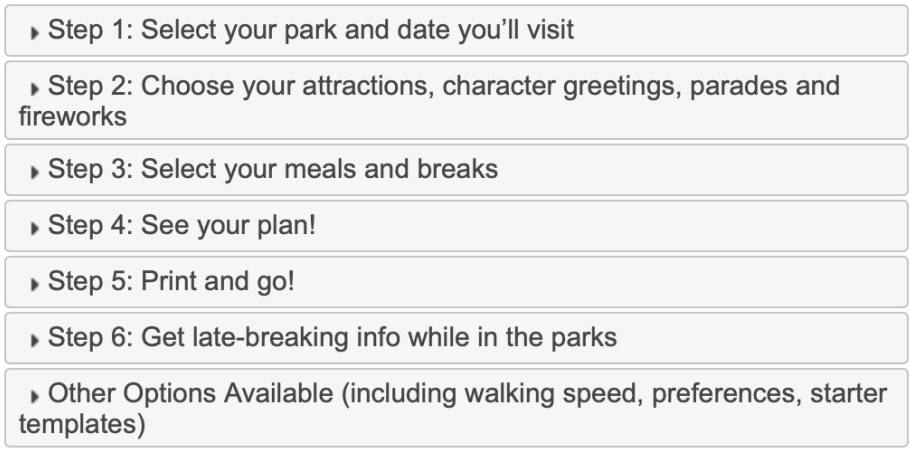

Figure 1: The entry fields for a personalized touring plan at WDW.

any other amusement park aficionados, for WDW or any other locations. TouringPlans's use of statistical methods to help plan the optimal day is unique and hard to develop, so, mimicking its function with a smaller data set and less complex methodology as I aim to do is an exciting and novel concept.

The personalized touring plans can be accessed from the Lines App as seen in Figure 2, also offered by TouringPlans.com. The Lines App is also a useful tool for park goers, offering real time wait data, access to their step-by-step touring plans, restaurant menus, and crowd level rates [6]. Users are able and encouraged to input their own "real" wait time data by using a stopwatch in the app to report their time on the line. TouringPlans.com utilizes this input data to improve their own ML models for predicting wait times. With 4.7 stars and 1,822 Ratings in the Apple App Store, it seems the technology is well appreciated by its users and provides accurate predictions that help guests at WDW save time, making a more optimal experience. In the case of my own predicition program, I opted to discard the user reported "actual" wait times provided in the data set in order to simply program and avoid unreasonable skews in predictions since my algorithm is largely based on averaging values.

While my plans for a prediction algorithm are ambitious, there certainly exist simpler projects to explore the wait time data available from TouringPlans.com. An example would be the work of Steven Chen, who completed a statistical analysis of the data while focusing on one ride, Expedition Everest [1]. By only using the values from one ride, Chen was able to plot values on graphs, looking

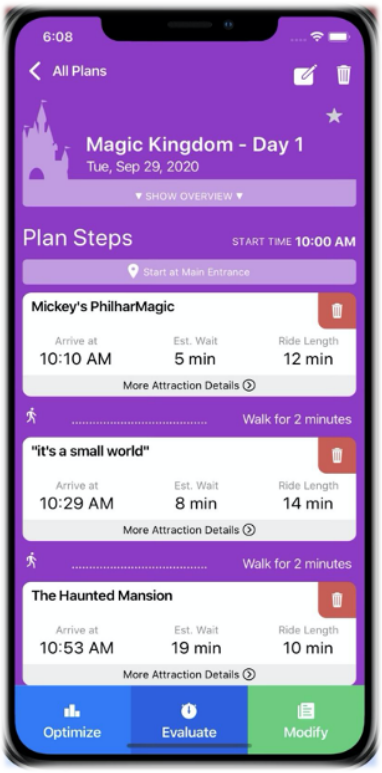

Figure 2: Example tour plan featured on the Lines App.

for a more visual interpretation of the data as compared to my more quantitative approach rooted in accuracy rates. For the purposes of this data set, I thought the conclusions that could be drawn from numbers and predictions would be more compelling and telling of the underlying causes of wait times. However, Chen still served as a guide for possible routes of analysis with the data. His most substantial conclusion as it pertains to my work was finding that K-Nearest Neighboor clustering (as compared to a linear regression model, regression tree, or random forest) has the most positive impact on "predictive power," improving the RMSE by 1.2 minutes. This finding helped me choose K-NN as my method of clustering.

2.2. Academic Background

In searching for clues on how to look for underlying causes of wait times in the data, I started reading some of the published guides for planning Walt Disney World Trips. The most helpful was *The Unofficial Guide to Walt Disney World 2020* by Bob Sehlinger and Len Testa which offered up the five keys to avoiding long lines at WDW [2]:

1. Decide in advance what you really want to see.
2. Arrive early.
3. Know what to expect when you arrive.
4. Use a touring plan.
5. Use Disney's FastPass+ ride-reservation system.
6. Use the single-rider line if one is available.

These tips stress early planning and anticipating crowd levels. Hailing arriving early as the number one key to optimizing your day, this guide also provides in depth explanations and rating of each of the park's attractions and dining options. These guidelines helped shift me away from my original hunch that weather would play a large role in how lines form when in reality, adverse weather does little to deter the Florida crowds. It instead lead me down a path more focused on anticipating crowd levels, something more directly tied to the holiday seasons or a new ride opening up.

The data set being used in my project was released on TouringPlans.com with the explicit intention of allowing data science students to play with it and share their findings with the original publishers [4]. The data was collected by the popular touring blog which offered a phone app, the Lines app, with live ride wait time postings even before Disney developed their own app to do so. They have been collecting wait time data over the past 10 years, using it to build their own ML programs to predict crowds in the parks on each day of the week. They offer paid services to map out ideal days in the park, much like my project sets out to do but at a much larger scale.

The collection of this data had its inception as blogger Len Testa's master thesis. Thinking of the project while standing on a painfully long line for Rock 'n' Rollercoaster, Testa thought that "there has to be a better way to do this" [7] and started working on the foundations of the Tour Plan algorithm of TouringPlans.com. Testa takes a complex approach to stitching together the optimal day for users, thinking of it like a travelling salesman problem. His approach is not unlike that of a decision tree [3], following down the path that gives the most utility at each step (see Figure 3). While this approach is surely more efficient in terms of taking into account user desires and factors outside saving time, my project was limited in the availability of user input and choice as I only

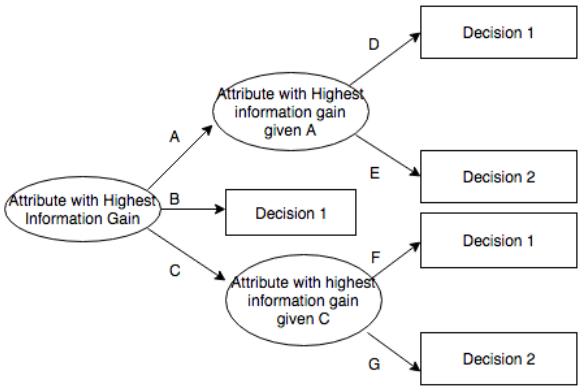

Figure 3: Example decision tree that chains to form tour plan.

had 14 attractions to work with and no data about restaurants or entertainment. Thus, I opted to optimize only for time rather than utility as Testa's algorithm does. Of course, given the timeline of this project as compared to the years of research invested into TouringPlans.com, the functionality of my predictive software will be much more limited. However, I hope to mimic the program's basic function and use that information to make inferences about the underlying causes of wait times in WDW and amusement parks in general.

2.3. Problem Statement

The excess of planning materials rendered in one Google search indicate that there is an overwhelming demand for products to help future WDW guests plan their trips in order to fit their needs. From blogs entries about what to do with small children to rankings of the biggest thrills, there are numerous different ways to spend the day in the parks. While there already exists a prediction algorithm that optimizes for user utility in the TouringPlans Personalized Tour Planner and numerous blogs with helpful tips and tricks for vacation planning based on your needs, my project aims to create a prediction algorithm that optimizes solely for time. My prediction algorithm will use K-NN to cluster the data and take averages across clusters to make predictions.

In doing so, I hope to uncover the underlying causes of long and volatile wait times. Though other work brushes over the concept of exterior forces like holiday seasons and new big ticket attraction causing upticks in wait times, they fail to clearly state so, instead focusing on timing in the parks as

the key to optimizing the day. I certainly believe timing in Disney is important (and my results later on will surely show as much); however, I hope that in discovering more general sources I will be able to apply my findings to other themes parks as well such as Universal Studios or Six Flags.

3. Approach

3.1. Key Novel Idea - A Time-Optimized Prediction Script

Currently, there does not exist a free, accessible way to plan the time optimal day at WDW for the average park-goer, limiting the magical quality of the Disney experience that so many crave. So, as stated before, the key novel idea of my project is the design and implementation of a prediction script that optimizes for time. There are a number of components that contribute to this idea. The base of the prediction script is choosing the metadata variables with which to cluster the days. I will explore many different combinations of variables including weather based, seasonal based, and extra magic hour based. Through experimentation, I will find the combination that I feel best fits the data set and cluster the training data accordingly using K-NN.

To make a future prediction, my program will input the relevant metadata and chosen park for a hypothetical day and use K-NN to fit the day into a cluster. From there, I will take averages across the existing data for that park in the cluster to build a complete data table of predicted wait times for the day. Using this table, I can compute the shortest possible path through the park by calculating all possible routes and returning the one with the minimum total wait time. In my design the prediction algorithm should return the predicted path of rides, the predicted start time to begin waiting for the first ride, and the predicted total wait time. I will also develop a similar retroactive prediction script that takes in a specific day and park and output the actual optimal ride order, start time, and wait times for that day.

Another key component of the prediction algorithm is defining what accuracy is. I plan to evaluate the accuracy of the program by splitting the data set into a train and validation set and use the validation set as hypothetical days to predict and then compare to. Given the outputs of the prediction script, it is impossible to use a strict 1 to 1 equals comparison in order to define an

accurate prediction. I want to make sure that the penalties for flipping ride orders or starting at a different time fit the crimes. Thus, I have to design some accuracy function that takes into account the severity of the prediction blunder when determining whether or not the prediction should be deemed "accurate."

In addition to predicting the path through the park with the smallest wait time, I will analyze the results, taking into consideration external factors such as new attraction openings and holiday events to explain discrepancies in predictions. In searching online, I found a shockingly limited amount of discourse surrounding these types of factors and their influence in the parks with people instead stressing the important of getting to the parks early as the best way to optimize the day. This analysis should uncover the underlying causes of sporadic wait times in WDW and will hopefully be transferable to other amusement parks with similar crowd appeal. In doing so, I will expand the scope of my research beyond the limitations of my restricted data set and help potential readers gain insight into how they should be making decisions while visiting a theme park.

3.2. Why It Works

I am confident that the prediction algorithm outlined above will be at least moderately accurate (i.e. it will produce results with an accuracy above 60%). The design of the algorithm is nearly fool-proof, making the trade-off of speed for accuracy by computing every possible path. Though this trade-off means many hours of waiting and computing, I think it will be the right decision as even minor deviations in the predicted ride order could potentially result in unreasonable increases to the total wait time. Crowds can be erratic in their behavior, and they have tendencies to flock towards shorter wait times, causing big bursts in lines. Without a doubt, the algorithm will output the shortest time path through the park given the input data table as the problem statement demands. However, it's certainly possible that there could be some mishaps as the computed prediction data table has many missing values upon merging and will have to be extensively cleaned in order to fill down times. Thus, I restrain from predicting too high an accuracy as I am less confident in my data filling methods and also anticipate that there will be unforeseen factors that will influence the actual

wait times that are unaccounted for in the predictions.

I am more confident in the choice of K-NN as the clustering method for the data. From my past experiences and readings about clustering, I feel that K-NN is the perfect fit for this data as I want to take into account a lot of different metadata elements in the grouping of the data. In terms of the potential fields to be used, elements like holiday season and weather are often correlated, making it easy for the K-NN algorithm to recognize boundaries and help make the prediction data more accurate and thus create outputs that are closer to the actual shortest time path.

In terms of the analysis following the evaluation on the validation set, I am hopeful that since I am designing accuracy to reflect penalty according to crime that my final total accuracy percentages will reflect the effects of not slight mishaps on the part of the rendered prediction data but rather larger underlying causes that were not taken into consideration in the clustering algorithm. The accuracy function should deem minor mistakes in the predicted ride path order or start time as forgiveable, marking the prediction as accurate despite not being directly on the money. This will also allow me to focus my analysis on the unspoken factors that control wait times in WDW and other amusement parks across the country.

4. Implementation

4.1. Problem Statement

The main question I want to answer in completing this project is: What are the main underlying factors that influence wait times in WDW parks?

My approach to the project clearly outlines how this question is being answered. In order to isolate these factors, I designed a prediction algorithm that will predict the shortest path through a park on a given day. I will then evaluate that algorithm using an accuracy metric that gives leniency for small mistakes to account for misgivings in the filling of data. Thus, I will be able to attribute inaccuracies in the algorithm to these underlying factors and identify them by analyzing the data. The goal is for these factors to be generalizeable to other amusement parks in order to expand the scope of the research.

4.2. Project Overview

The initial spark for this project stemmed from the discovery of the published and available wait time data sets for 14 different attractions in WDW. The breakdown of attractions is as follows:

- **Magic Kingdom:** 7 Dwarfs Mine Train, Pirates of the Caribbean, Splash Mountain
- **EPCOT:** Soarin', Spaceship Earth
- **Hollywood Studios:** Alien Swirling Saucers (2018), Rock 'n' Rollercoaster, Slinky Dog Dash (2018), Toy Story Mania!
- **Animal Kingdom:** Avatar Flight of Passage (2017), DINOSAUR, Expedition Everest, Kilimanjaro Safaris, Na'vi River Journey (2017)

Each ride was contained within its own .csv file with official wait times being documented every 5-9 minutes from January 1, 2015 until December 31, 2019 (with the exception of newly opened rides, indicated above), culminating in around 20K entries for each ride. The exact number of entries varies ride to ride as post times often differ. Figure 4 and 5 depict the inital state of the data for the 7 Dwarfs Mine Train. The data sets can be accessed at https://touringplans.com/walt-disney-world/crowd-calendar#DataSets.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250873 entries, 0 to 250872
Data columns (total 4 columns):
date        250873 non-null object
datetime    250873 non-null object
SACTMIN     4509 non-null float64
SPOSTMIN    246364 non-null float64
dtypes: float64(2), object(2)
memory usage: 7.7+ MB
```

Figure 4: Summary of values for 7_dwarfs_train.csv

	date	datetime	SACTMIN	SPOSTMIN
0	01/01/2015	2015-01-01 07:51:12	NaN	45.0
1	01/01/2015	2015-01-01 08:02:13	NaN	60.0
2	01/01/2015	2015-01-01 08:05:30	54.0	NaN
3	01/01/2015	2015-01-01 08:09:12	NaN	60.0
4	01/01/2015	2015-01-01 08:16:12	NaN	60.0

Figure 5: Sample values for 7_dwarfs_train.csv

While in pretty good shape, the existing data needs a lot of cleaning - merging values across

parks, filling in empty values, removing down times, removing user reported "SACTMIN" times. Once cleaned, my project aims to use this data to implement two main algorithms with the same fundamental backbone - a retroactive script that outputs the actual shortest ride path on a given day and the future predictive script that uses clustering data to estimate the shortest path on a given day. I will also define an accuracy function that uses the output from both these algorithms to determine whether or not a predict for a given date is correct or not. The evaluation and analysis of the script output can be found in Section 5.

The repository with the .py files used to develop this project can be found at: `https://github.com/kellypetrino/iw07`. A disclaimer that these files were originally written as python notebooks and cannot be run as a single cohesive file as function definitions and files loaded changed as I worked with different data over the course of the semester.

4.3. Implementation Details

In executing this project, I had to do a lot of independent research on how to develop an ML program in python using existing packages. I ultimately landed on using `scikit-learn` and `pandas` quite a bit. I spent the first couple of weeks trying to identify a data set to use and once I found it, getting familiar with the contents and trying to figure out a project by plotting values and looking for initial trends to spark questions. As detailed in Section 2, I did a lot of online research surrounding similar projects, and that is how I ultimately decided on developing a predictive program to output optimal paths through the park.

From the beginning, I wanted to do more than just report numbers and was inclined to look for trends in the data. While I originally intentioned to find a more visual way to display my findings throughout the semester, the trends ended up manifesting themselves in the results of the evaluation stage as I was able to attribute discrepancies across parks to unaccounted factors (more in Section 5).

The first major step I took towards implementing my project was cleaning the data which proved a much more arduous task than imagined. I defined a function `clean` to streamline the process as

well as a hand full of other helper functions to iteratively load in each .csv file and prepare it for merging. Cleaning each file consisted of standardizing the column headings, converting the DATE and DATETIME values from `str` values to `datetime` values, dropping the SACTMIN values, and removing down times when the ride wasn't operating from the entries. A snippet from `clean.py` is included below:

```
def clean(ride):
    ride.columns = ["DATE", "DATETIME", "SACTMIN", "SPOSTMIN"]
    ride["DATE"] = pd.to_datetime(ride["DATE"])
    ride["DATETIME"] = pd.to_datetime(ride["DATETIME"])
    new = dropActTimes(ride)
    new = dropDownTimes(new)
    return new
```

Once each individual ride was cleaned, I was able to create a separate DataFrame for each park consisting of all of the ride data for each of the attractions. After merging with `pandas.merge`, there were many gaps in the value due to differences in exact reporting times among the rides. Thus, I had to fill in the missing values. As defined in `fill()`, I followed the following algorithm to fill each individual cell:

1. If non null values in cells preceding and following for that ride, take the average of the two and fill in the gap. If filled, skip following steps.
2. If non null values at the cell's same time value from day before and after, take the average of the two and fill in the gap. If filled, skip following steps.
3. If non null values from year before at same date and time of empty cell, fill in the gap. If the ride is new, apply exponential decay ($60e^{-t}$ where t is years since ride open). If filled, skip following steps.
4. If non null values from year after at same date and time of empty cell, fill in the gap. If the ride is new, apply exponential growth ($60e^{-t}$ where t is years since ride open). If filled, skip following steps.

5. If covered edge case (first time of day, last time of day, first entry, last entry), fill gap.

The cleaning, merging, and filling of the data sets yielded the following (previews of) data tables for each of the parks that contain entries for each of the rides in the park at 5-9 minutes intervals.

	DATE	DATETIME	DWARFS	PIRATES	SPLASH
0	2015-01-01	2015-01-01 07:51:12	45.0	5.0	5.0
1	2015-01-01	2015-01-01 08:02:13	60.0	5.0	5.0
2	2015-01-01	2015-01-01 08:09:12	60.0	5.0	5.0
3	2015-01-01	2015-01-01 08:16:12	60.0	5.0	5.0
4	2015-01-01	2015-01-01 08:23:12	60.0	5.0	5.0

Figure 6: Head values of Magic Kingdom data.

	DATE	DATETIME	SOARIN	SPACE
0	2015-01-01	2015-01-01 07:45:15	10.0	5.0
1	2015-01-01	2015-01-01 07:52:16	10.0	5.0
2	2015-01-01	2015-01-01 08:03:17	10.0	5.0
3	2015-01-01	2015-01-01 08:10:16	35.0	5.0
4	2015-01-01	2015-01-01 08:17:19	45.0	5.0

Figure 7: Head values of EPCOT data.

	DATE	DATETIME	ALIEN	ROCKN	SLINKY	TSM
225239	2019-12-31	2019-12-31 23:30:02	10.0	35.0	30.000000	20.0
225240	2019-12-31	2019-12-31 23:37:02	10.0	50.0	27.927234	15.0
225241	2019-12-31	2019-12-31 23:44:02	10.0	40.0	27.927234	15.0
225242	2019-12-31	2019-12-31 23:51:02	10.0	40.0	27.927234	5.0
225243	2019-12-31	2019-12-31 23:58:02	10.0	15.0	22.927234	5.0

Figure 8: Tail values of Hollywood Studios data.

	DATE	DATETIME	FLIGHT	DINO	EVEREST	SAFARI	NAVI
224394	2019-12-31	2019-12-31 23:38:02	50.0	10.0	5.0	7.5	5.0
224395	2019-12-31	2019-12-31 23:45:02	45.0	5.0	5.0	7.5	5.0
224396	2019-12-31	2019-12-31 23:45:57	45.0	5.0	5.0	7.5	5.0
224397	2019-12-31	2019-12-31 23:52:01	45.0	5.0	5.0	7.5	5.0
224398	2019-12-31	2019-12-31 23:59:02	45.0	5.0	5.0	7.5	5.0

Figure 9: Tail values of Animal Kingdom data.

With the wait time data processed and ready, I was able to switch my attention to the meta data and start developing the clusters. In order to start the process for the clusters, I read in `metadata.csv` and split the data into a training and validation set. I would use the training set to form the clusters and the validation set to evaluate the accuracy on and tune the parameters. As explain in Section 2, I chose the K-NN clustering algorithm based on previous experiences and other's research with these data sets. Clustering was simple to implement by using `sklearn.cluster.KMeans` which takes a DataFrame of meta data values and computes n clusters using the K-NN algorithm. I did a lot of experimentation when it came to deciding on which metadata values to use and how many clusters to compute. In evaluating which meta values we're best, I did manual analyses of each of the options, looking at the wait time data for the days and comparing against other options to see which seemed like the most cohesive grouping. These options included using only available weather data, using only available holiday data, and including all of the possible meta data. The set of meta data I ultimately landed on was `['DAYOFWEEK','SEASON','MKHOURSEMH','EPHOURSEMH','HSHOURSEMH','AKHOURSEMH', 'WDWMINTEMP_mean','WEATHER_WDWPRECIP']` which includes the the day of the week, the season (of which there are 20, including Christmas, Summer Peak, and September Low), the hours of each park, the mean temperature, and the precipitation. Upon my own analysis, I felt that the SEASON variable was the best differentiator of crowd levels in the parks, something that strongly influences the wait times. I also landed on 35 clusters as any fewer and there is too much variety in the wait times of each cluster and any more, there are not enough values in each cluster for a good prediction. I then used `sklearn.cluster.KMeans.predict` to predict which cluster each of the validation set entries best fit in.

I then started implementing the prediction scripts, both of which rely on the same basic algorithm but with different data pre-processing. While the retroactive prediction script takes in just the cleaned data for the given day, the future predictive script computes a new predictive table for the day by grouping each of the days in the predicted cluster by the time and taking the average for each ride. The predictive table looks identical to those shown above, but instead of actual values,

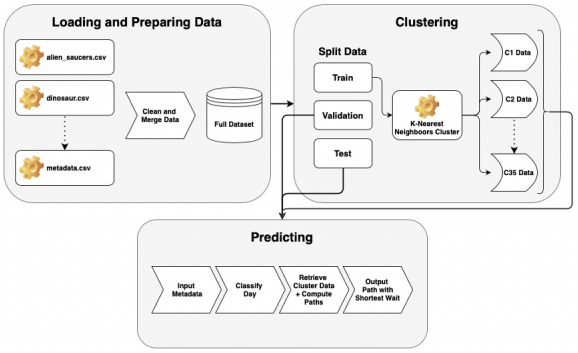

Figure 10: Diagram of implementation of program.

they are predicted values derived from the averages in the cluster. The algorithm is as follows:

1. Obtain the relevant data (for either retroactive or future predictive) to feed the script.
2. Iterate first through each of the timestamps and then each rides at each times stamp, computing the shortest possible path from every starting point.
3. Keep track of the path with the shortest possible wait time, as well as the start time for that path and the total wait time.
4. Once every path had been computed, return the predicted path, predicted start time, and predicted total wait times of the shortest path.

I also had to define an accuracy function that took into account that small errors within the predictions might not be fatal to the overall plan. Thus, I wanted something that would give punishments fitting to the crime and decided to compute the actual time that would be waited on the given day using the prediction path and start time and compare it to what the actual shortest path wait time was on that day. The accuracy function is defined as follows:

1. Compute what total wait time would be on the day given predicted start time and order of rides Use threshold of ±30 mins for a "correct" prediction (comparing computed total wait time to actual shortest total wait time)

The following are examples of output from the algorithm:

```
MAGIC KINGDOM - 2019-08-17

ACTUAL

Path: SPLASH PIRATES DWARFS
```

```
Start Time: 08:55:02

Total Wait: 70.0

PREDICT

Path: PIRATES SPLASH DWARFS

Start Time: 08:56:00

Total Wait: 61.22612847222222

Computed wait time: 70.0 --> CORRECT

MAGIC KINGDOM - 2019-12-26

ACTUAL

Path: SPLASH PIRATES DWARFS

Start Time: 07:20:02

Total Wait: 36.875

PREDICT

Path: SPLASH PIRATES DWARFS

Start Time: 07:46:00

Total Wait: 37.34375

Computed wait time: 68.75 --> INCORRECT
```

5. Evaluation

5.1. Experiment Design

I designed my evaluation plan to pinpoint the unaccounted variables that control wait times in the parks. By implementing my predictive scripts to output the path with the shortest wait time by computing all possible paths and customizing the accuracy function to give some leniency for small errors likely caused by mishaps in the data cleaning, I can attribute the majority blame for inaccuracies in scripts to be outside factors. In `predict.py`, I define scripts that select random subsets of size 30 of the validation data, compute the actual and predictive paths for each, compute

the accuracy value for each, and then output a final accuracy score for the set. I ran this script in its final iteration three times for each of the parks. I chose to do subsets rather than the full validation set due to the time inefficiency of the prediction algorithm (see Subsection 6.2 for more details on this).

I was also lucky enough to secure a true test set in the final days of working on the project as the 2020 data was released by TouringPlans.com. While the available data is sparse and patchy as compared to other years, it is interesting to see how COVID and limited park capacity has shifted the accuracy of the algorithm and the behaviors of park goers.

5.2. Results

When computing the accuracy rates on the validation subsets, I test a number of different threshold for the accuracy function - 15, 20, 25, 30, 45, 60. While the algorithm still performed fairly well at the lower threshold, the optimal threshold was 30 minutes. Each of the results below reflect that threshold. Thus, the overall accuracy on the validation set was 74.45%, well above the benchmark of

	Iter 1	**Iter 2**	**Iter 3**	**Avg**
MK	0.866	0.766	0.733	78.89%
EP	0.833	0.9	0.866	86.67%
HS	0.566	0.766	0.633	65.56%
AK	0.733	0.6	0.666	66.67%

Table 1: Accuracy rates for validation set.

60% that I set for myself in the beginning. As you can see, there is a lot of variety between the parks. While EPCOT is fairly stable in its prediction, Hollywood Studios is much more unpredictable. This can be attributed to a number of factors - the number of rides included in each park, the inclusion of new, highly sought after rides in Hollywood Studios (including the addition of Star Wars Galaxy's Edge) that bring guests to the park, the availability of extra magic hours in each park.

Ultimately, I can summarize my findings of the trends that influence wait time rather succinctly, unlike the bloggers who have devoted their lives to planning Disney vacations. My most shocking finding was that weather played practically no role in wait times. Come rain or shine, Disney

visitors are ready to tough out the conditions and wait on line no matter what. On the other hand, a not so shocking revelation was that the wait times are highly impacted by large events in the parks and holidays (especially if children have off of school). Most importantly, new openings in a park cause huge spikes in attendance of the whole park, not just that ride. This is why we see so much variability in Hollywood Studios which is practically all new right now and to a lesser extent Animal Kingdom which saw the opening of a new land not long ago. High crowd attendance causes volatility in the wait times, constantly going up and down, making it hard to predict when the best time to get on line for a ride is and lowering the overall accuracy. However, when in doubt, it seems the best time to hit the parks is early in the day. Nearly all of the predictions computed thoroughout this project occurred during the first 2 hours of opening time. These findings, such as hitting the parks early and being mindful of new attraction openings, can also be applied to other amusement parks when planning other trips.

Since the available 2020 was limited, I chose to use subsets of 20 rather than 30 when computing these accuracy scores. I again used a threshold of 30 minutes when computing the accuracy for each entry. The overall accuracy on the test set was only 60%, which while it hits out benchmark

	Iter 1	**Iter 2**	**Avg**
MK	0.85	0.75	80%
EP	0.9	0.9	90%
HS	0.4	0.2	30%
AK	0.55	0.65	60%

Table 2: Accuracy rates for 2020 test set.

goal of adequacy, is a significant drop from the accuracy of the validation set. This is likely due to the unpredictable crowd size in the parks. While EPCOT is less popular, it see less variability in wait times. On the other hand, Hollywood Studios is almost always booked to capacity on any given day as it is a huge crowd draw right now with its new lands. Overall, I feel the 2020 data further emphasized the importance of being mindful of crowd sizing when planning a trip and, in times of COVID, maybe avoiding the newer attractions until there's a vaccine distributed.

6. Summary

6.1. Conclusions

Over the course of the last three months, I was able to to clean and merge the publically available data sets for 14 different WDW ride wait times and use them to build a predictive program that outputs the optimal path through a chosen park. Its basic design is as follows:

- Take in a park and a day's relevant metadata to determine K-NN
- Build a data table of predicted wait times for that day based on existing K-NN cluster data
- Predict the path through all rides with the shortest total wait time
- Predict the best time to begin your day at the park
- Predict your total wait time to follow that path at that time
- Have a prediction classified as accurate when it predicts the correct path and start time or errors within the range of margin as outlined by accuracy function

This prediction program was largely successful, reaching an overall accuracy rate of 74.45% on the validation set across all four parks. In addition to correctly predicting paths, the program revealed trends in the data within its inaccuracies, pointing towards crowd level increases from new attractions and popular holiday seasons as some of the main driving forces behind surging wait times and irregularities across data clusters.

Thus, I accomplished my goal of using available WDW wait time data in order to predict best paths through a park on a given day. I also used these predictions to answer my main question of the underlying causes of wait times in WDW.

6.2. Limitations

While working with this data set, I encountered a number of limitations that hindered the performance and/or restricted the design of the prediction algorithm and its analysis. The most notable limitation was the availability of data. While I was able to gain access to wait time data recorded over the last 5 years for 14 different attractions, this number is small in comparison to the total

number of attractions in the park. This was somewhat favorable as it forced me to use all of the available rides when computing a path and optimize strictly for time rather than utility, allowing a later analysis of results that highlighted the underlying causes of wait time data. However, it would have been nice to have an even number of rides in each park and a more even distribution of old and new attractions in each of the parks. Currently, EPCOT's predictions are more or less stagnant as I only have data for two relatively old rides while Hollywood Studio's predictions are much more unpredictable as half of the data is from new, very popular attractions. I was not able to collect my own data for this project as the backlogged data is not publically accessible via Disney.

The limited amount of data also allowed me to use a non-optimal algorithm, computing all possible paths through the park and boosting accuracy of the algorithm. This sometimes yielded incredibly slow performance - the elapsed time to compute both the actual and predicted paths and get an accuracy label for a single date took just under five minutes. While the improved accuracy was a bonus, the slow performance was not ideal during the evaluation stage. In order to compute an accuracy score for all four parks on the entire validation set would've taken nearly seven days to run, so I had to opt to compute validation set accuracies in smaller batches and takes averages instead.

While I originally thought I would not be able get values for 2020 for a COVID analysis, I was lucky enough to notice that TouringPlans had uploaded the 2020 wait times at the beginning of the new year, allowing me a little time to play with it. While this data is more patchy than other years, it was still nice to get to see it.

6.3. Future Work

There are many potential expansions to this project that I did not have the time to implement within the given time frame. The potential addition that is most realistic is expanding the prediction algorithm to include "Park Hopping", providing routes across multiple parks in WDW while accounting for travel time between them. I could accomplish this by combining park data sets and including constant values when computing paths to represent travel time. I would also have to make

sure the paths completed all rides within one park before hopping to another.

Another potential expansion would include manually collecting data for all of the rides in WDW to build a more comprehensive predictive program that takes into account user utility. I would need to write a scrapper to collect the data from the MyDisneyWorld App. This expansion would require a couple years of data collection in order to match the robustness of the current data sets. It would also require major changes in the predictive programs implementation to make it more time efficient.

However, perhaps the future work I am most excited about is getting a chance to test out my predictive program in real life by inputting the conditions of the day that I'm visiting and following the path to see if it was accurate of not. Hopefully, I will get the chance one day soon.

7. Acknowledgements

A big thank you to my advisor for this project, Professor David Dobkin, along with the entire IW07 seminar for all of your input and guidance over the semester.

I would also like to thank Len Testa of TouringPlans.com for the collection and public release of wait time data at Walt Disney World for academic purposes and for his work on TouringPlans that inspired this project.

8. Ethics

This paper represents my own work in accordance with University regulations.

/Kelly Petrino/

References

[1] S. Chen, "Expedition everest wait time machine learning models," GitHub, June 2019. Available: https://github.com/schenx/Disney/blob/master/Everest_Report.pdf

[2] B. Sehlinger and L. Testa, *The Unofficial Guide to Walt Disney World 2020*. Birmingham, Alabama: Unofficial Guides, 2020.

[3] A. Stilwell, "Len testa and touringplans.com — part two: How does it work?" Coaster101, August 2018. Available: https://www.coaster101.com/2018/08/14/len-testa-and-touringplans-com-part-two-how-does-it-work/

[4] L. Testa, "Disney world wait times available for data science and machine learning," TouringPlans.com, June 2018. Available: https://touringplans.com/blog/2018/06/25/disney-world-wait-times-available-for-data-science-and-machine-learning/

[5] "Disney world personalized touring plans," TouringPlans.com. Available: https://touringplans.com/walt-disney-world/touring-plans/personalized

[6] "Walt disney world lines app," TouringPlans.com. Available: https://touringplans.com/disney-world-app

[7] M. Weinberger, "Meet the computer scientist using artificial intelligence to help 140,000 paying customers plan the perfect disney vacation," Business Insider, June 2019. Available: https://www.businessinsider.com/touringplans-disney-world-len-testa-interview-2019-5