DISNEY: ENHANCING THE GUEST EXPERIENCE

https://www.informs.org/Impact/O.R.-Analytics-Success-Stories/Industry-Profiles/Disney

Table of Contents

Abstract

Guests from all over the world come to the Walt Disney World Resort in order to enjoy many of the entertainment and attractions the resort has to offer. But, there are lots of careful planning that happens “behind the scenes” in order to run the operation smoothly and analytics plays a big role in ensuring the guest experience is maximized. This project aims to create a hypothetical analytics solution on real problems solved by Disney’s Analytics team through knowledge gained in Georgia Institute of Technology’s Introduction to Analytics Modeling course. The Introduction to Analytics Modeling course is a survey course and therefore, readers should not expect this paper to dive deep into mathematical theory behind these models. I also have no idea how any of Disney’s database systems and internal tech stack works so this proposal is completely hypothetical. We’ll focus on maximizing the guest experience by forecasting their wait time per attraction, offering customized vacation packages and planning our labor distribution to match customer demand.

Forecasting: Wait Times

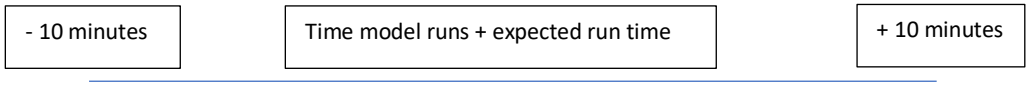
The goal of this forecast is to help guests choose whether they want to enter the line, take a FastPass ticket or return to the attraction later in the day by projecting their wait times at the front of the attractions.

What is a FastPass?

The FastPass system is one of Disney's most used features. It's basically a tool that lets you skip the regular line. At the time of this writing, you can make up to 3 FastPass selections in advance for a single theme park per day. There are 4 theme parks at the Walt Disney World Resort so you can essentially make 12 selections per day maximum in advance. You can make your bookings up to 30 days in advance of your visit or up to 60 days in advance if you've got a room reservation at one of the selected hotels on their website. After you redeem your initial set of FastPass selections, you can make another FastPass selection until the last arrival window has passed. I suggest you make these FastPass selections on the App rather than the kiosk (and make sure you bring a portable phone charger! I made the mistake of not bringing one when I was a Disney noob, but thankfully my phone lasted until the end of the day). One last important note about the FastPass system is that guests can enter the FastPass line at any time during their 1-hour window and this is the part that makes forecasting wait times a challenge.

One of the notes in the article is that these models are run every 5-10 minutes in order to give guests real time estimates so our hypothetical model will update with the same frequency. With respect to how often these models are run and that the aim is to give guests as real-time of wait times as possible, I would suspect that we'd have to choose models that have quick run times. The methodology below will have to be run for each attraction and entertainment as there are part of the Disney experience that are more popular than others.

Allotted time that this methodology will model is



General Overview (TLDR)

Conditional Probability: How many of our FastPass guests are going to show up?

The existence of the FastPass makes forecasting complicated to determine. The amount of people in the FastPass line will affect the wait-time for those in the standard queue. Also,

FastPass guests can enter the FastPass line at any given time during their 1-hour window. We'll utilize conditional probability for this part of the problem.

Given:

- The total number of FastPass guests that can enter in the allotted time.
    - The FastPass system is digital and so every guest's allotted time per attraction should be in the Disney database. In this case, the database is hopefully constantly refreshed every minute.
- The historical time of entry into the FastPass line.
    - Guests generally have to scan their FastPass in front of the FastPass line before entering so this data should be available in the database.
- The historical total number of FastPass guests that can enter at any given time.
    - Because the FastPass system is digital, this should be in the database.

Use: Conditional Probability as follows

Let

- x = the number of guests entering the FastPass line in the allotted time
- y = the total number of FastPass guests that can enter in the allotted time

$$P(x|y) = \frac{P(x \cap y)}{P(y)}$$

Note that this is equivalent to Bayes' theorem $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$.

Having historical data in the database makes calculating P(y) and $P(x \cap y)$ a simpler task as you'd take the data in 20-minute segments within 1 hour of the start of run time (if it's for a weekday, take only weekday data and vice versa for weekends), randomly select (I usually bootstrap) from the most recent data and then calculate the probability of having exactly y FastPass guests that can enter in the allotted time for that particular attraction over the course of many iterations (which should converge with many iterations). The same methodology can be used for having exactly x guests in the FastPass line and y total FastPass guests that can enter in the allotted time. It is much harder to determine P(y|x) which is why I shall opt for the conditional probability version of the equation. At the time this model begins running, we'll assume that there are going to be the y FastPass guests already registered in the database that can enter in the allotted time and that any additional FastPass guests that can enter in the allotted time will be an insignificant amount. Also, another thing to note is that there is usually a maximum y for each attraction. So, on a busy day where the change in y can become significant, I'm gambling that we'll have already hit the max y by the time this model begins running. Running the conditional probability for all possible x where $0 \leq x \leq y$ and x, y are integers, we will assume x of the max P(x|y) will be number of guests entering the FastPass line in the allotted time.

To: Determine the most likely number of guests entering the FastPass line in the allotted time

Linear Regression – Elastic Net

We'll use Linear Regression in order to estimate the approximate wait times of our guests should they choose to enter the standard queue. These will typically have fast run times, allowing the data refresh as close to real time as possible. While it is true that y can range from $-\infty$ to ∞ in a Linear Regression, I do not foresee the Linear Regression used for this case to predict very unheard-of outputs. The reason being, each attraction/entertainment has only a fixed number of seats and their run times are fixed. As such, there shouldn't be very many variations in the predictions. Also, the independent variables I'm deciding on are very few compared to the large amount of data points and, thus, is less influenced by outliers and randomness in the data. Also, the independent variables chosen should have a fairly linear relationship with wait time.

Given:

- The expected number of guests entering the FastPass line in the allotted time as outputted by our conditional probability model
- The approximate number of guests in the standard queue currently
    - If you go to Disneyland, you'll see attendants throughout the queues. I believe that these attendants are there to not only maintain the queues but to give the park an estimate of the number of people currently in line. A line at ending closest to the first attendant could have approximately 10 people and a line ending closest to the second attendant could have approximately 20 people.
- Weekend or Weekday
- Whether it's a national holiday
- Weather (i.e. rainy, cloudy, etc)
- These next two variables are only important if the attraction is new and does not have a good amount of its own past data
    - The attraction run time
    - Whether this is a show or a ride
    - Available seating
- Historical wait times – our response variable
    - These also should be in the database. I have heard that Disney used to give guests tickets which the frontline attendant would hand out and write the start time and the ride attendant would be given and write the end time. I suspect that nowadays wait times are still obtained, especially with the digital nature of how guests can interact with the park and refreshed at almost real time.

Use: Linear Regression with Elastic Net as follows.

- Remove outliers
    - This point cannot be stressed enough and must be done before scaling the data.
- Impute missing values

    - Removing missing values will cause a loss in the amount of data points available. Instead, it's better to impute them especially if there's another closely correlated variable available in the database. In that instance, I would suggest the use of imputation with perturbation in order to account for variability which is simple enough to do with the mice package in R.
- Scale
    - This point also cannot be stressed enough due to the coefficient constraints of Elastic Net. Also, note that there will typically be less people in the FastPass line than general line but we wouldn't want their coefficients to be on different scales. Scaling has many benefits even if you're not using Elastic Net.
- Linear Regression via Elastic Net
    - Cross validate via K-Fold Cross Validation
        - The training data is split into k parts (k=10 is the most common)
        - In each iteration, the data is split on k-1 parts and evaluated on the remaining untrained part
        - The average of the k evaluations is the estimate of the model's quality
        - The final model (model with the max quality) is then trained again using all of the data before being used on the test data
    - Linear Regression
        - One of the most basic models that is well known throughout the world and one of the most explainable models as well.
          Let
            - y = wait time
            - x_i = the ith independent variable (see Given section for our independent variables)

$$y = a_0 + \sum_i a_i x_i$$

    - Elastic Net
        - Advantages
            - Variable selection
            - Predictive benefits
        - Disadvantages
            - Arbitrarily rules out some correlated variables
            - Underestimates coefficients of very predictive variables
        - Of course, the scale of these advantages and disadvantages are influenced by your choice of λ. Fun fact: setting λ to either 1 or 0 will result in Lasso or Ridge Regression respectively. We'll pick the λ, τ with the maximum R^2.

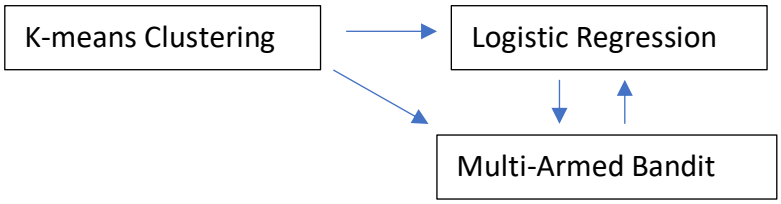
$$\text{Minimize } \sum_{i=1}^{j}(y_i - (a_0 + a_1 x_{1i} + \cdots + a_j x_{ji}))^2$$
$$\text{Subject to } \lambda\sum_{i=1}^{j}|a_i| + (1-\lambda)\sum_{i=1}^{j} {a_i}^2 \leq \tau$$

<u>To:</u> Predict wait time

Determining Optimal Vacation Packages

The goal of this model is to present our guests with the vacation package we think they'll most enjoy. Our model output would be featured in Disney's website and used by call center agents to present to customers. Because this isn't a time sensitive issue, I'd favor exploration over exploitation in this case due to vacation packages taking a while to develop. As such, a good starting point would be to refresh and re-run the model at the monthly level.

General Overview (TLDR)



K-Means Clustering

We'll start off with K-Means Clustering in order to determine the best groupings of our customers. Often times when marketing, I've noticed that analyzing campaigns will indicate that different types of audiences will react differently to different types of products and offerings. We'll want to present our guests with as personalized an experience as possibly.

A high-level overview of what k-means clustering does is:

1. Pick k cluster centers within range of data
2. Assign each data point to nearest cluster center
3. Recalculate cluster centers (centroid)
4. Repeat until no changes

I would suspect the independent variables below to be in Disney's database since these are typical consumer inputs when creating a My Disney Experience account and these should be refreshed at real-time with the biggest lag being 1 day.

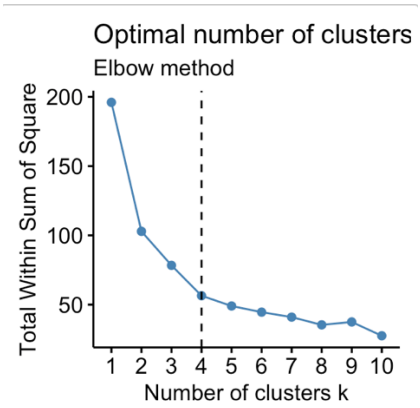
<u>Given:</u>

- Guest Age
  - Different age groups are at different parts of their lives. One of the things this'll help us determine is how much time they can spend on vacation which is one of the main differences in the vacation packages (i.e. spend x nights y days at…)
- Guest Location (State)
  - States might have different perceptions of Disney. Californian residents are close to the Anaheim Disneyland and going to the Walt Disney World Resort might be an extension of their local Disneyland.

- Character
    - This might give us an indication of which parts of Walt Disney World Resort appeal the most to them.

Use: K-Means Clustering as follows.

We'll select our K using the elbow method. An example of it is below.



The 4 in this chart highlights the point of inflection, which is how we determine our optimal k. The rest of the algorithm will run as usual.

To: Divide our guests into groups.

Logistic Regression

For each cluster and vacation package combination, we'll run a logistic regression to output the probability that the cluster will choose the vacation package. After doing so, we'll rank the vacation packages in each cluster by their probabilities.

The probabilities are calculated by

$$\log\left(\frac{p}{1-p}\right) = a_0 + \sum_i a_i x_i$$

Where p is the probability of the cluster choosing the vacation package.

The data collection portion of this is identical to the K-means part. The only difference is the historical data of whether or not the members of this cluster chose the vacation package, but this should be digital and thus, I suspect the latest lag of this information in the database to be 1-day.

Given:

- Cluster average age

- Cluster mode state
- Cluster mode character
- Historical data of whether or not the members of this cluster chose the vacation package

Use: Logistic Regression as follows.

- Scale
  - Scaling data tends to give much better results when running a regression than not scaling.
- K-fold Cross Validation (See page 6 for an overview of what this is)
- Logistic Regression

To: Determine probability of cluster choosing vacation package

Multi-Armed Bandit

The Multi-Armed Bandit is essentially a way of mixing exploration (gathering information) and exploitation (having an immediate value). In a way, this will influence the logistic regression model and the results of this will be the new historical data of whether or not members of the cluster picked the vacation package. Because I'm using the logistic regressions to update the Multi-Armed Bandit probabilities, the logistic regression feeds the Multi-Armed Bandit as well.

The data needed for this will essentially just be the outputs of the previous two models.

Given:

- Clusters from the K-Means clustering
- Probabilities of each vacation package in each cluster as outputted by Logistic Regression

Use: Multi-Armed Bandit as follows

- Select the top 5 vacation packages in terms of logistic regression probabilities
  - Over time, as the best top 5 vacation packages become clear, these should change less frequently as probabilities tend to converge as the number of observations reaches ∞.
- Equally offer each of the vacation packages to each member of our cluster
  - Although we don't pick anything outside of the top 5 vacation packages for a particular cluster, those not picked for that cluster might be picked as a top 5 in another cluster.
  - The top 5 vacation packages don't necessarily remain in the top 5 in the first set of iterations because a customer not picking it will lower the probability of the vacation package while those not picked remain the same unless it's in a different cluster and the k-means will change the clusters in the next run of the entire methodology (although this shouldn't be likely to happen as time goes on given the low amount of independent variables for the k-means cluster). But, as

the probabilities converge with more observations, the top 5 packages should eventually become clear.

- A brand-new vacation package that has just arrived would be offered as a 6th vacation package to every member of every cluster for at least 1 month in order to gather information

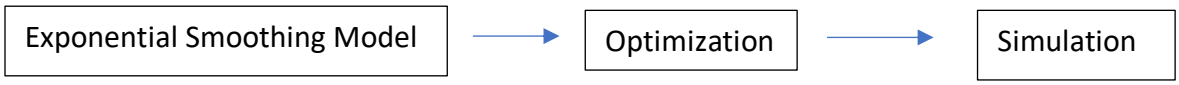
To:

- Customize offered vacation packages that better match resort guests' desires
- Update probabilities of each vacation package

Labor Demand Planning System

We need to ensure that there are enough staff to meet our customer demand for every 15-minute period at each location. Because staff schedules generally remain fixed for a long period of time, I suspect this model would need to be run for each location on a quarterly basis.

General Overview (TLDR)



Exponential Smoothing Model

This model would be run separately for weekdays and weekends as I suspect weekends will be much busier than weekdays. The exponential smoothing model is a model that is effective for shorter term forecasting then longer-term forecasting, considers trend and seasonality and puts more emphasis in more recent data. We'll only forecast sales at the weekly level for the next quarter at the time this model is run. For R users, this model is available as the HoltWinters function.

Given:

- Historical sales transactions for each location
  - I suspect this data would be in the database as sales transactions are digital and a latest lag data would most likely be 1-day.

Use: Exponential Smoothing Model as follows

Let

- S_t= the expected baseline response at time period t
- T_t= the trend at time period t
- C_t= the multiplicative seasonality factor at time period t

$$F_{t+k} = (S_t + kT_t)C_{(t+1)+(k-1)}$$

- Separate the data into 2 groups: weekday and weekend for each location
    - New locations would be a general model with data from each location in a similar category (i.e. quick service restaurants, merchandise locations).

To: Forecast sales at a weekly level for the next quarter

- We'll assume we have a linear distribution of sales for each day in a weekday and weekend.

Optimization

We'll use stochastic optimization in order to determine how many workers we'll need to match customer demand. One package that is useful for linear programming is pulp in Python. The model, like the Exponential Smoothing Model, will be separated into weekday and weekend optimizations. This model will have to be run to give an optimization at the weekly level as well.

Given:

- Hourly wage of each worker
    - The HR department should have this information and it's hopefully stored somewhere in the database with the most up to date data.
- Expected sales transactions for the week as outputted by the Exponential Smoothing model
- Maximum number of hours each worker can work
    - These should be in the database as hourly workers can state how many hours they're willing to work. If there's no information of this, we'll assume the 40 hours per week for full time and 20 hours for part time.
- Minimum number of hours each worker can work
    - If this isn't stated, we'll assume 0.
- Average amount of time it takes for a worker to assist a customer
    - We'll ask workers to estimate average amount of time it takes for them to assist a customer via an online survey that'll be distributed quarterly. This will be a rough estimate and we'll take the average of all the workers inputs. By refreshing this survey, we'll put more emphasis on recent work flow.

Use: Optimization as follows.

Let

- x_i = the number of hours worker i works
- c_i = the hourly rate of worker i
- d_d= demand of week d (week being modeled)
    - This will be expected sales transactions of week d * average amount of time it takes for a worker to assist a customer
- M_i = the maximum number of hours for worker i

- m_i = the minimum number of hours for worker i

Constraints:

$$m_i \leq x_i \leq M_i$$
$$\sum_i x_i \geq d_d$$

Objective:

Minimize $\sum_i c_i \, x_i$

To: Number of hours each worker will work

Simulation

We'll simulate our model to determine how many workers will work at any given time for each location. This model, like the rest of the models in this section, will be run for each location and separated into weekday and weekend simulations. This simulation will be constructed in a way that assumes we'll be using the SimPy package in Python.

Given:

- Number of hours each worker will work by our Optimization model
- Average transactions per hour
    - Transactions are digital so this should be available in the database with a max lag time of 1-day. We'll use this as a way to determine the number of people that will enter the location per hour. It's a rough estimate as the average may be understated for data on a very busy day, but digital data is usually cleaner than having park attendants count the number of people that enter the location. Also, not everyone that enters makes a transaction/needs to be helped.

Use: Simulation as follows.

- Each worker is a resource
    - When they hit the number of hours they will work as determined by our optimization model, they will not be a resource anymore.
    - This variable will be the one we'll play with as a parameter to see how to distribute a worker's hours throughout the weekday/weekend in order to meet customer demand.
- Average transactions per hour will be assumed to have a Poisson distribution
    - Each person that enters the location will enter a single queue and wait for the next available resource (worker) to assist them.
- Average amount of time it takes for a worker to assist a customer
    - We'll assume uniform distribution.
    - We'll ask workers to estimate average amount of time it takes for them to assist a customer via an online survey that'll be distributed quarterly. This will be a

rough estimate and we'll take the average of all the workers inputs. By refreshing this survey, we'll put more emphasis on recent work flow.

To: Determine workers working per hour at each location.