Group Members: Phil Greene, Dan Forbush, Martha Menchaca
IDS 575 Final Report

Predicting DIVVY Station Availability

Abstract

Divvy is a bike sharing system where customers rent bikes from one station and return it to another. Station availability is an inherent problem with the system that Divvy management must deal with. We first provide general analysis of Divvy trips to show the busiest times in Divvy usage and how that differs on weekdays versus weekends. We then attempt to create models that predict when stations need to be restocked either with more bikes or need bikes taken away to make room for further trips. Overall, machine learning models are useful to provide prediction of when a station needs to be restocked, with AdaBoost modeling performing the best.

Introduction

Divvy is a bike sharing system in the City of Chicago that serves residents by offering bikes as a form of public transit. Divvy bikes are organized into hundreds of docks across the city, and people bike from one dock to another within an allotted time that depends on the service purchased by the customer. Some people, called “customers," buy short-term 24 hour passes, where they get unlimited 30-minute rides within that 24 hour periods, and other people, called “subscribers,” buy annual passes where they get unlimited 30-minute rides (raised to 45-minute rides in 2018) for a full year.

Divvy has released a data set, available on Kaggle, that contains millions of rows, each with 23 attributes. Each row represents one trip, where a bike is rented at one dock at one time and returned to another dock at a later time. The attributes contained in each row include identifying information, several variables about which docks are used, whether using those trips are taken by customers or subscribers, information about the rider, information about weather, and variables about the start and end location.

In this report, we look to this dataset to try to explore one of the central problems that riders experience when using Divvy. There are only a certain number of bikes and docks contained in each station, and this creates a limitation for riders: sometimes when a rider wants to rent a bike from a dock, they will arrive and find that that dock is empty and does not have any bikes available to be used. Conversely, sometimes when a rider has a bike and arrives at their destination dock, they find that the dock is full and there are no open spots available, so they must find a different dock to return their bike. We try to use the dataset to come up with different models that can help managers of the Divvy system predict when stations are empty or full.

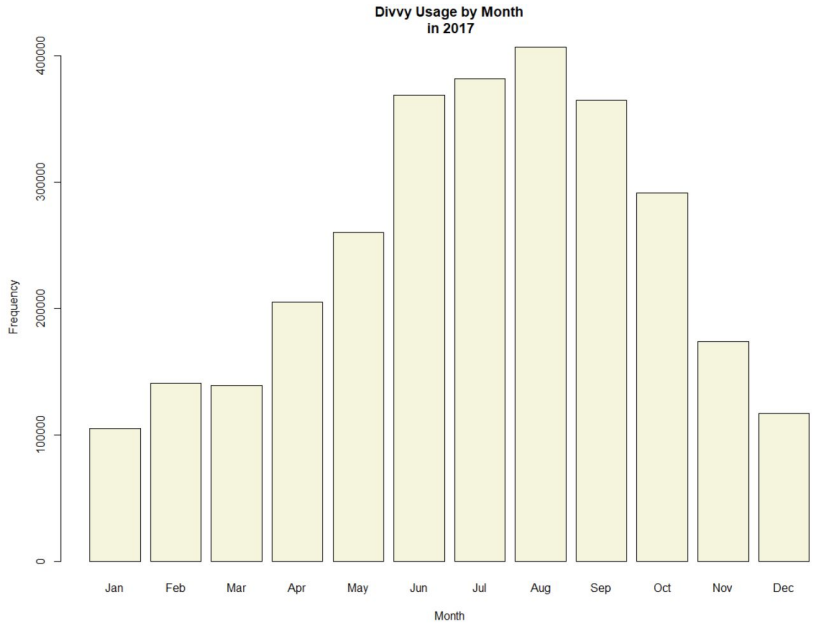
Related work

Researching the topic, several online works were found that were relevant to our topic and helped us develop which models to implement. The first was titled “3 Takeaways from the Divvy Bikeshare Data Challenge” which detailed results of a Divvy bike share system contest from 2015 that encourage designers, developers and scientists to create visualizations that

represent the data and awards in seven categories including Most Original, Most Insightful, and Most Comprehensive. Some of these entries compared usage patterns of annual members and day-pass users using three metrics: trip length, route and frequency; weather and ridership correlations and Divvy bike measure up against the CTA in a station-to-station race. A second resource work was titled “Divvy Bike Use Data Analysis and Recommendations using Tableau” and detailed analysis of the use data. In addition to identifying the busiest stations, days and times, recommendations made were to install a new station in Near West Side and consideration of a Metra- Divvy joint pass. A third relevant work was titled “Chicago Divvy Bike Analysis” presented analysis that detailed seasonal analysis including rental duration by season, duration of rentals, rental time, rental duration by day of week, rental duration and rental time by customer type, and rental station analyses.

Data Preparation

Though the Divvy dataset contains extensive information about its operations and the trips its customers take, we ran into two problems early in our analysis. The first issue had to do with the size of the dataset. The Divvy dataset contains several millions of rows of data, and that makes analyzing the data costly in terms of computation time and effort. For our model, we had to limit our data, and because we are examining station usage and when they may be full or empty, we choose to focus on the busiest month of the most recent year for our training data. The Divvy dataset was released in January 2018, so the most recent year was 2017. After splitting the data, we determined that the busiest month for Divvy was in August, as is shown in the figure below:



Additionally, we selected the month of September for our test data by which to compare the performance of our models. It was comparatively busy but no month was as busy as August.

The second issue that we ran into has to do with a limitation of attributes present in the dataset. The dataset does tell us which dock the bike was rented from, which bike the dock was

returned to, and the total number of docks available at both the starting dock and the ending dock. However, there is no information telling us exactly how many bikes or docking slots are available at a particular dock when the bike is either rented or returned. Therefore, we don't know directly from the dataset when a station is full or empty.

To solve this second problem, we created a simulation based on iterating through each line of data to try to identify stations that are losing more bikes than they are gaining (and therefore becoming empty), or vice versa, where they are gaining more bikes than they are losing (and therefore becoming full). Below is the code for that loop:

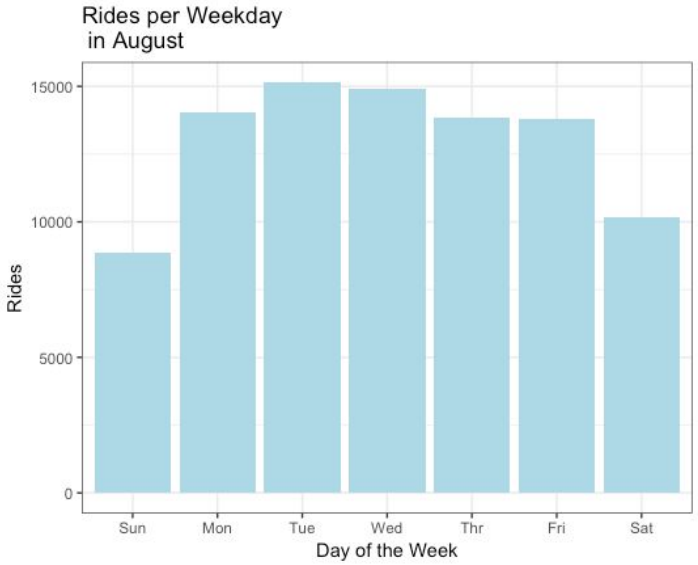
```
for (j in 1:length(unique(Divvy$DateTime))) {
  for (i in 1:length(unique(Stations$Stations))) {
    p <- nrow(Divvy[(Divvy$to_station_name == Stations$Stations[i] & Divvy$DateTime == unique(Divvy$DateTime)[j]),]) -
      nrow(Divvy[(Divvy$from_station_name == Stations$Stations[i] & Divvy$DateTime == unique(Divvy$DateTime)[j]),])
    NetGain[i,j] <-p
  }
}
```

This code goes through each station, hour by hour, and calculate a "net gain" value for all 565 stations that are part of the dataset. The net gain represents either the net number of bikes gained at the station (if the value is positive) or the net number of bikes lost at each station (if the value is negative). There are two limitations to this value: first, the loop only goes through the number of bikes gained or lost each hour, and is not more precise than that. Second, this simulation does not take into account one element of Divvy's operation: bike balancing. This is where Divvy removes bikes from a station that is full (or nearly full) or adds bikes to a station that is empty (or nearly empty). This action would create a bias within the data, because the net gain would reflect the activity of Divvy balancing the stations rather than riders taking or depositing bikes. Despite these limitations, however, we believe this method offers us significant insight into the station usage and can form a baseline of a predictive model that can help predict which stations might be full and which might be empty.

We then applied the net gain value to create a baseline for classification models. We set a threshold of half the station capacity as meaning that the station requires restocking. In other words, if within a given hour, a station loses half of its bike capacity, or it gains half its bike capacity, we classified that station as requiring a restock.

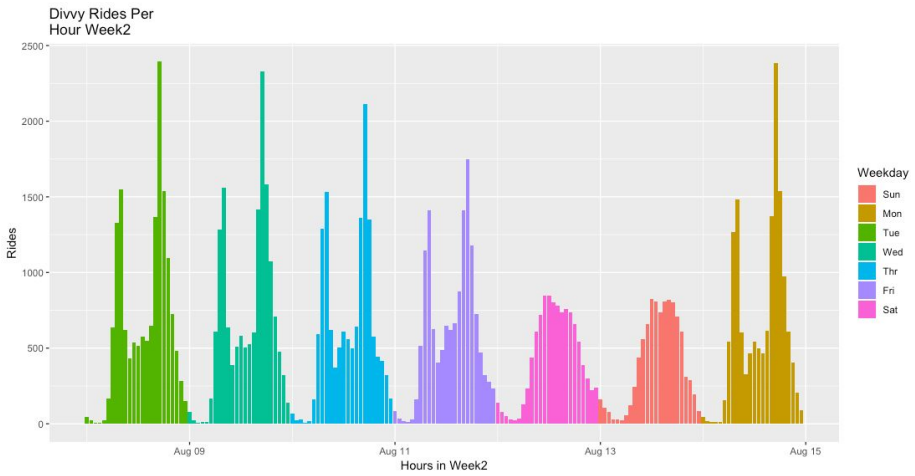
Data Exploration

Before creating and evaluating predictive models for this dataset, it is necessary to first examine what can be learned about station usage from an exploration of the raw data set. First, in the month of August of 2017, there is a pattern in terms of bike usage on each day of the week, as is shown by the following graph depicting the total number of trips per day of the week:



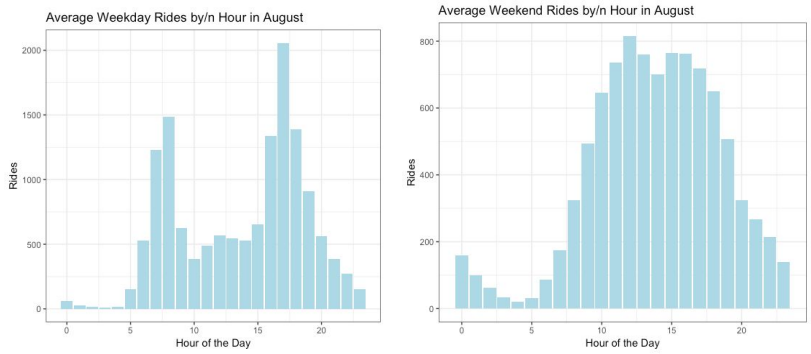
From this graph, the busiest day of the week was Tuesday, and weekdays were significantly busier than weekends.

Second, there is a pattern of usage within each day, that is distinct if the day is a weekday versus a weekend. The graph below shows the number of trips per hour for the week of Sunday, August 6, 2017 through Saturday, August 12, 2017:

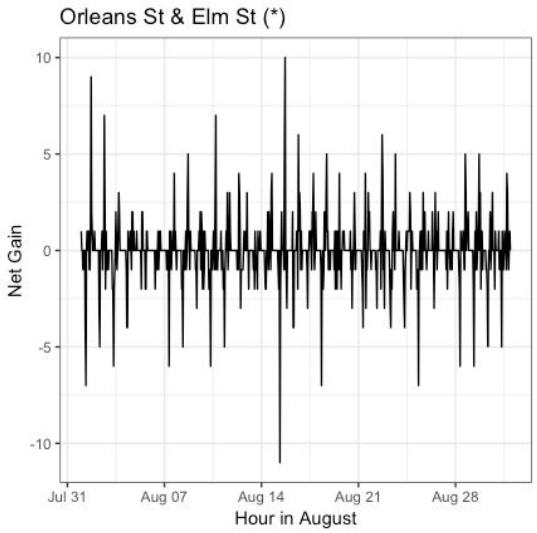


From this graph, there are two distinct peaks in each weekday, one during the morning hours and the other during the afternoon hours. This reflects that the concept of rush hour also translates to Divvy bike usage, where there are spikes during the morning and evening commutes of many workers. Additionally, these morning/evening peaks are not present during the weekends, where the distribution of rides per hour is a smoother bell curve. This pattern is reflected in every week in August, as is shown by the following two graphs which depicts the

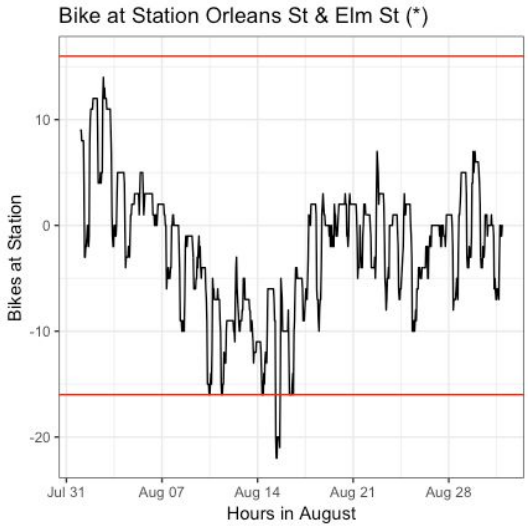
total number of rides per hour on all weekdays (in the first graph) and all weekends (in the second graph):



Using the iteration loop that produces the netgain values for each station allows us to examine the change in bikes over time. We can first plot the netgain value so we see the hour-by-hour change in bike total over time for a particular station. That graph for the station at the intersection Orleans St. and Elm St. looks as follows:



We can also iterate through the NetGain data to get a graph of the cumulative change in bikes at a particular situation. For this iteration, we set the starting value of the number of bikes at each dock as equal to half the total number of bike stations available at that dock. We can then add the NetGain value to plot the change in bikes over time:



On the graph above, the red lines added are the positive and negative station capacity, which are added to give a scale that is can be interpretable per station, because each station has a different number of bikes and docking stations.

Initial Model - Decision Tree

The formula we used for all models was based on predicting whether a station would need to be restocked based only on the day of the week of the start of the trip (i.e. whether the trip was started on Monday, Tuesday, Wednesday, etc.), hour of the day of the start of the trip, and the station the bike was initially rented from. Though we examined considering other variables such as weather conditions and gender, these did not meaningfully add to our models.

The initial model we have created is a decision tree model. Decision trees were logical given the vast number of stations we have available, and the idea that many of them could be easily classified as not requiring restocking because they are in an area that is naturally balanced by the typical input and output of the Divvy customers. Below is the confusion matrix for the decision tree's success in classifying the training data (Figure A) and test data (Figure B):

```
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0 347941   9446
         1   8418  41159

               Accuracy : 0.9561
```

A

```
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0 311960  13524
         1  13304  26408

               Accuracy : 0.9265
```

B

Second Model - AdaBoost

Our second model used AdaBoost to attempt to improve upon our first decision tree model. AdaBoost works by combining weak classifiers into weighted sums, thereby simplifying the decision tree and reducing the effects of outlier stations that maybe having an outsize influence on decision tree node-splitting. However, the use of AdaBoost did not appreciably improve the results of the decision tree, as is shown by the confusion matrices below (again, training data is Figure A and test data is Figure B):

```
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0 344757   6670
         1  11602  43935

               Accuracy : 0.9551
```

A

```
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0 308518  12000
         1  16746  27932

               Accuracy : 0.9213
```

B

It is interesting that the AdaBoost model has a slightly lower training and test accuracy compared with a lone decision tree. This is likely due to the large number of stations, and the possibility that taking out those stations (and their corresponding nodes) that are outliers actually decreased the accuracy of the model compared with a straight decision tree. However, the AdaBoost model did improve on the true positive prediction compared to the decision tree.

Third Model - Naive Bayes

Our third and final model used a Naive-Bayes algorithm to classify the probability that a station would need to be restocked based on the day of the week and the hour of the initial trip. Naive-Bayes is a logical choice for this situation due to the large number of stations and because it outputs probability of restock for each individual station, making it potentially useful as a station-by-station prediction model for Divvy management to use to predict when a station needs attention and bike balancing. Our model included a Laplace smoothing value of 1 in case there were stations that did not contain information or trips in the training and/or test data. Below are the confusion matrices for training data (Figure A) and test data (Figure B):

```
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0 346375  33526
         1   9977  17079

               Accuracy : 0.8931
```

A

```
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0 315048  27632
         1  10216  12300

               Accuracy : 0.8964
```

B

The training and test accuracies were not as high as the decision tree or AdaBoost models, but they are more consistent between each other, suggesting that Naive-Bayes may be

a more dependable model for predicting station restocking over time, even if the accuracy may be better predicted with the prior two models.

Conclusion

Overall, these models prove the difficult task that Divvy management faces when predicting which stations will need to be restocked at which times. Though our accuracy was high for all models, this is likely due to the large number of trips that do not require restocking. This makes sense - the vast majority of trips, individually, will not cause a station to be emptied of bikes or to become full. Therefore, the details of the confusion matrices for each model matters. For this reason, the AdaBoost model is likely the best model we have examined due to the high true prediction accuracy compared with the other two models.

Bibliography

Choi, H. (2016). Divvy Bike Use Data Analysis and Recommendations using Tableau. Retrieved from https://www.slideshare.net/HanbitChoi1/data-analytics-portfolio-1-hanbit-choi

Editor, (2015). 3 Takeaways from the Divvy Bikeshare Data Challenge | Shared-Use Mobility Center. Retrieved from https://sharedusemobilitycenter.org/news/3-takeaways-from-the-divvy-bikeshare-data-challenge/

Layne, L. (2015). Chicago Divvy Bike Analysis. Retrieved from https://rstudio-pubs-static.s3.amazonaws.com/63061_90f5136ffdf74740b6ba4ad8f2fd72fe.html