Design Document

Bike Demand and Distribution Optimization (BDaDO)

By: Andrew Blanco, Zachery Capell, Austin Huang, Philip Quinn, and Matthew Rodriguez

Scrum Master: Philip Quinn

Github: https://github.com/phillyq22/Senior-Project

Slack: senior-project-team-1.slack.com

*High level description*:

Bike Demand and Distribution Optimization consists of two algorithms, a Bike Demand Prediction Algorithm (BDPA) designed to determine the expected demand of a certain bike station at a certain time, which will be discussed further within this document. The second is a Bike ReDistribution and Incentivized Suggestions Algorithm (BRISA) designed to intelligently encourage users to drop off a bike in an area where demand is high or will be high in the near future. Additionally, suggestions will be given to users to incentivize picking up bikes in a nearby area where there is overcrowding within a station that does not have a large projected demand. Together, it is believed these two algorithms will reduce the need for forced redistribution of bikes via trucking during off-peak hours.

Given the numerical predictive nature of the BDPA, a regression machine learning model was implemented. Several bike sharing systems are currently in business in multiple large cities across the United States in order to provide an additional form of public transportation. Data has been made available for analysis through many of these bike sharing companies, and with it a predictive model is capable of being trained to determine future demand for any given station. Specifically, data spanning over the year of 2018 was pulled from Capital Bikeshare, a company based in Philadelphia Pennsylvania. The problem of predicting the future demand at a particular station falls under the category of time series analysis, an area of statistical analysis that has been researched for several decades. Multiple algorithms have been proven effective in various use-cases for such analyses, such as Random Forest Regression, Support Vector Regression, Long short term memory neural networks, and several others. The most trivial of all machine learning regression models is Linear Regression, which simply finds a line of best fit through the n-dimensional space derived from the trained features. The mathematical formulation of Linear Regression is a the linear combination where is the y-intercept (or bias), is a feature, and is its corresponding weight. Linear Regression utilizes an optimization algorithm known as gradient descent, which takes steps to reach a local minima such that the weights and bias for this line of best fit are optimized for making accurate predictions based on the training data. Given that this algorithm is linear in nature, it is incapable of prediction of non-linear mathematical dependencies, which can prove to be quite limiting. With that being said, it is thought to be a practical starting point for any regression model, which is why we have chosen to implement it, but it is certainly not the best. Demand is the predictor of the BDPA, which is defined as the number of bikes needed at a specific station at a specific time. For example, if every Monday to Friday at a certain station 5 people request bikes within a 20 minute period, it then can be deduced that this specific station has a demand of 5 during that 20 minute period. The machine learning algorithm learns all of these patterns based on month, day, time, station number, and daily weather. Daily weather data was found separate from the provided bike share data, but due to its validity in the BDPA, it was joined with the bike data in order to make more accurate predictions.

The BRISA that the team determined to be best for handling the process of making suggestions is a graph based algorithm that considers the paths between stations through a weighted system based on distance and predicted availability of each station. The predicted availability is derived from the aforementioned Linear Regression model’s demand output and known expected bike returns. Distance between stations will initially be derived as the euclidean distance between two pairs of longitude and latitude coordinates, which are conveniently provided by Capital Bike Share. Depending on the destination station of a user, the algorithm will suggest nearby stations that are no more than a mile away, and incentivize users to pick up/drop off bikes based on the demand of that station. With this, it is projected that the need for redistribution of bikes via trucking can be completely eliminated.

*Existing related work on the project topic:*

Research has been done before on predicting bike share patterns using machine learning. However, most of it focuses exclusively on check-in and check out numbers and redistribution using existing methods, like trucks. Very little, if at all, does it discuss users redistributing the bikes, either through suggestions or incentivization.

The key-words used for this search were “citi bike trucking issue” (<https://www.crainsnewyork.com/article/20150426/TRANSPORTATION/150429891/citi-bike-turnaround-so-much-promise-so-many-problems>). The article mainly highlights the many issues that Citi Bikes (New York City’s bike share program) has and how the CEO Jay Walder is attempting to fix up the bike station and app software. One bit of relevant information to BDaDO that is worth pointing out is the trucking issue they presented. Walder receives a “list of 9 or 10 that will receive overnight overhauls.” This exact problem is what BDaDO is trying to prevent, to minimize the amount of trucking needed to maintain the demand of bikes within a certain region. The article lacks an explanation if Citi Bikes will ever try to address the problem of trucking but it does mention that it is an issue.

The key-words used for this search were “Time series analysis for bike sharking” (<https://business-science.github.io/timetk/articles/TK03_Forecasting_Using_Time_Series_Signature.html>). This research and machine learning type of tutorial uses machine learning on a time series basis to predict the amount of bikes in a station in the future. Within basic time statistics are used to create the model. This differs from BDaDO since BDaDO uses time and weather data to predict what the demand of a certain station would be at a given time, as opposed to this algorithm which really will just count how many stations could potentially be at a station. It’s interesting to see the use of graphical information used by this project, it definitely gives a solid visualization of the data as opposed to just seeing the output which in this case is the count of bikes.

The key-words used for this search were “issues with bike sharing” (<https://www.nbcnews.com/tech/innovation/dockless-bikes-promise-future-transportation-litter-city-dallas-n866351>). These keywords were chosen to highlight some of the possible alternatives to stationed biking (Citi Bikes, IndeGo, etc.) which include dockless bikes. The article highlights a lot of issues with dockless biking but of course the biggest one is the issue of people not leaving the bikes in an obtainable area. The author, Tara Nieuwesteeg highlights the region of Dallas, where dockless bikes have been taking over the streets and unfortunately, the swamps. Bikes are literally being thrown into mud, rivers, creeks, you name it, and a bike has probably been found there. This lead to the companies to start having a lot more bike maintenance issues and cost companies an arm and a leg to keep bikes in rotation. Dockless bikes would not remove the need for BDaDO, just rather than recommend a specific station for a user it would recommend an area. Regardless of dockless or docked bikes, BDaDO can still be useful; but reading up on dockless bikes docked stations seem a lot more practical for the company and a bit inconvenient for the users. Whereas dockless is very convenient for the user, but as seen within the article, is terrible for the company.

The key-words used for this search were “why bike sharing is failing” (<https://motherboard.vice.com/en_us/article/vby5j4/bike-sharing-doomed-to-fail-dallas-limebike-ofo-transportation-cycling>). This article also highlights the issues with bike sharing within Dallas. The difference is this article speaks more to why it is failing, and according to Tracey Lindeman, it’s because of the people and the geography of the situation. “You can’t just drop bikes in an area of non bike riders and expect them to start using them” Lindeman states. This makes sense, not all areas are susceptible to biking (dockless or docked). Because of this people see them more as a chore/ annoyance, hence why they are always littered around the city or as the article photographed on a Dakkas billboard “Lime Bikes (are) Dallas’ Version of Road Kill”. It is important moving forward with BDaDO the users and area of the users are kept in mind.

Li, Zheng Zhang, and Chen (2015) use a Gradient Boosting Regression Tree to predict where bikes will be rented from and returned, grouping the stations as clusters. This is designed for use by the bikeshare company, rather than the user, so that the bikes may be allocated ahead of time. Li, et al. begin by grouping stations based on location and their history of bike transition. They also use a traffic prediction algorithm to determine the total number of bikes that would be checked out based on time and weather, and an algorithm that attempts to calculate the proportion of those bikes taken from each station cluster. Then, an algorithm is used to attempt to determine the paths the bike will take and, finally, check-out and return numbers are calculated from there. While there are some similarities between BDaDO and the system used by Li et al., such as the use of weather patterns in determining demand, the clustering of stations makes the algorithm unable to determine which specific stations near to the user’s destination most require the bike to keep the system balanced.

Kaltenbrunner, Meza, Grivolla, Codina, and Banchs (2010) also work to analyze movement patterns for Barcelona company *Bicing*. Their system focuses on using an Auto-Regressive Moving Average to predict a short time into the future of a station based on the patterns that can be fit to a line as well as its recent history and the recent history of the stations surrounding it. This model also analyzes a mobility pattern for the city. Yang et al. (2016) design a Random Forest Model using data from the world’s largest bike sharing system. Through the algorithm they design, they predict check-in and check out numbers with a relative error of 0.6. However, the system of Kaltenbrunner et al. and Yang et al. are designed mostly to predict only the the demand and to optimize where the bikes should be distributed by truck, and therefore does not calculate the immediate need of any given station, only the number of bikes predicted to be there in the near future so users will not attempt to retrieve a bike at an empty station or return to a full one, nor does it suggest alternatives or incentivise users for rebalancing the system.

*Problem solving approaches:*

To implement the BDPA machine learning strategies were deployed. The machine learning algorithms considered were linear regression, k-nearest neighbor, and random forest. When reading related work on this topic, k-nearest neighbor and random forest resulted in a more accurate predictions compared to linear regression. Linear regression being less powerful is much simpler than the other two algorithms, so it was chosen to as a good place to begin testing. As a stretch goal, k-nearest neighbor and random forest may be implemented in order to compare obtained accuracies.

The data that was fed into the linear regression model consists of: station number, date, time, average temperature, total precipitation, visibility, snow depth, wind speed, and the number of bikes rented (Demand). The station number provides the location where a bike is being rented. Date and time shows when the bike was rented. Average temperature, total precipitation, visibility, snow depth, and wind speed are all environmental factors that are believed to be obvious factors in a customer’s decision to rent a bike on a given day. The demand would then be the output of our linear regression algorithm, which will predict how many bikes will be taken at a future time.

In order to retrieve the bike purchasing data, comma separated value (CSV) files were downloaded from the capital bike share website. Originally, the each row of the CSVs contained information pertaining to single transactions. Each transaction detailed the start location, end location, duration of the trip, a datetime that the transaction began, and the customer’s membership type. This data was collected for all of the year 2018. Since this data was not in a form that could be trained by a machine learning algorithm, the CSV needed to be parsed through and reformatted. Ultimately, the goal was to prediction future demand, so the demand itself had to be derived from the given data fields. In order to derive the demand, a parser was written to first group together all transactions by station number and then datetime. For all transactions that began at the same time at the same station, it can be inferred that this is the demand for that station at that exact hour. We then count the number of entries that grouped together and made that our demand field for the row. As a result we have the number of bikes taken at a specific station for each hour.

The average temperature, total precipitation, visibility, snow depth, and wind speed (weather data) were found on www.almanac.com. This website did not have an API nor CSVs to download, so an html web scraper was used to gather the information needed. The website had weather data for a specific zip code and day displayed on a single webpage. This meant that a program was needed to go through multiple web pages and take the weather data from each day. Luckily, the URL structure of the website made this easy. To find the webpage with the data needed, the [zip code] and [date] were specified in this URL: https://www.almanac.com/weather/history/zipcode/[zip code]/[date]. The zip code 19019 was utilized since it is close to the location where Capital Bikeshare’s purchasing data is being made. He dates Jan-01-2018 - Jan-01-2019 are then looped through in order to capture the weather data for all days in 2018. For each iteration, the html of the webpage pertaining to the date and weather data was scraped.

The BDPA was trained on 80% of the data that obtained by Capital Bikeshare for 2018. The other 20% of the data was utilized for testing the algorithms accuracy. It was found that the algorithm correctly predicted the demands associated with data it had never seen before 93% of the time. Accuracies above 90% are considered to be excellent in the world of data science, which proves that the BDPA is performing very well. In order to further test the validity of this algorithm, data is being pulled from the Capital Bikeshare API and stored into a relational database using sqlite in python. With current data, demands can be derived and then predictions of the demands can be made by our BDPA. Proving that the BDPA can accurately and consistently predict demand correctly is the next step in assessing the BDPA.

The original BRISA that was implemented involved using a graph with connected nodes that would represent paths. By analyzing the paths and fields within each station at a given time, the BRISA should return all nearest routes with some sort of priority associated with each option. Upon further investigation and beginning to implement the algorithm, it was decided that a graph with node structures and connected paths would not be necessary. This would largely be due to the fact that information from an outside API would be needed to create the graph. Additionally, the objective of the project is proof of concept. The goal is to prove that the prototype and research will in fact improve the efficiency of bikesharing, and there should be an investment for its production. Thus, custom classes and data structures were used to implement the BRISA.

The algorithm was created as follows. It first processes all the individual bike stations in the function called preProcess(). The function takes in an input that is an array of stations. The function iterates through each station creating an array of Pair objects. The class Pair stores the field stationId and distance, thus providing the ability to know the distance from one station to another. To create the array of Pair objects, the function first calculates distance from given longitude and latitude coordinates between the stations. An outside distance calculator class was used to compute the distance. Once the distance is found, it is used to create a new Pair. The Pair is then stored in the array. The array is then sorted by using Arrays.sort from Java’s library. A custom compareTo() method was written so that the Pair objects could be properly sorted. This concluded the pre-processing stage. The result an array of Pair objects stored within each Station object, which indicate how far every other station is from itself. The function’s time complexity is O(nlog(n)) with a space complexity of O(n^2), n representing the total number of stations.

The method getSuggestion() takes in the input Station and distance. The function will return all stations within that distance in descending order of priority. Priority is determined by proximity of the station to the input station as well as the necessity variable that defines a state in Station used to calculate how urgent a station needs a bike returned to it. Based on these two factors, getSuggestion() returns an array of stations in order of priority. The function offers multiple suggestions to allow an end-user to have options based off of these priorities and the potential to incentivise the user to return bikes to specific stations based off of these priorities.

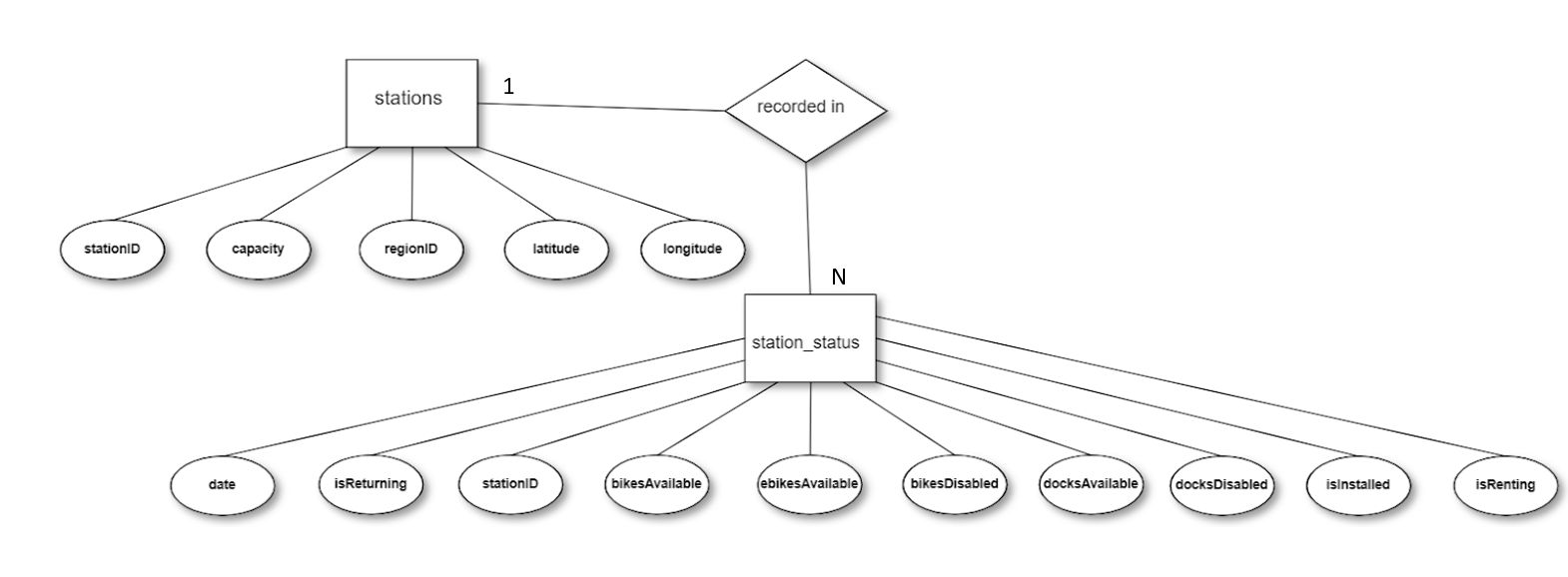
The getSuggestion() method is implemented as follows. It iterates through the array of sorted Pair objects discussed in the pre-processing stage to find all distances less than or equal to the input distance. Once each distance is found, the method then uses the input Station’s necessity variable to calculate priority based off of distance and necessity. The current calculation in terms of how each is weighed against each other is as follows: distance\*necessity^2. Thus, necessity has a higher precedence than distance. This calculation is certainly open for change and experimentation in the future. The priority is then stored in a SugPair object. The SugPair class serves the same purpose as the Pair class, as it also has a station id, except it stores the priority instead of distance. The SugPair is then stored in an array, also much like the Pair class. The SugPair is then sorted in descending order. The method finally returns an array of station ids, which represent the suggested stations for the users in order. The function’s time complexity is O(n) and space complexity is O(n), n representing the number of stations within the input distance.

*Functionality*

As described above in *Problem Solving Approaches*, the functionality provided by the BDPA is the bike demand prediction. The BDPA can provide these demand predictions when given new data that the algorithm was never trained on. For example, if the algorithm is fed a station number, the day of the week, the hour of the day, the month, and what the weather is on that day, the BPDA will be able to provide a demand prediction. The demand prediction is then fed into the BRISA algorithm in order to calculate the necessity field, which was also described above. With these two algorithms, the need for bike redistribution via trucking can be significantly reduced.

*Backend information*

As mentioned above, a relational database was createdin order to store data obtained from the Capital Bikeshare API. This database is not very complex, as it does not contain many entities. The data that we cared about from the API came in the form of two JSON files, which could be broken down into two relational tables. Please refer to the ER diagram below for further details.



*Tech Stack*

The tech stack utilized throughout this project was comprised of an Amazon EC2 Linux instance which had code written in Python deployed to it for the training of Machine Learning algorithms. The EC2 instance utilized includes a GPU, which is optimized for the processing of matrix operations. Matrix operations are essential to the training of machine learning algorithms, thus making GPUs an extremely valuable tool in this project. Python scripts were written utilizing several different python packages such as pandas, sklearn, sqlite, and many others. Pandas is an optimized way of importing and exporting data to CSVs, which is essential to the data preprocessing and algorithm training stages of this project. Sklearn hosts several different pre-defined and optimized machine learning algorithms which can be easily imported and used to train various machine learning algorithms effortlessly. Sqlite is a serverless relational database API that allows for sql queries to be implemented on imported data from CSVs. Database instances are created and stored locally rather than on a server. Several tables can be inserted into the database, making it a fully functional relational database. Sqlite allowed for the creation of a local database, which allowed insights to be made on the millions of rows of bike share data that were worked with. Java was utilized to complete the BRISA.

**Works Cited**

Dancho, Matt. “Forecasting Using a Time Series Signature with Timetk.” *Forecasting Using a Time Series Signature with Timetk • Timetk*, 17 Aug. 2017, business-science.github.io/timetk/articles/TK03\_Forecasting\_Using\_Time\_Series\_Signature.html.

Hawkins, Andrew J. “Citi Bike Turnaround: so Much Promise, so Many Problems.” *Crain's New York Business*, 24 Apr. 2015, www.crainsnewyork.com/article/20150426/TRANSPORTATION/150429891/citi-bike-turnaround-so-much-promise-so-many-problems.

Kaltenbrunner, A., Meza, R., Grivolla, J., Codina, J. and Banchs, R. (2010). Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system. *Pervasive and Mobile Computing*, 6(4), pp.455-466.

Li, Y., Zheng, Y., Zhang, H. and Chen, L. (2016). Citywide Bike Usage Prediction in a Bike-Sharing System. *IEEE Transactions on Knowledge and Data Engineering*, pp.1-1.

Lindeman, Tracey. “Bike Sharing Is Doomed to Fail in Most American Cities.” *Motherboard*, VICE, 5 Feb. 2018, motherboard.vice.com/en\_us/article/vby5j4/bike-sharing-doomed-to-fail-dallas-limebike-ofo-transportation-cycling.

Nieuwesteeg, Tara. “Dockless Bikes Promise the Future of Transportation, but Litter the City of Dallas.” *NBCNews.com*, NBCUniversal News Group, 16 Apr. 2018, www.nbcnews.com/tech/innovation/dockless-bikes-promise-future-transportation-litter-city-dallas-n866351.

Yang, Z., Chen, J., Hu, J., Shu, Y. and Cheng, P. (2015). Mobility Modeling and Data-Driven Closed-Loop Prediction in Bike-Sharing Systems. *IEEE Transactions on Intelligent Transportation Systems*, pp.165-178.