

Boston BlueBike Trip Optimizer

By: Lucas Jang, Aditya Saha, Justin Salim, Ayushi Shirke, and Felix Yang
Northeastern University, Boston, MA

Abstract

BlueBikes are a public bike-sharing system that is very convenient and accessible for people to use across Boston. However, many users have faced problems when they arrive at their destination, and find there are no available docks for them to end their ride. Not only do they have to travel to a further docking station, but also results in an increase in cost and travel time. In order to provide a better solution, we obtained previous BlueBike trip history data, as well as real-time dock availability, and used a combination of features to train a Support-Vector-Machines learning model to predict dock availability at a user's given destination station. We uploaded our processed BlueBike data into a Neo4j database, and ran cypher queries against it to determine expected travel times for users from a start to a given destination. As a result, we built vector features from our resulting Cypher queries in Neo4j, and we found our machine learning model performed at 99% accuracy, which has great potential to deliver accurate predictions for real-time destination dock availability.

Introduction

The city of Boston is ranked as one of the most walkable and bikeable cities in the United States [1], which makes it convenient for people to travel to work, school, museums, or even grocery stores. Since 2011, biking around Boston has become even more popular and accessible thanks to a public bike-sharing company called BlueBikes [2]. People can pay to unlock a BlueBike from a docking station and ride it around the city bike lanes to get to a destination docking station [2]. Taking a BlueBike is definitely a cheaper option than using a car-ride-sharing app or buying and maintaining a personal bicycle [2]. However, while Boston BlueBikes are certainly a popular and eco-friendly option, it comes with its own slew of issues.

Situations often arise where the rider unlocks a BlueBike and travels to their destination, but arrives to find that all the docks are full, so they cannot dock their bike and end their ride. Unfortunately, the only solution to this problem is to travel farther to another nearby BlueBike dock station to check for an available dock, adding travel time and potential additional costs for the rider. Since there is no current measure of the number of free docks available, this became the primary motivation for our BlueBike Trip Optimization project.

Our goal is to not only provide users with the shortest possible bike path from a given location to their destination location, but to also predict whether there will be available bikes at their starting station and open docks at their destination station. We will analyze previous trip history data from [BlueBikes-System-Data](#), along with current BlueBike station data from the same source, in order to make trip recommendations with the most relevant information.

We will predict available docks using a Support-Vector-Machines machine learning model, and train it based on the number of BlueBikes that leave or arrive at each docking station in hour-increments, the day of the week, and the time of day, as well as a comparison of the changes in BlueBikes at each docking station an hour before the current time, and a day before the current time. We considered including external training factors such as the weather on the day of the historical data, but we realized that extreme weather conditions do not considerably improve our predictor. In combination with this linear regression predictor, we will input this previous trip history data into a Neo4j graph database, which can easily be used to provide an optimal combination of start and end stations.

Based on our project goals, we predict that our trip optimizer will provide good trip recommendations for users, while accounting for dock availability at their end-destination station. We also predict that our chosen features will result in an accurate machine learning model, and we can incorporate its predictions along with our graph database for resulting trip recommendations.

Finally, this project will incorporate a user-interface that connects to an API program, which in turn will connect to our remote graph database, so this ease for user-interaction with our program, should result in a fast and reliable trip duration information, along with predicted dock availability as a better solution for users to avoid traveling to destinations without any available docks. The output is expected availability where 0 corresponds to “none”, 1 corresponds to “less than or equal to three”, and 2 is “more than three” available docks.

Methods

I. Data Acquisition and Pre-processing

We downloaded CSV files from [BlueBikes-System-Data](#), for trip history from March 2023, which is shown in Figure 1, and the current stations’ data, which is shown in Figure 2.

tripduration	starttime	stoptime	start station id	start station name	start station latitude	start station longitude	end station id	end station name	end station latitude	end station longitude	bikeid	usertype	postal code
169	2023-03-01 00:03:54.2260	2023-03-01 00:06:43.6980	12	Ruggles T Stop - Columbus Ave at Melnea Cass Blvd	42.336244	-71.087986	160	Wentworth Institute of Technology - Huntington...	42.337586	-71.096271	4727	Subscriber	02115
372	2023-03-01 00:05:42.5770	2023-03-01 00:11:54.8490	554	Forsyth St at Huntington Ave	42.339202	-71.090511	379	Tremont St at W. Dedham St	42.342549	-71.074214	7441	Subscriber	02118
593	2023-03-01 00:11:32.6930	2023-03-01 00:21:26.0270	554	Forsyth St at Huntington Ave	42.339202	-71.090511	558	St. Alphonsus St at Tremont St	42.333293	-71.101246	7345	Subscriber	02115

Figure 1. Shows a snippet of the March 2023 trip history data. Its attributes are “trip duration”, “start-time”, “stop-time”, “start station id”, “start station latitude”, “start station longitude”, “end station id”, “end station latitude”, “end station longitude”, “bikeid”, “usertype”, and “postal code”.

	Number	Name	Latitude	Longitude	District	Public	Total docks	Deployment Year
0	K32015	1200 Beacon St	42.344149	-71.114674	Brookline	Yes	15	2021.0
1	W32006	160 Arsenal	42.364664	-71.175694	Watertown	Yes	11	2021.0
2	A32019	175 N Harvard St	42.363796	-71.129164	Boston	Yes	17	2014.0
3	S32035	191 Beacon St	42.380323	-71.108786	Somerville	Yes	19	2018.0
4	C32094	2 Hummingbird Lane at Olmsted Green	42.288870	-71.095003	Boston	Yes	17	2020.0

Figure 2. Shows a snippet of the data obtained for the current stations data. Its attributes are “Number”, “Name”, “Latitude”, “Longitude”, “District”, “Public”, “Total docks”, and “Deployment Year”.

Originally, we were planning to use the Python Requests library to obtain JSON objects of real-time station-status data, which shows the number of bikes that are available. An example result from the request is shown in Figure 3.

```
{'stations': [{'num_docks_disabled': 0,
  'num_ebikes_available': 0,
  'num_docks_available': 0,
  'legacy_id': '3',
  'num_bikes_available': 13,
  'station_id': '3',
  'station_status': 'active',
  'num_bikes_disabled': 1,
  'is_returning': 1,
  'is_installed': 1,
  'eightd_has_available_keys': False,
  'is_renting': 1,
  'last_reported': 1681953970},
```

Figure 3. Shows the results from the real-time dictionary object that has a key called “num_bikes_available”.

To clean this data, we first realized that the station_id values do not correspond across both datasets, which meant that we would have to rely on the station names themselves to join these datasets. As a result, we would first need to compare the stations' names across the trip histories and current stations to determine if any do not match or are named with different conventions. Once the names corresponded across the datasets, we filtered the data to only include the city of Boston itself. For the trip histories, we needed to filter by the Python zipcodes library and only include those that have Boston zip codes, and for the current stations, we could simply filter by the “District” attribute to only include Boston.

II. Generating Features for Linear Regression Model

Originally, we trained a linear regression model, but we ended up finding that the Support-Vector-Machines model yielded the best predictions. Our approach began by converting the start-timestamp attribute into Python's datetime object, so that we could extract out the day of the week and the hour, which should contribute significantly to predicting whether docks will be available. From here, we divided the trip history data into one hour intervals for each BlueBike station, for each day of the week.

We used the JSON requests from the same source as the data, and its URL is linked here: https://gbfs.bluebikes.com/gbfs/es/station_status.json. From this, we could request real-time data concerning dock availability, which we could then incorporate in our feature vector.

As a result, we trained our Support-Vector-Machines model on a feature vector of each station's total dock capacity, the hour, the day of the week, a previous measurement of its dock availability in the previous hour and day, as well as the popularity of a station, which is found from the frequency of trip histories with each station as the start or destination.

III. Creating Nodes and Edges for Neo4j Graph Database

We stored the BlueBike trip history data in the following model:

Nodes: <i>Station {name: name of the station, lat: latitude, long: longitude , docks: total number of docks}</i>
Edges: <i>Trip {total: total number of trips between stations, average_duration: average duration of trips between stations, distance: distance between stations, anticipated_time: time between stations, w(weekday)_h(hour)_count: number of trips between stations at this time}</i>

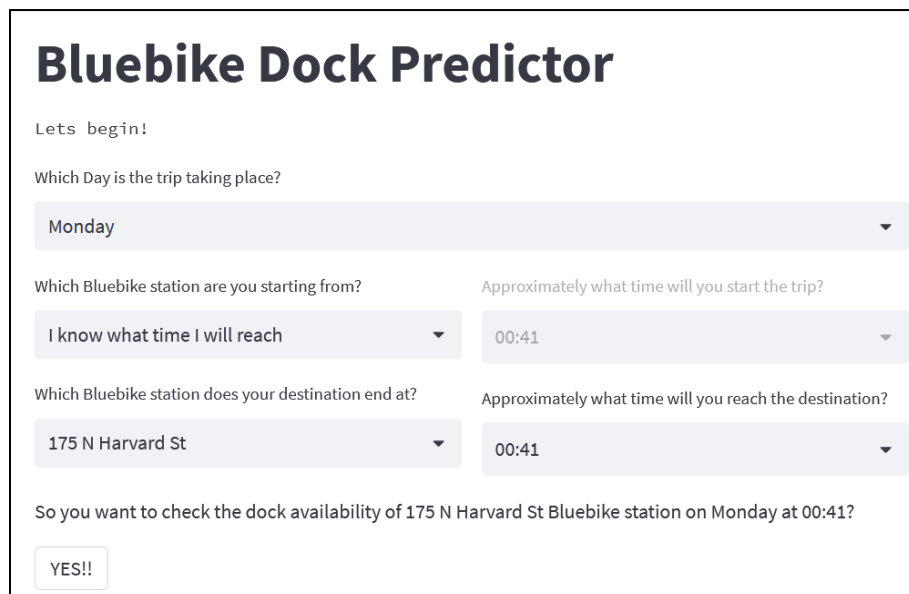
Figure 4. Shows the data model for the Neo4j graph database.

When exporting our data to Neo4j, stations were represented by nodes and trips between stations were represented by edges, as shown in Figure 4. Each node held the station name, its location in the form of longitude and latitude coordinates, and the total number of docks. Each edge held the total number of trips between the given stations, the average trip duration across all trips, the distance between the given stations, the anticipated travel time, and the total number of trips between the given stations given a weekday and hour. Because our edges were directional, signifying the direction of the trip, there was a maximum of two edges and a minimum of zero edges between two stations. Due to the limitations and filtering of our data, there were many stations without edges between them because there have been no recorded trips between those two stations.

Nonetheless, with such a model we can easily calculate various counts of information needed for our various models and analysis in order to pull necessary information to predict.

IV. Implementing the Front-End User-Interface

We created a website for users to input their trips, and it will show the predicted dock availability at the destination. The user simply needs to choose the day of the week, the starting station, the approximate starting time, the destination docking station, and what time the user wants to arrive at the destination. This page then connects to a remote neo4j database, where the trip information can be queried, and the Support-Vector-Machines model can predict the expected dock availability at the destination. An screenshot of the website with the drop-down menus for user input is shown in Figure 5.



The screenshot shows a web form titled "Bluebike Dock Predictor". Below the title is a greeting "Lets begin!". The form contains several input fields with drop-down menus:

- A label "Which Day is the trip taking place?" followed by a drop-down menu showing "Monday".
- A label "Which Bluebike station are you starting from?" followed by a drop-down menu showing "I know what time I will reach".
- A label "Approximately what time will you start the trip?" followed by a drop-down menu showing "00:41".
- A label "Which Bluebike station does your destination end at?" followed by a drop-down menu showing "175 N Harvard St".
- A label "Approximately what time will you reach the destination?" followed by a drop-down menu showing "00:41".

At the bottom of the form, there is a text prompt: "So you want to check the dock availability of 175 N Harvard St Bluebike station on Monday at 00:41?". Below this prompt is a button labeled "YES!!".

Figure 5. Shows the Front-End User-Interface with drop-down menus to enter trip information.

Analysis

I. Support Vector Machines Learning Model

To train and test our data, we allocated 75% of our data to training and 25% to testing. As there were negligible differences in predictive accuracy between the classification and regression algorithms we tried, we settled on using Support Vector Machines to train our model because its results were more digestible and easily interpreted compared to the linear regression model. As reflected in the confusion matrix in Figure 6, we resulted in a 99% prediction accuracy. While this may seem too high and overfitted, we believe that the main cause for such a high prediction accuracy was the way we generated our feature vector. When training our data, we calculated our output vector using a value that we calculated ourselves from our limited dataset which resulted in the features and output value being highly associated. In combination with the limited

scope of our dataset, being only one month's worth of data, as well as our filtering to remove any start or destination stations outside of Boston, it is not unexpected that our predictive accuracy was so high. Another important note is that some BlueBike employees are tasked to rebalance the bikes across docking stations both in Boston and outside of the city [2], so this replacement is also not documented within the trip history, which may also account for any discrepancies in the dock availability.

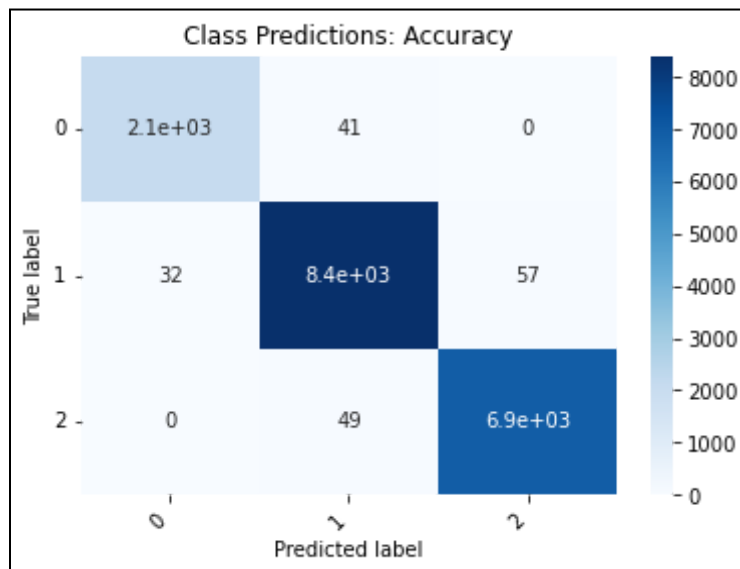


Figure 6. Shows the Confusion Matrix for our machine learning model, which correctly classified the vast majority of the labels.

II. Neo4j Graph Database

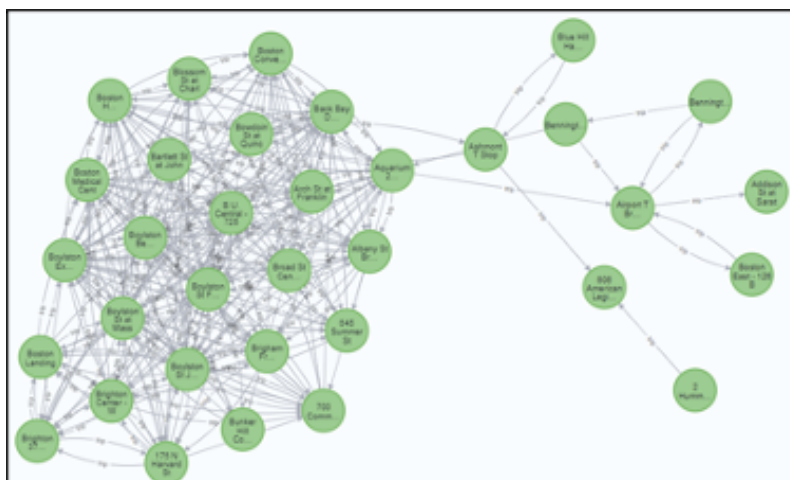


Figure 7. Shows the the visualized graph database containing one month's worth of trip and station data

Once the Neo4j database was loaded with nodes and edges, as shown in Figure 7, we were able to query it using Python's neo4j library to obtain stations, get information and properties about a trip, and even count the number of trips. This proved to be useful to determine the average duration of a trip from a starting BlueBike station to a destination BlueBike station. In combination with the Machine Learning model above, we were able to predict the destination station's dock availability and retrieve information about average trip duration for the user's given trip.

Conclusions

Our motivations for this project were based on a problem that BlueBike riders face when they arrive at their destination and they find that all the docks are full, so they cannot end their ride, and instead, have to travel further to another station. In order to provide a solution, our implementation not only retrieves data about the average duration of the user's BlueBike trip, but it also predicts the dock availability at the user's destination at an expected time. Through our Neo4j implementation, we were able to successfully query properties about the user's trip information such as the geo-spatial location of the, the popularity of the starting or destination BlueBike stations, or the anticipated time between the starting and ending stations. Additionally, we retrieved these attributes to create a feature vector to predict the expected dock availability using a Support-Vector-Machines machine learning model, which had an accuracy score of about 99%, so it performed really well. All in all, we are very happy with these results since they are much better than expected. In the future, we could always improve our implementation by adding more data, since that would further improve our machine learning model's performance. Also, it might be nice to incorporate the machine learning implementation within our Neo4j graph database itself to increase our efficiency. Overall, we are happy with these results, and we are excited to see its potential in solving many BlueBike users' current problem!

Author Contributions

All project members contributed to the design and implementation of this project. Everyone participated in presenting this project during the Science Fair. Justin, Ayushi, Lucas, and Felix contributed to the data preprocessing and machine learning parts of the project. Justin did a majority of the Neo4j importing and querying. Justin, Aditya, and Felix were mainly responsible for the frontend and interactable portions of the project. Ayushi and Lucas were the main writers for the project report although everyone contributed.

References

1. "Walker's Paradise." *Walk Score*,
<https://www.walkscore.com/score/berklee-college-of-music-boston-ma>.
2. Motivate International, Inc. "Bluebikes: Metro-Boston's Bikeshare Program." *Blue Bikes Boston*, <https://www.bluebikes.com/>.