# Predictive Modeling with Citi Bike Data

Tommy Geiger & Isaac Manly
University of Rochester

## ABSTRACT

This paper examines the data mining techniques and results of predictive modeling using publicly available Citi Bike data. The goal of this project was to build a model that could effectively predict the end location of a Citi Bike user in New York City from the moment they check out a bike. This project was created for the purposes of Fall 2019 CSC/DSC 240 at the University of Rochester.

## 1 INTRODUCTION

### 1.1 Motivation

In 2010, the economy made of short-term jobs, otherwise known as "The Gig Economy"[3] exploded and gave birth to many other smaller industries, like the microtransit industry. As a function of the gig economy's growth, the microtransit industry has also shown remarkable growth in the past decade, beginning with companies like Uber and Lyft. Those two companies specialized in convenient transportation via a car. Following the successes of these companies, other firms like Citi Bike innovated to capitalize on the industry's growth and began offering loanable, trackable scooters and bikes in cities across the US which allows users to check out a bike use the bike for a period of time.

This innovation created a vast dataset with 14 attributes and millions of objects. This dataset stretches back to 2013 and has been made publically available by Citi Bike. The objective is to train multiple models to predict as accurately as feasible the end location of users at the start of their trip.

The steps outlined by our method to predict the end location of CitiBike users are:

1. Data Preprocessing - This includes an exploratory stage, feature selection, feature engineering and data cleaning.
2. Creating an accuracy function - An accuracy function was crucial in measuring how close our model could predict a trip's geographic end point.
3. Model building and testing - This step includes the various algorithms and models created to achieve our objective.

After each of these steps has been examined, this paper will include a demonstration of how the model could be implemented in a practical manner, followed by a conclusion.

### 1.2 The Data

The dataset consists of millions of observations, Citi Bike trips in New York City, grouped by month from 2013 to 2019. There are 14 features in the original dataset:

Trip Duration (seconds)
Start Time
Stop Time
Start Station Name, ID
Start Station Latitude, Longitude
End Station Name, ID
End Station Latitude, Longitude
Bike ID
User Type (Day Pass or Subscriber)
Gender
Birth Year

The data has been processed by Citi Bike to remove trips that were taken by staff for service and inspection of the system, trips that were taken during the testing phase of a station, and trips that were less than one minute long.

## 2 PREPROCESSING

### 2.1 Data Exploration

The number of unique stations that a bike could be parked is 720. This implies that the prediction domain is 720 stations. Each station has an associated latitude and longitude value. The bikeid feature has a unique value per bike. Other features like gender and user type have 2-3 possible values. The most variable feature is birth year. Figure 1 is a distribution of birth years, indicating that the largest group of users is in the 20-35 age bracket.

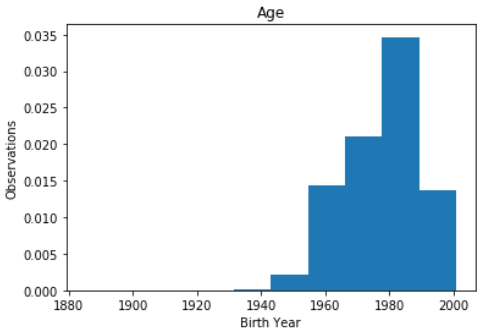


Fig 1.

### 2.2 Correlation

An important indicator of predictive power is the inter-correlation of features in the dataset. This information allows for educated feature selection during model testing. The correlation implemented is Spearman's correlation coefficient, because the distribution of each attribute was not necessarily gaussian. Table 1. in the Table section shows a table of the Spearman correlation between all of the attributes, including Start Cluster and End Cluster.

### 2.3 Data Cleaning

The data was fairly clean when it was acquired; only a few changes had to be made before it could be used. (1) Rows with missing birth year data are dropped, of which there were relatively few, (2) start and end times in datetime format (e.g. 5/1/2019 12:09:40 AM) converted to a float timestamp, and (3) string user type labels changed from "Subscriber" and "Customer" to integer labels 0 and 1, accordingly.

### 2.4 Feature Engineering

Due to the large prediction domain, model accuracy would be driven down as the features lack the predictive power to determine a specific street on which a trip would end. A solution to this problem is to create a new feature which greatly reduces the domain of possible predictions. K-means clustering was implemented to create 11 clusters across NYC and, in doing so, reduced the prediction domain to 11 possible end locations. Fig. 2 depicts the clustering of stations and since it was created using latitude and longitude points, features of NYC can be seen such as the Hudson river through the middle of the groupings.

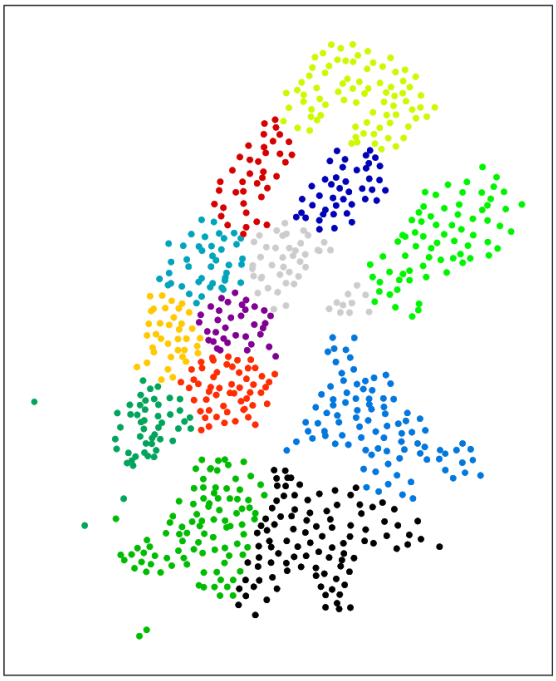

Fig 2.

This method is affected by an important tradeoff: the cluster vs. accuracy tradeoff. As the model abstracts from the street level and creates more clusters, information is lost.

### 2.5 Dimensionality Reduction

The first dimensionality reduction used in the model is the density clustering from section 2.4. This method reduced four relevant features to two features.

The second dimensionality reduction method used is feature selection due to low correlation found in table 1. BikeID a low overall correlation so it was not included in testing. Another reason to eliminate a feature is due to its multicollinearity with other features such as start station name's with other geographic attributes.

## 3 MEASURING PERFORMANCE

### 3.1 Haversine Distance

A standard measure of model performance is using a ROC analysis which charts sensitivity and specificity to illustrate the diagnostic capability of a particular test. This test of accuracy is not compatible with the objective to measure geographic distance as the model's predictions do not include any false positivity or negativity. Since binary measures lose a large amount of information, a more appropriate measure would be measuring the difference between the centroid of the predicted end cluster and the centroid of the true end cluster. Below, Figure 3. illustrates this accuracy measure. In a traditional model, prediction 1(P1) and prediction 2(P2) would both be equally incorrect as neither prediction is the true end cluster. P1 is notably closer than P2 and this difference must be accounted for.

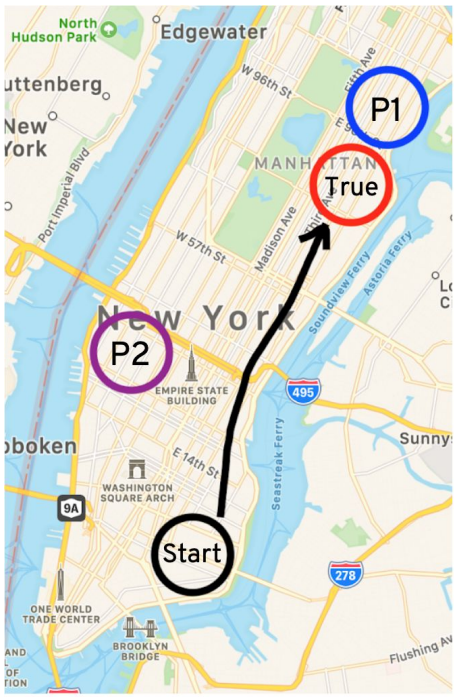


Fig. 3

To account for the spatial aspect of the accuracy measure, a distance metric must be implemented in this step. Euclidean and, topically, Manhattan distances are nearly suitable options but neither account for the curvature of the earth. The most appropriate metric would be the Haversine distance formula:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

The addition of this geometric functions support computing distance on that of a sphere. The implementation used in this model is inspired by Bruno Rochac's version.[2]

## 4 BUILDING MODELS

### 4.0 Data & Model Selection

Before training and testing various models, appropriate features must be selected to be used in the model. The following features were selected:

Start Time
Start Cluster
User Type
Birth Year
Gender

These features were selected because they are available in real time, at the moment a customer checks out a bike. Some features were left out due to redundancy--specifically, Start Station Latitude & Longitude, Start Station Name and Start Station ID, all of which are encapsulated in Start Cluster. Bike ID was also left out due to negligible correlation.

The best dataset to train and test on is the May 2019 data set due to sample size. Also, there is likely significant variation in the behaviors of bikers in Manhattan across different months and seasons, so in a real world application of this project, predictions could be improved by training the model on the data from the same month (perhaps even the same day) of the previous year. For example, there are likely very different bike patterns in June vs. February, and perhaps there are noticeable biking patterns in the days leading up to Christmas. For this reason, testing on a small time slice rather than a random sample of the entire year or several years would more accurately reflect behavior. The models use 100,000 random samples from the May 2019 data set, and an 80/20 train test split. Each test was implemented in Jupyter Notebooks using SciKit-Learn's machine learning package.

### 4.1 Naive Bayes

Implementing Naive Bayes, the average error that the model produced was 1.8950 miles. Naive Bayes likely outperformed KNN due to the algorithm's ability to handle discrete data and its robustness to noise. Real transportation data will include many outliers.

### 4.2 K-Nearest Neighbors

KNN's best average error was 2.0956 miles. KNN performed the worst of the three algorithms. This is likely due to the algorithm's sensitivity to noise.

### 4.3 Neural Network

The average error of the model's neural network test was 1.8863 miles. The neural network produced the lowest average error of the three tests.

### 4.4 Tradeoffs

A significant tradeoff encountered early on in the development of this project was choosing the appropriate number of clusters in our initial KNN preprocessing step. Fig. 4 shows a plot of the average error of the Naive Bayes classifier against the number of clusters.


Fig. 4

As the number of clusters increases, the model generally performs worse. However, with too few clusters there would be a significant amount of artificial success, as there would be many trips that start and end in the same cluster. For example, with 1 cluster the error rate would be 0, because all trips start and end in the same cluster. The optimal number of clusters would minimize the average error while still accurately and realistically representing the data. A method known as the Elbow method was employed in determining the approximate optimal number of clusters. The approximate optimal number for our data is 13 clusters, which lies on the "elbow" of the graph in Fig. 4. The Elbow method informs the model of an appropriate number of clusters to choose in order to minimize error without misrepresenting the data.

## 5 CONCLUSION AND IMPLICATIONS

This predictive model employed various algorithms with the purpose of predicting the end location of CitiBike users in NYC during May 2019 given relevant features. The most accurate algorithm used was a neural network with an average error of 1.8663 miles from the centroid of the predicted cluster. This result is more accurate than describing the end location by a given NYC burrough.

An important challenge faced during the building of this model was the accuracy method, as general methods were inappropriate to gauge the accuracy of the model.

In the future, microtransit companies like CitiBike could optimize their industry presence by employing similar techniques to the ones present in this paper. This model would also work in real time. The company would feed real time input to the model and approximate the end area of a user within moments. This is valuable to evaluate aggregate behavior and these results exemplify the power of machine learning algorithms given appropriate data. These results could also be used for targeted advertising, for example, such as suggesting attractions or routes to a user as soon as they check out a bike.

## 6 REFERENCES

1. CitiBike: *System Data*. 2013
2. Rochac, Bruno. 2011. *Calculate distance between latitude longitude pairs with Python*. Github.

3. Ian Hathaway, Mark Muro *Tracking the gig economy: new numbers*, 2016. Brookings.

## 7 TABLES

| | 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 | birth_year | gender | start_cluster | end_cluster |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| tripduration | 1.0000 | 0.0565 | 0.0582 | 0.0210 | 0.0104 | 0.0433 | -0.0086 | 0.0157 | 0.0015 | 0.0374 | -0.0099 | 0.0022 | 0.3255 | -0.0347 | -0.0450 | 0.0478 | 0.0396 |
| starttime | 0.0565 | 1.0000 | 0.9999 | 0.0060 | -0.0029 | -0.0166 | 0.0026 | 0.0149 | -0.0111 | -0.0151 | 0.0060 | -0.0098 | 0.0756 | 0.0035 | -0.0223 | 0.0094 | 0.0041 |
| stoptime | 0.0582 | 0.9999 | 1.0000 | 0.0061 | -0.0029 | -0.0166 | 0.0027 | 0.0150 | -0.0111 | -0.0151 | 0.0061 | -0.0099 | 0.0763 | 0.0034 | -0.0225 | 0.0096 | 0.0043 |
| start_station_id | 0.0210 | 0.0060 | 0.0061 | 1.0000 | -0.0729 | 0.2348 | 0.4065 | 0.2658 | -0.0678 | 0.1730 | 0.3100 | -0.0044 | 0.0260 | -0.0053 | -0.0001 | 0.2496 | 0.2031 |
| start_station_name | 0.0104 | -0.0029 | -0.0029 | -0.0729 | 1.0000 | -0.0377 | -0.2220 | -0.0686 | 0.1125 | -0.0424 | -0.1400 | 0.0052 | -0.0117 | -0.0058 | -0.0039 | 0.0174 | -0.0665 |
| start_station_latitude | 0.0433 | -0.0166 | -0.0166 | 0.2348 | -0.0377 | 1.0000 | 0.2950 | 0.1720 | -0.0246 | 0.8137 | 0.2183 | -0.0029 | 0.0140 | -0.0586 | -0.0340 | 0.2172 | 0.1664 |
| start_station_longitude | -0.0086 | 0.0026 | 0.0027 | 0.4065 | -0.2220 | 0.2950 | 1.0000 | 0.3070 | -0.1326 | 0.2352 | 0.6509 | -0.0014 | -0.0044 | 0.0393 | 0.0205 | 0.3205 | 0.2506 |
| end_station_id | 0.0157 | 0.0149 | 0.0150 | 0.2658 | -0.0686 | 0.1720 | 0.3070 | 1.0000 | -0.0670 | 0.2352 | 0.4103 | 0.0019 | 0.0257 | -0.0014 | 0.0011 | 0.2022 | 0.2488 |
| end_station_name | 0.0015 | -0.0111 | -0.0111 | -0.0678 | 0.1125 | -0.0246 | -0.1326 | -0.0670 | 1.0000 | -0.0263 | -0.2195 | 0.0010 | -0.0222 | -0.0084 | -0.0032 | -0.0604 | 0.0124 |
| end_station_latitude | 0.0374 | -0.0151 | -0.0151 | 0.1730 | -0.0424 | 0.8137 | 0.2191 | 0.2352 | -0.0263 | 1.0000 | 0.2994 | 0.0004 | 0.0098 | -0.0612 | -0.0339 | 0.1805 | 0.2047 |
| end_station_longitude | -0.0099 | 0.0060 | 0.0061 | 0.3100 | -0.1400 | 0.2183 | 0.6509 | 0.4103 | -0.2195 | 0.2994 | 1.0000 | 0.0039 | -0.0040 | 0.0378 | 0.0170 | 0.2518 | 0.3242 |
| bikeid | 0.0022 | -0.0098 | -0.0099 | -0.0044 | 0.0052 | -0.0029 | -0.0014 | 0.0019 | 0.0010 | 0.0004 | 0.0039 | 1.0000 | -0.0827 | -0.0133 | -0.0006 | -0.0016 | -0.0071 |
| usertype | 0.3255 | 0.0756 | 0.0763 | 0.0260 | -0.0117 | 0.0140 | -0.0044 | 0.0257 | -0.0222 | 0.0098 | -0.0040 | -0.0827 | 1.0000 | -0.0560 | -0.2966 | 0.0565 | 0.0483 |
| birth_year | -0.0347 | 0.0035 | 0.0034 | -0.0053 | -0.0058 | -0.0586 | 0.0393 | -0.0014 | -0.0084 | -0.0612 | 0.0378 | -0.0133 | -0.0560 | 1.0000 | 0.2035 | -0.0138 | -0.0162 |
| gender | -0.0450 | -0.0223 | -0.0225 | -0.0001 | -0.0039 | -0.0340 | 0.0205 | 0.0011 | -0.0032 | -0.0339 | 0.0170 | -0.0006 | -0.2966 | 0.2035 | 1.0000 | -0.0102 | -0.0123 |
| start_cluster | 0.0478 | 0.0094 | 0.0096 | 0.2496 | 0.0174 | 0.2172 | 0.3205 | 0.2022 | -0.0604 | 0.1805 | 0.2518 | -0.0016 | 0.0565 | -0.0138 | -0.0102 | 1.0000 | 0.3289 |
| end_cluster | 0.0396 | 0.0041 | 0.0043 | 0.2031 | -0.0665 | 0.1664 | 0.2506 | 0.2488 | 0.0124 | 0.2047 | 0.3242 | -0.0071 | 0.0483 | -0.0162 | -0.0123 | 0.3289 | 1.0000 |

Table 1.