CECS 632 Data Mining

Project 3: Data Analysis for Citi Bike System Data in New York City

Aliaa Elshamekh 11/26/19 Fall 2019

1 Abstract :

Bike-sharing systems are becoming increasingly prevalent in urban environments. They provide a low-cost, environmentally friendly transportation alternative for cities.
People initially tried and failed to introduce bike-sharing schemes in the 1960s due to the technical limitations such as tracking bikes and instant payment. However, nowadays, technological advances, such as bike tracking, solar powered sensors, mobile phones and wide-area internet and online aces, have helped transform bike-sharing from an aspiration to reality. The growth of bike sharing systems brings both health benefits and environmental benefits. In addition, the growing usage of bikes as means of transportation is associated with reduction in pollution and traffic congestion.
In this paper we analyze extensive operational data from bike-sharing systems in order to derive bike activity patterns. A common issue observed in bike-sharing systems is imbalances in the distribution of bikes. We use Data Mining to gain insight into the complex bike activity patterns at stations. Activity patterns reveal imbalances in the distribution of bikes and lead to a better understanding of the system structure. We try to solve the problem of maintain system balance during peak rush-hour usage as well as rebalancing overnight to prepare the system for rush-hour usage. We analyze system data to discover the best placement of bikes to facilitate usage. Moreover, try find some association rules between the location of the stations and the flow in and flow out and what time of the day and flow of the stations. The second task here is turn data into different clusters and explain what the cluster means. We will try spatial clustering, temporal clustering and the combination of both.

The dataset used in this project is from Citi Bike System Data | Citi Bike NYC

In this project, the main two tasks here is working with association rules and clustering. The first task is to analyze the data and answer some important questions about this provided dataset here such as:

- what is the most popular station in NY city?
- Rider performance by Gender and Age based on avg. trip distance (station to station), median speed (trip duration /distance traveled),
- How many stations are there in this dataset, and what is the average distance between them,
- What is the busiest bike in NYC in 2017? How many times was it used? How many minutes was it in. and so on.

Next, I will work on designing 3 mining tasks with their definitions of transactions and find some rules behind them.
For each task, we should

- try at least two discretization methods (divided by 10, divided by 20, …)
- Try at least two algorithms (Apriori, FP-growth, …) to find association rules.
- List the interesting rules.
- Compare the differences between them.

2 Data Analysis for Citi Bike System Data:

2.1 Problem Description

A bicycle-sharing system is a service in which users can rent/use bicycles available for shared use on a short-term basis for a price or free. Currently, there are over 500 bike-sharing programs around the world. Such systems usually aim to reduce congestion, noise, and air pollution by providing free/affordable access to bicycles for short-distance trips in an urban area as opposed to motorized vehicles. The number of users on any given day can vary greatly for such systems. The ability to predict the number of hourly users can allow the entities (businesses/governments) that oversee these systems to manage them in a more efficient and cost-effective manner.

Bike-sharing systems are a cost-effective way of promoting a sustainable lifestyle in urban areas. The number of bikes sharing systems has more than doubled since 2008. These systems generally consist of stations where users can take out or drop off bikes and may return a bike to a free dock at any station. New York City launched the largest bike-sharing system in North America, Citibike, in May 2013 with over 300 stations and 5000 bikes. The system has been a success with ridership approaching 40000 trips per day. With this success comes a set of management problems.

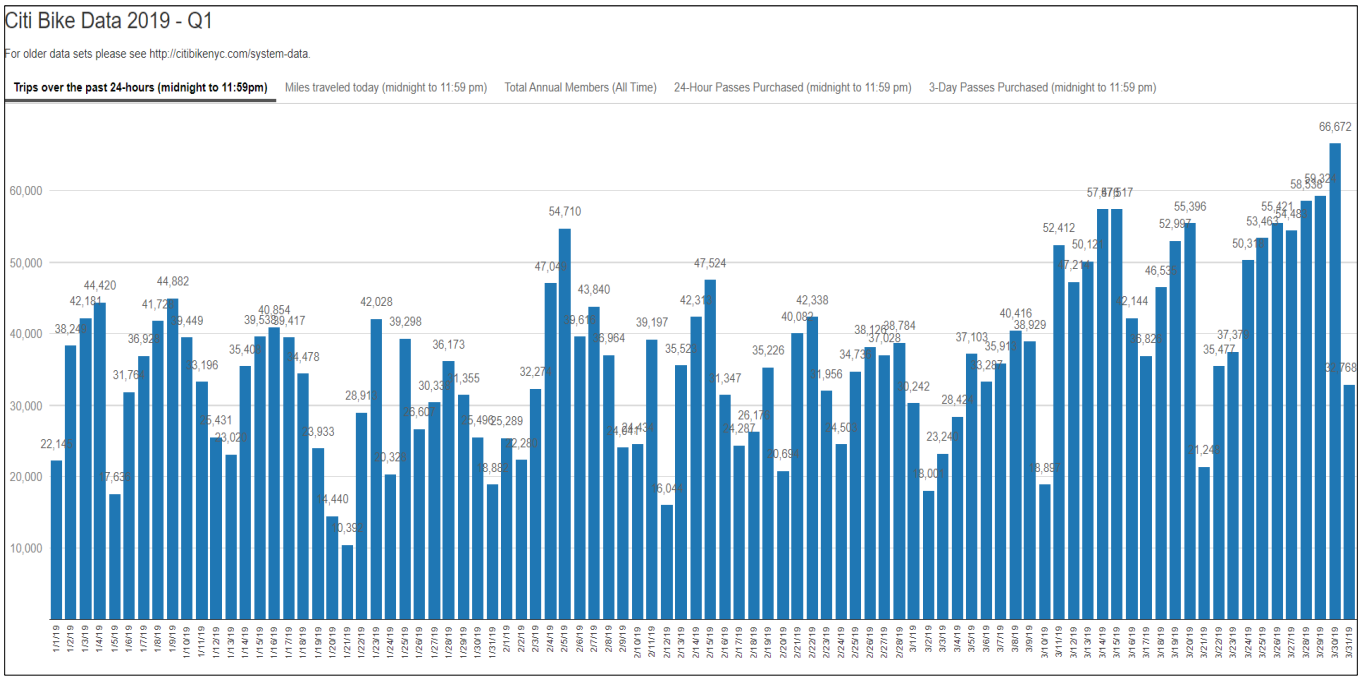


Figure 1 Citidata data provided on their website for trips on the first quarter of 2019

One major problem is the issue of system imbalance; bikes become clustered in certain geographic areas which leaves other areas devoid of bikes; for example, the traders wake up early in their East Village apartments, rapidly deplete the supply of bikes there, and then overwhelm the capacity for docks in the Wall Street area. This system imbalance necessitates moving bikes around the city. We begin by tackling the problem of how best to use system data to plan for usage. That is, we want to place bikes at stations to handle the surge in usage experienced during rush-hours. We approach the issue of inferring true demand for trips and use these amounts to better plan for system usage. From these computations we know both when and where bikes are needed and progress to answering the question: how do we get them there? We investigate the issue of rebalancing the system during rush-hour, developing novel

methods for optimizing rebalancing resources. During rush-hour, system usage is high, rendering large truck routes unreliable as the system state might shift dramatically before the route is completed. Traffic is also at its peak during rush hour, which greatly limits the ability of trucks to move easily through the city; this motivates the use of bike trailers instead of large trucks. Our goal is not to maintain system balance but to ensure that users are never too far from either a bike or a dock. To achieve this, we formulate a clustering problem that yields stations in the city for which rebalancing resources can be targeted.

Here are some of the hypothesis which I thought could influence the demand of bikes:

- Hourly trend: There must be high demand during office timings. Early morning and late evening can have different trend (cyclist) and low demand during 10:00 pm to 4:00 am.
- Daily Trend: Subscriber users demand more bike on weekdays as compared to weekend or holiday.
- Time: Total demand should have higher contribution of Subscriber user as compared to customer because Subscriber user base would increase over time.
- Traffic: It can be positively correlated with Bike demand. Higher traffic may force people to use bike as compared to other road transport medium like car, taxi etc

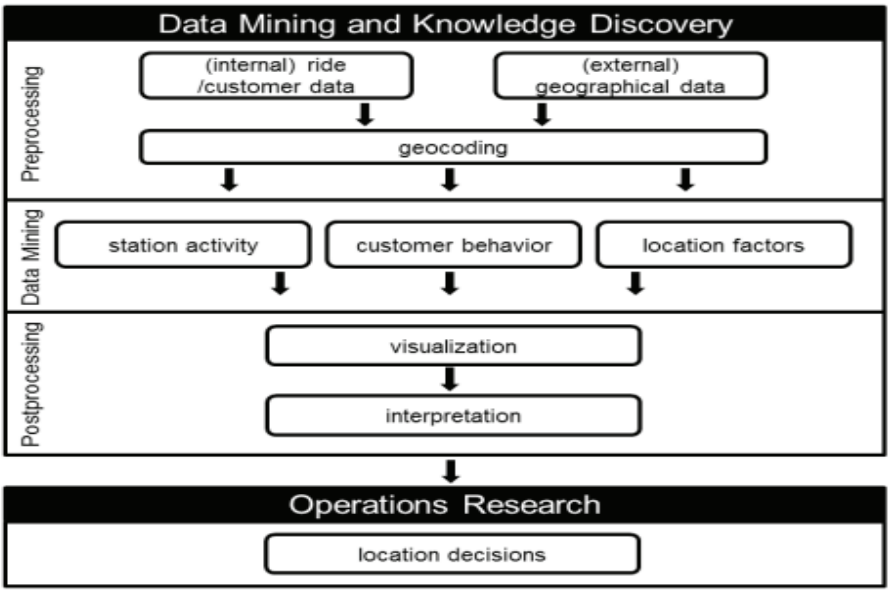


Figure 2 The outline of the methodology in this project

2.2 Data Description

In this project, The data set called Citibike trip data set which is provided by the Citi Bike official website. In this paper, I used the data on October 2019. This data set contained the following features:

- Trip Duration (seconds)
- Start Time and Date
- Stop Time and Date
- Start Station Name
- End Station Name
- Station ID
- Station Lat/Long
- Bike ID
- User Type (Customer = 24-hour pass or 3-day pass user; Subscriber = Annual Member)
- Gender (Zero=unknown; 1=male; 2=female)
- Year of Birth

This data has been processed to remove trips that are taken by staff as they service and inspect the system, and any trips that were below 60 seconds in length (potentially false starts or users trying to re-dock a bike to ensure it's secure). As for this data set, three features are mainly studied, the Trip Duration, Start Time and Date and Bike id. Finally, from the previous four Citibike features we calculated the explicit features used in our model: week number, weekend, weekday, bike demand between all pairs of stations, and mean/median trip duration.

In addition, the data type of each column is as following:

tripduration	int64	starttime	object
stoptime	object	start station id	int64
start station name	object	start station latitude	float64
start station longitude	float64	end station id	int64
end station name	object	end station latitude	float64
end station longitude	float64	bikeid	int64
usertype	object	birth year	int64
gender	int64		

I want to define the data types in this format

- Trip Duration - Int
- Start Time - DateTime
- Stop Time - DateTime
- Start Station ID - Categorical
- Start Station Name - Categorical
- User Type - Categorical
- Birth Year - Ordinal
- Gender – Categorical

In this report, I used the dataset for October 2019 that I am going to do all my analysis and observations on it. The Citi Bike was already quite well-organized and involved only fixing small errors, such as trip duration that lasted over 24 hours (presumably these trips were either data input errors or the bikes were stolen). The duration information is directly provided in the Citi Bike trip data set.

The dataset is provided in a csv file on the website provided above. It contains 2092564 record before any data cleaning processes and the data does not have any null value in any of each record at all.

	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
0	527	2019-10-01 00:00:05.618000	2019-10-01 00:08:52.943000	3746	6 Ave & Broome St	40.72431	-74.00473	223	W 13 St & 7 Ave	40.73782	-73.99995	41750
1	174	2019-10-01 00:00:15.875000	2019-10-01 00:03:10.168000	3301	Columbus Ave & W 95 St	40.79196	-73.96809	3283	W 89 St & Columbus Ave	40.78822	-73.97042	18264
2	759	2019-10-01 00:00:19.824000	2019-10-01 00:12:59.707000	161	LaGuardia Pl & W 3 St	40.72917	-73.99810	174	E 25 St & 1 Ave	40.73818	-73.97739	25525
3	615	2019-10-01 00:00:21.068000	2019-10-01 00:10:36.679000	254	W 11 St & 6 Ave	40.73532	-73.99800	477	W 41 St & 8 Ave	40.75641	-73.99003	30186
4	761	2019-10-01 00:00:26.380000	2019-10-01 00:13:08.313000	161	LaGuardia Pl & W 3 St	40.72917	-73.99810	174	E 25 St & 1 Ave	40.73818	-73.97739	25597

Figure 3 Sample from the data set used

2.3 Software tools:

For this project Python is used in both preprocessing and data mining process. Here is a list of packages that I used in the project with a brief description.

- **Pandas Library**

is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle many typical use cases in finance, statistics, social science, and many areas of engineering.

- **MatplotLib**

is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

- **Scikit-learn**

library in Python that provides many unsupervised and supervised learning algorithms. It’s built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib!
The functionality that scikit-learn provides include: Regression, Classification, Clustering, Model selection and Preprocessing.

- **Seaborn Library**

Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures

- **Geopy**

is a Python 2 and 3 client for several popular geocoding web services. geopy makes it easy for Python developers to locate the coordinates of addresses, cities, countries, and landmarks across the globe using third-party geocoders and other data sources.

- **Basemap Matplotlib toolkit**

The matplotlib basemap toolkit is a library for plotting 2D data on maps in Python. It is similar in functionality to the matlab mapping toolbox, the IDL mapping facilities, GrADS, or the Generic Mapping Tools. Basemap does not do any plotting on its own but provides the facilities to transform coordinates to one of 25 different map projections (using the PROJ.4 C library). Matplotlib is then used to plot contours, images, vectors, lines or points in the transformed coordinates. Shoreline, river and political boundary datasets (from Generic Mapping Tools) are provided, along with methods for plotting them. The GEOS library is used internally to clip the coastline and political boundary features to the desired map projection region.

- **Apriori**

Apyori is a simple implementation of Apriori algorithm with Python 2.7 and 3.3 - 3.5, provided as APIs and as commandline interfaces. It Consists of only one file and depends on no other libraries, which enable you to use it portably. In addition, it can use as APIs. Apriori supports a JSON output format.

- **pyfpgrowth**

FP-Growth (Frequent Pattern Growth) algorithm helps us to do Market Basket Analysis on transaction data. FP-Growth is preferred to Apriori because Apriori takes more execution time for repeated scanning of the transaction dataset to mine the frequent items. FP-Growth builds a compact- tree structure and uses the tree for frequent itemset mining and generating rules.

2.1 Data Preprocessing

2.1.1 Data Analysis

For this project, the first step is to identify the data, or the source of information, and from that we will be able to determine what information should be studying to retrieve important data from. Previously we mentioned that our target in this project to find some rules between the flow of some stations and time of the day and try to cluster the station locations to figure out some important common features for them. Now, we investigate the the dataset how it is look like and try to identify descriptions of some features and explore the feature to make brain storming about observation from the data. The aim is to answer some questions like:

- How many stations are there in this dataset, and what is the average distance between them?
- Top 5 stations with the most starts (showing # of starts)
- Most popular trips (based on start station and stop station)
- What are the top 3 frequent stations pairs (start station, end station) in weekdays, how about in weekends?
- Find the top 3 stations with highest average out-flow, and top 3 highest average in-flow
- Trip duration by user type
- What is the most popular station (highest average inflow + outflow)?

Let's start with some basic preprocessing steps like finding missing value. I find out that the dataset provided doesn't have any null value on any features which is an excellent condition.

tripduration	0
starttime	0
stoptime	0
start station id	0
start station name	0
start station latitude	0
start station longitude	0
end station id	0
end station name	0
end station latitude	0
end station longitude	0
bikeid	0
usertype	0
birth year	0
gender	0

Figure 4 There is missing values in the dataset

Next, to detect and eliminate strange values, we plot all the histograms on numeric values.

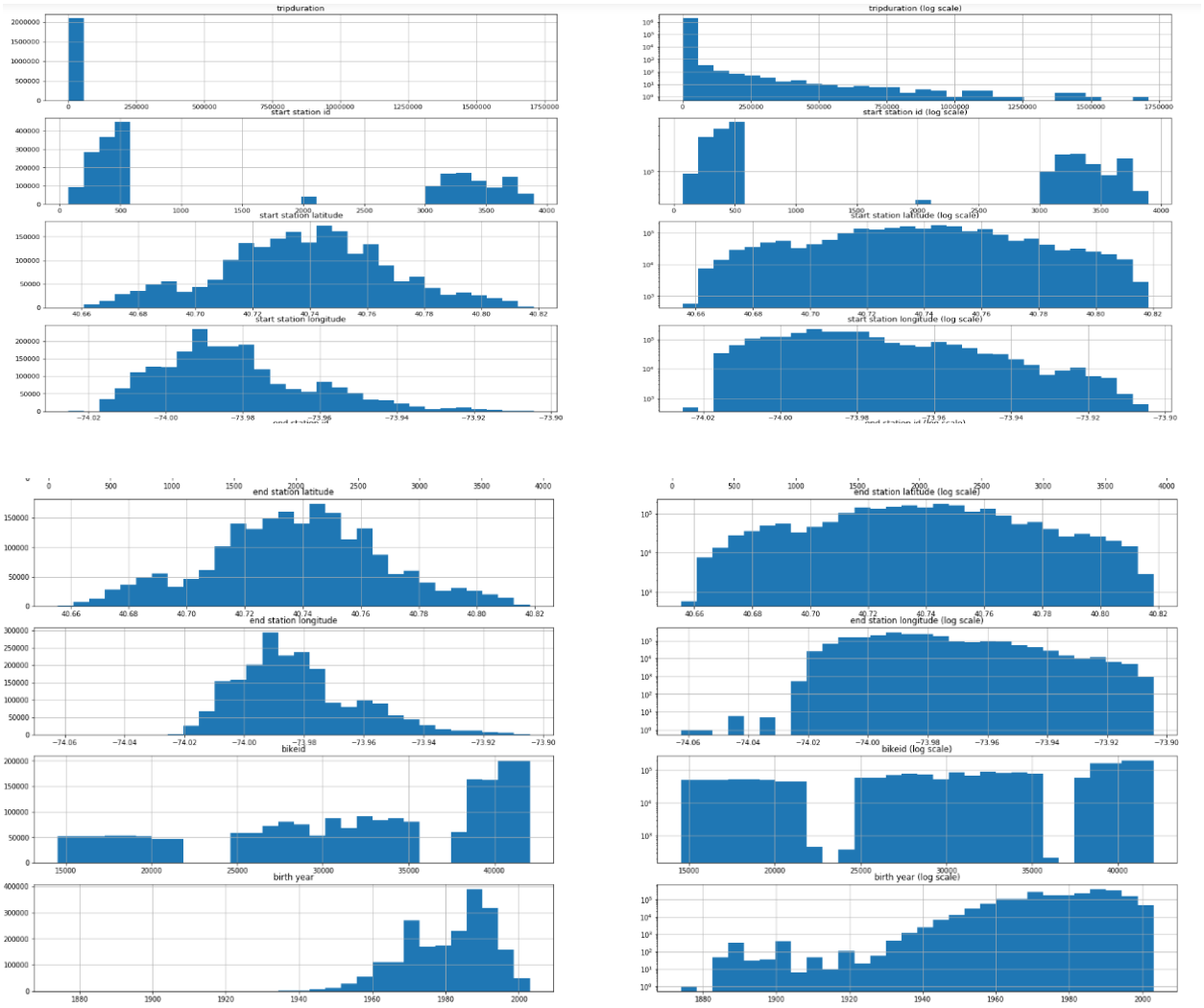


Figure 5 histogram for numeric values in the dataset

There are some values that are strange:

- The maximum value of trip duration is 2500000 sec(s), which is about 28 days. This is very uncommon for a person to rent a citibike for such a long time.
- There are trips that less than 1.5 minute (90 sec) which is illogical.
- There are stations that have coordinates that are very far away from other stations as seen above in the stations coordinates histogram.
- There are some samples (trips) that have start station = end station (circular trip)

According to the website of CitiBike, it provides information about bike rental and maximum trip duration for their customers. So, I can use this information provided to deal with the first problem by removing any records whose trip duration is over 20 days.

Moreover, to deal with trips which lasted less than 1.5 minute (90 seconds). If so, in the ideal world, we should not include this start, we may double count. If a bike is broken, a user will dock it again within a minute or two and pick-up another one. (This is my interpretation for that situation).
For the third observation, I check if all stations have one-to-one matching among their id, name, latitude, and longitude. **station id <—> station name, station id <—> station latitude, station id <—> station longitude.** I find out that all the values are one to one in the dataset.

So, I draw all stations on map to see if there is any strange location using the **Basemap** package of matplotlib to draw the map.

The station “3254”, “3182” and “3479” seems weird, they are in the water. So, I used google maps and station list on the official website provided by CitiBike website to confirm the existence of these stations and It seems that they are real stations that are on island near NYC called Governors Island.

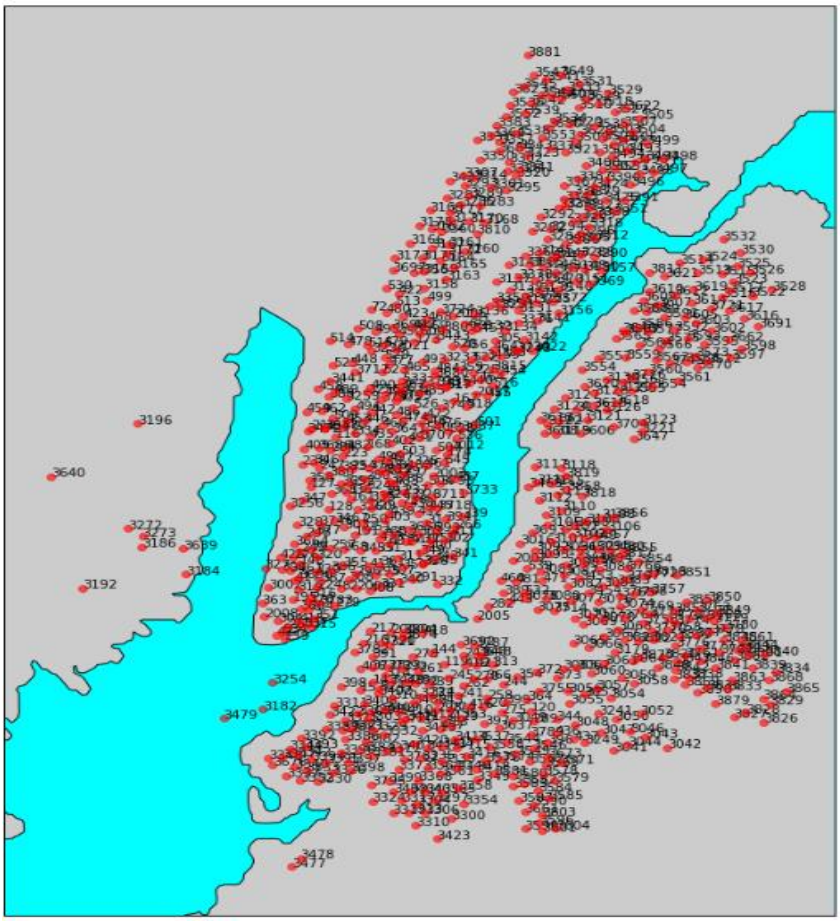


Figure 6 Map of the stations in the dataset

- **For Circular trips:**

Circular trips are trips which start and end at the same station. The distance for these trips will come out to 0, however, that is not the case. These points will skew the data and visuals. Will be removing them to account for this issue. For the model, this data is also irrelevant. Because if someone is going on a

circular trip, the only person who knows how long the trip is going to take is the rider him/herself, assuming they know that. So, it's safe to drop this data for the model.

- **Possibility of stolen bike:**

Any trip which lasts longer than 2 hours (7,200 seconds) probably indicates a stolen bike, an anomaly, or incorrect docking of the bike. As an avid Citibike user, I know firsthand that it doesn't make any sense for one to use a bike for more than one hour! However, I've added a one-hour cushion just in case. No rider would plan to go over the maximum 45 minutes allowed.

The following I will provide of the data analysis on this data by answering some of the questions mentioned above, so that I can make some hypotheses according to the answers.

- **How many stations are there in this dataset, and what is the average distance between them?**

There are 848 stations found in the dataset of October 2019. The average distance between two different station is 6160.6 (meters), which is calculated by the following equation

$$average\ distance = \frac{\sum_{i!=j} dist(S_i, S_j)}{\frac{N(N-1)}{2}}$$

- **Top 5 stations with the most starts (showing # of starts)**

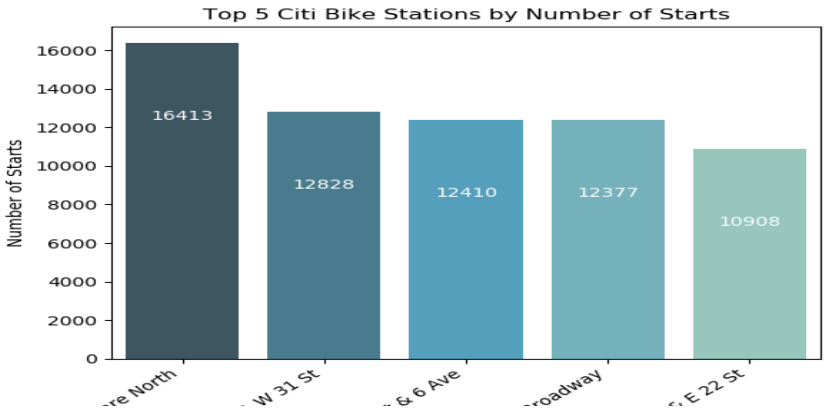


Figure 7 Top 5 stations in the dataset

- **Most popular trips (based on start station and stop station)**

In the following figure, I draw the most 10 popular trips in the dataset (trips that share the start and end station names). This will help in building an idea on how many bikes needs to be there in these stations. In addition, I will find the most three frequent pairs in weekdays, how about in weekends?

$$(S_i, S_j)\ is\ not\ (S_j, S_i)$$

Number of Trips	Trip
776	E 7 St & Avenue A to Cooper Square & Astor Pl
745	W 26 St & 8 Ave to 11 Ave & W 27 St
635	Vesey Pl & River Terrace to North Moore St & Greenwich St
622	W 21 St & 6 Ave to 9 Ave & W 22 St
606	Pershing Square North to E 24 St & Park Ave S
589	North Moore St & Greenwich St to Vesey Pl & River Terrace
588	11 Ave & W 27 St to W 26 St & 8 Ave
576	Central Park S & 6 Ave to Central Park S & 6 Ave
573	Central Park S & 6 Ave to 5 Ave & E 88 St
556	Grand Army Plaza & Central Park S to 5 Ave & E 88 St

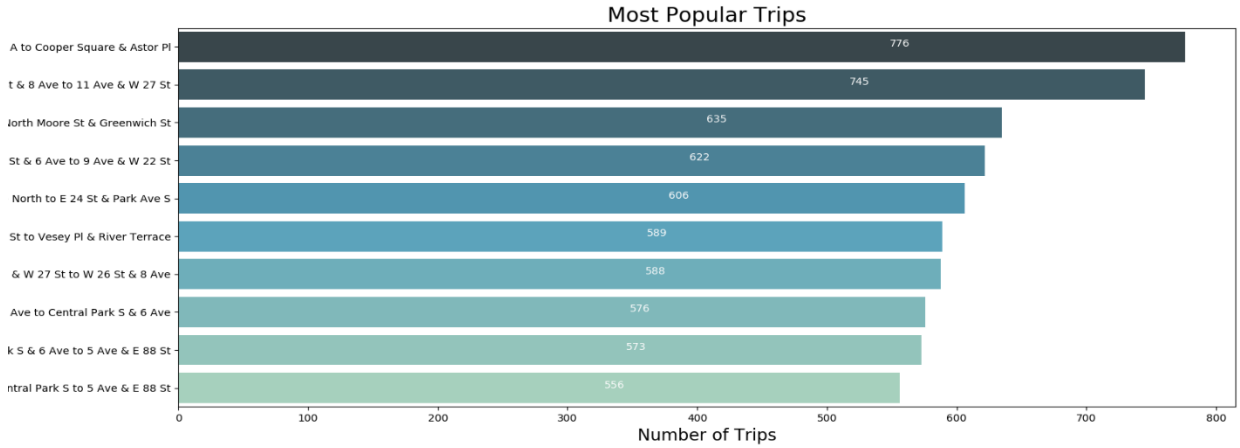


Figure 8 Most popular trips

Moreover, I can find that:
The top 3 frequent stations pairs in weekdays are: (494, 458), (432, 3263), and (327, 3664).
The top 3 frequent stations pairs in weekends are: (3182, 3182), (2006, 2006), and (3254, 3182).

- **Trip Duration by user type:**

This question is a bit unclear in terms of what to do with the anomalies or outliers in the data.
So it is an important part in data preprocessing to find out the outliers in the values of the features and figure out how to deal with them. I will look back into the website of citi Bikies to understand the logic of bike rentals and how the system handle trip duration and how much it cost and all the possible alternatives to be able to make decision about the threshold of the outliers.
According to Citi Bikes' website: The first 45 minutes of each ride is included for Annual Members, and the first 30 minutes of each ride is included for Day Pass users. If you want to keep a bike out for longer, it's only an extra $4 for each additional 15 minutes.
It's safe to assume, no one (or very few people) will be willing to rent a bike for more than 2 hours, especially a clunky citibike. If they did, it would cost them an additional $20 assuming they're annual subscribers. It would be more economical for them to buy a bike if they want that workout or use one of the tour bikes in central park if they want to tour and explore the city on a bike. There may be a better way to choose an optimal cutoff, however, time is key in a client project. Or just docking and getting another bike. The real cost of a bike is accrued ~24 hours.

I will try first to draw two graphs. One with outliers, one without. To figure out what is the reason behind that. Figures 9,10 show the trip duration with the usertype with and without outliers
As we can see in figure 9, there is too much noise for this to be useful. It'll be better to look at this without anomalies.
Figure 10 show the data without outliers' sill not useful, let's add a column with minutes for trip Duration. Boxplot with minutes is much more useful. There are still some outliers, however, it is informative.

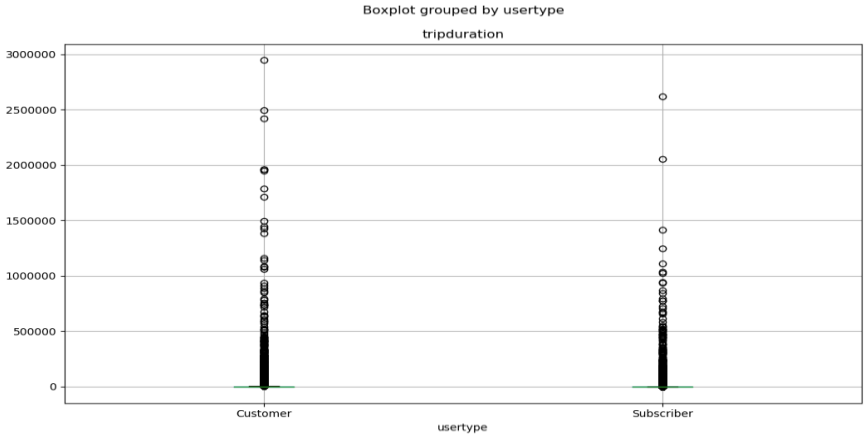


Figure 9 Boxplot of trip duration with anomalies

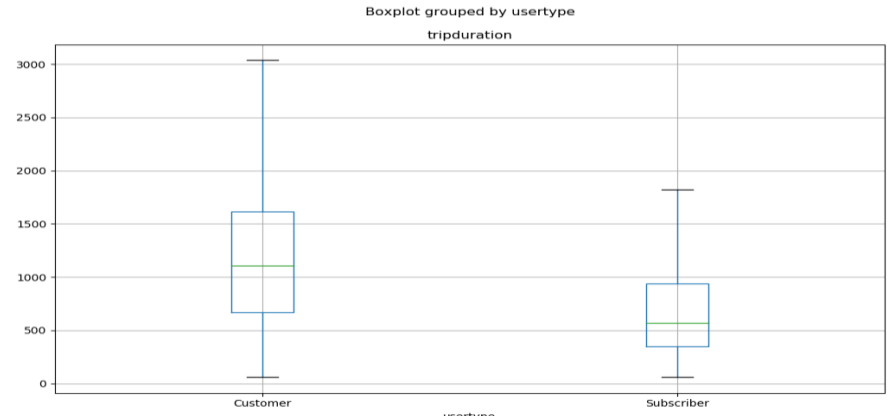


Figure 10 Box plot of trip duration without anomalies

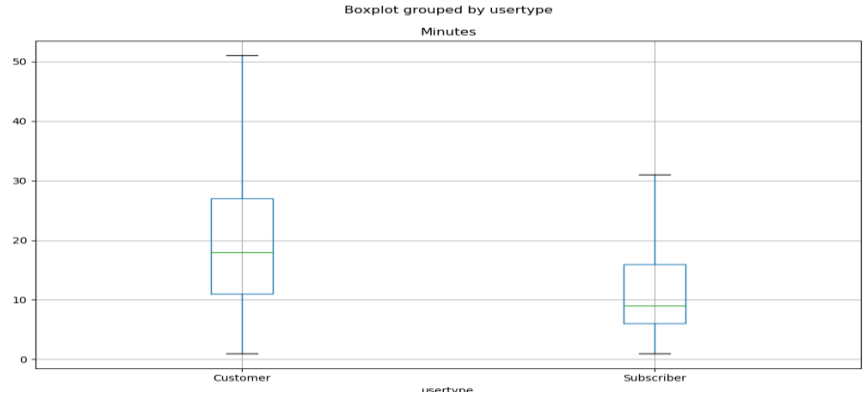


Figure 11 Boxplot of trip duration in minutes without anomalies

- **What is the most popular station (highest average inflow + outflow)?**

I find out that the station id is 519 which is the station called: Pershing Square North.

2.2 Features Creation

As my goal in this project to find out some rules between the time of the day how heavy is the flow and if there any ule between the location of the station and the flow on specific time

I need to calculate **in-flow** and **out-flow** for each stations every half hour. The result data set should contain station_id, time, in_flow_count, out_flow_count, this result was saved in csv format.

So, I define the flow as the number of trips move to/from the station within 30 minutes period which implies that the whole day can be split into 48 segments.
In order to accomplish this, I have changed the columns "starttime" and "stoptime" into datetime format. then split group and count trips in each segment. The data is structured on this format:

	station id	time	in_flow_count	out_flow_count
0	72	2019-10-01 10:00:00	7.0	3.0
1	72	2019-10-01 10:30:00	6.0	5.0
2	72	2019-10-01 11:00:00	3.0	1.0
3	72	2019-10-01 11:30:00	4.0	3.0
4	72	2019-10-01 12:00:00	2.0	5.0

Next, as I know now that on this dataset the station 519 is most popular station then the question here is what the flow in is and out in this station. The following graph shows the flow in and flow out of station 519 on Oct 2019.

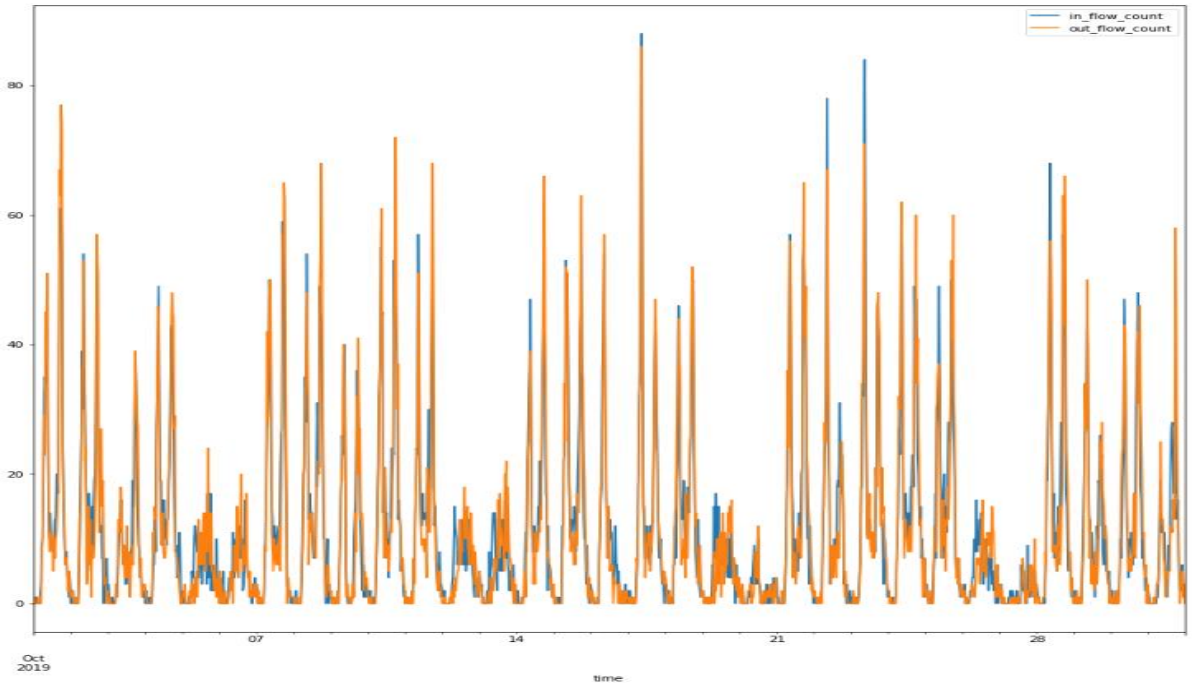


Figure 12 In/out flow for station 519

I need to make some observations on the flow of the stations over the whole month if there is any trend in that data.

I have portioned the month into 3 trimester **Early Oct, Mid Oct, Late Oct.** make this as a new feature in the data. Then plot every popular station on map where the size of the station point is calculated as

Size=2^{flow} , where flow=sum of flow count in this time period/1000.

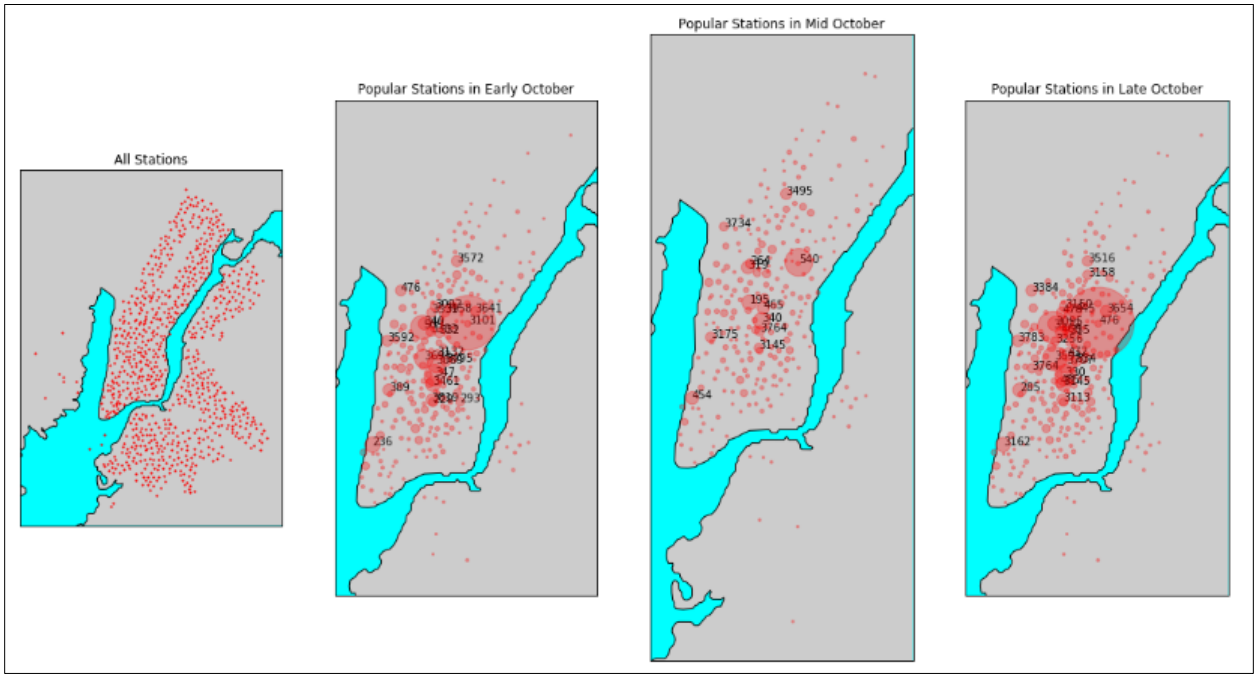


Figure 13 Popular stations over the month

Here are my observations:

The main flow of citibike concentrates in Manhattan according to comparing the map here with google map.
The popular stations in Brooklyn are nearly the same across the whole month.
Although the top popular regions (locations with large red circle) look similar, the top popular stations are not the same across the whole month. (Check station id are not the same.)
For popular regions, more and more people ride citibikes in Late October.

2.3 Data mining Process

First, before discussing the tasks that I am going to do, I will explain the algorithms and approaches I am going to use generally.

- **Discretization:**

. Generally, data discretization if we want to transform a continuous attribute to a categorical. In this project as mentioned earlier that I am going to use two types of approaches : equi-depth and equi size.

a) Equi-sized approach

It divides the range into N intervals of equal size . If A and B are the lowest and highest values of the attribute, the width of intervals with be:
$W = (B - A)/N$.

The most straight-forward but outliers may dominate presentation also, skewed data is not handled well. In addition, it loses a lot of information.

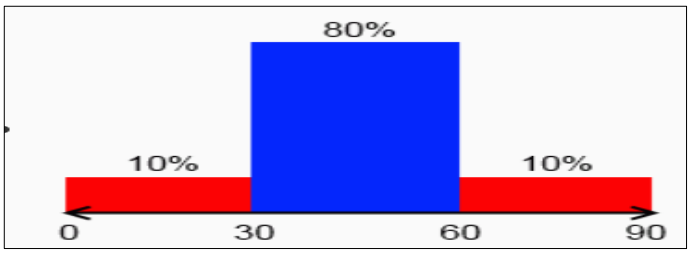


Figure 14 Equi-sized approach for discretization

pandas.cut(*x, bins, right=True, labels=None, retbins=False, precision=3, include_lowest=False, duplicates='raise'*)
Bin values into discrete intervals.
Use cut when you need to segment and sort data values into bins. This function is also useful for going from a continuous variable to a categorical variable. For example, cut could convert ages to groups of age ranges. Supports binning into an equal number of bins, or a pre-specified array of bins.

b) Equi-depth approach

It is also called frequency partitioning. It divides X into
N intervals, with each interval containing approximately
same number of samples not sensible to outliers distribution of values is considered (Good data scaling)

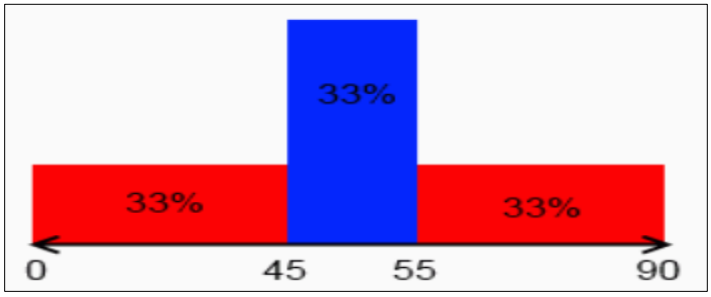


Figure 15 Equi-depth approach for discretization

For equi-depth algorithm, there is a function called ***pd.qcut*** in pandas library

pandas.qcut(*x, q, labels=None, retbins=False, precision=3, duplicates='raise'*)
Quantile-based discretization function. Discretize variable into equal-sized buckets based on rank or based on sample quantiles. For example, 1000 values for 10 quantiles would produce a Categorical object indicating quantile membership for each data point.

- **Association Rule Mining**

Association Rule Mining is one of the ways to find patterns in data. It finds:

- features (dimensions) which occur together
- features (dimensions) which are "correlated"

What does the value of one feature tell us about the value of another feature? For example, people who buy diapers are likely to buy baby powder. Or we can rephrase the statement by saying: If (people buy diaper), then (they buy baby powder). Note the if, then rule. This does not necessarily mean that if people buy baby powder, they buy diaper. In General, we can say that if condition A tends to B it does not necessarily mean that B tends to A. The measures of effectiveness of the rule are as Follows: Support, Confidence, Lift and Others like: Affinity, Leverage.
I am going to discuss Support and Confidence as these are the criteria I work with.

- Support means how much historical data supports your rule and Confidence means how confident are we that the rule holds.
- Support can be calculated as the fraction of rows containing both A and B or joint probability of A and B.
- Among rows containing A, Confidence is the fraction of rows containing B or conditional probability of B given A.

Algorithms that I am using in this project are Apriori algorithm, FP growth algorithm.

- **Apriori Algorithm – Frequent Pattern Algorithms**

Apriori algorithm was the first algorithm that was proposed for frequent itemset mining. It was later improved by R Agarwal and R Srikant and came to be known as Apriori. This algorithm uses two steps "join" and "prune" to reduce the search space. It is an iterative approach to discover the most frequent item sets. The main idea of this algorithm is: The probability that item I is not frequent is if:

P(I) < minimum support threshold, then I is not frequent.

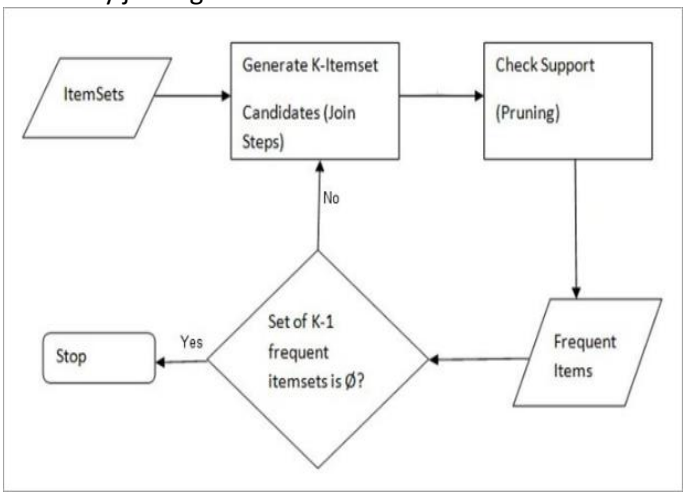
P (I+A) < minimum support threshold, then I+A is not frequent, where A also belongs to itemset.

If an itemset set has value less than minimum support then all of its supersets will also fall below min support, and thus can be ignored. This property is called the Antimonotone property.

The steps followed in the Apriori Algorithm of data mining are:

Join Step: This step generates (K+1) itemset from K-itemset by joining each item with itself.

Prune Step: This step scans the count of each item in the database. If the candidate item does not meet minimum support, then it is regarded as infrequent and thus it is removed. This step is performed to reduce the size of the candidate itemset.Apriori algorithm is a sequence of steps to be followed to find the most frequent itemset in the given database. This data mining technique follows the join and the prune steps iteratively until the most frequent itemset is achieved. A minimum support threshold is given in the problem or it is assumed by the user.



1) In the first iteration of the algorithm, each item is taken as a 1-itemsets candidate. The algorithm will count the occurrences of each item.

(2) Let there be some minimum support, min_sup. The set of 1 – itemset whose occurrence is satisfying the min sup are determined.
Only those candidates which count more than or equal to min_sup, are taken ahead for the next iteration and the others are pruned.

Figure 16 Apriori algorithm flow chart

3) Next, 2-itemset frequent items with min_sup is discovered. For this in the join step, the 2-itemset is generated by forming a group of 2 by combining items with itself.

4) The 2-itemset candidates are pruned using min-sup threshold value. Now the table will have 2 –item sets with min-sup only.

5) The next iteration will form 3 –item sets using join and prune step. This iteration will follow antimonotone property where the subsets of 3-itemsets, that is the 2 –itemset subsets of each group fall in min_sup. If all 2-itemset subsets are frequent then the superset will be frequent otherwise it is pruned.

6) Next step will follow making 4-itemset by joining 3-itemset with itself and pruning if its subset does not meet the min_sup criteria.

The algorithm is stopped when the most frequent itemset is achieved.

The equi-sized approach is implemented by a function in the panadas library called ***pd.cut***

- **Frequent Pattern Growth Algorithm**

This algorithm is an improvement to the Apriori method. A frequent pattern is generated without the need for candidate generation. FP growth algorithm represents the database in the form of a tree called a frequent pattern tree or FP tree.
This tree structure will maintain the association between the item sets. The database is fragmented using one frequent item. This fragmented part is called “pattern fragment”. The item sets of these fragmented patterns are analyzed. Thus, with this method, the search for frequent item sets is reduced comparatively.

FP Tree

Frequent Pattern Tree is a tree-like structure that is made with the initial item sets of the database. The purpose of the FP tree is to mine the most frequent pattern. Each node of the FP tree represents an item of the itemset.
The root node represents null while the lower nodes represent the item sets. The association of the nodes with the lower nodes that is the item sets with the other item sets are maintained while forming the tree.

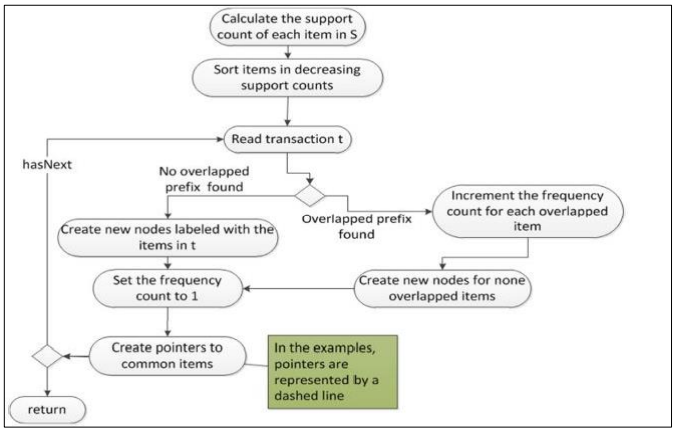


Figure 17 FP growth flowchart

The frequent pattern growth method lets us find the frequent pattern without candidate generation.
Steps followed to mine the frequent pattern using frequent pattern growth algorithm:

1) The first step is to scan the database to find the occurrences of the item sets in the database. This step is the same as the first step of Apriori. The count of 1-itemsets in the database is called support count or frequency of 1-itemset.

2) The second step is to construct the FP tree. For this, create the root of the tree. The root is represented by null.

3) The next step is to scan the database again and examine the transactions. Examine the first transaction and find out the itemset in it. The
itemset with the max count is taken at the top, the next itemset with lower count and so on. It means that the branch of the tree is constructed with transaction item sets in descending order of count.

4) The next transaction in the database is examined. The item sets are ordered in descending order of count. If any itemset of this transaction is already present in another branch (for example in the 1st transaction), then this transaction branch would share a common prefix to the root.

This means that the common itemset is linked to the new node of another itemset in this transaction.

5) Also, the count of the itemset is incremented as it occurs in the transactions. Both the common node and new node count is increased by 1 as they are created and linked according to transactions.

6) The next step is to mine the created FP Tree. For this, the lowest node is examined first along with the links of the lowest nodes. The lowest node represents the frequency pattern length 1. From this, traverse the path in the FP Tree. This path or paths are called a conditional pattern base.

Conditional pattern base is a sub-database consisting of prefix paths in the FP tree occurring with the lowest node (suffix).

7) Construct a Conditional FP Tree, which is formed by a count of item sets in the path. The item sets meeting the threshold support are considered in the Conditional FP Tree.

8) Frequent Patterns are generated from the Conditional FP Tree.

1. Mining association rules

As I mentioned in the beginning of the report, I would work through designing 3 mining tasks with their definitions of transactions and find some rules behind them. For each task, I am going to:

i. Try **two discretization methods** (divided by 10, divided by 20)
ii. Try **at two algorithms** (Apriori, FP-growth) to find association rules.
iii. List the interesting rules.
iv. **Compare** the differences between them.

My Focus here is on the most popular station i.e. station 519, since it is a very good representative of many of transactions I define.

- *Task 1: Find Rules between in-flow and out-flow of Station 519*

As I discussed earlier, I stored each station which each in and out flow in a separate file, this file is here my data source since my tasks depending on the flow of the station. Here, I want to find that if there is any association rules between in-flow counts and out-flow counts of station 519. Like for example:

{High in- flow Count} ➔ {High out-flow count}

i) Define Transactions

A transaction consists of in-flow for station id = 519, out-flow for station id = 519.
I do some observations on the transaction dataset. To make sure there is no missing values. See the minimum/maximum of flow counts values and See the distribution of flow counts values.

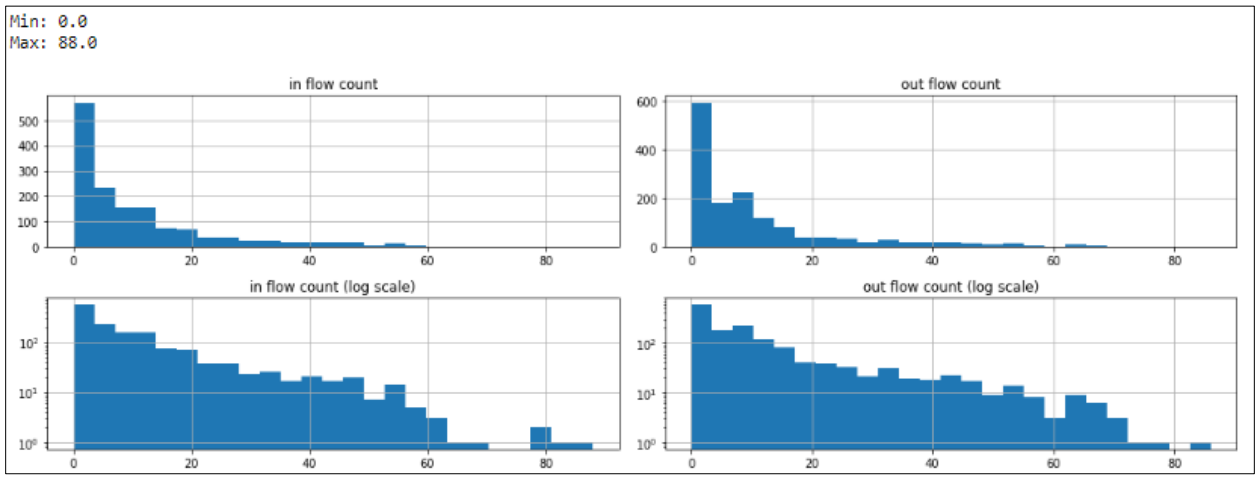


Figure 18 Distribution of flow counts

It is found that the flow counts values highly concentrate in the interval 0~20 for both in and out flow. Next, step is to split intervals of in/out flow counts using two discretization method I have mentioned earlier, the first discretization method ,I used is equi-sized approach where I have portioned the continuous domain into intervals with equal width, and the labels are defined as following :

- "Level -1" = size 0%~20%
- "Level -2" = size 20%~40%
- "Level -3" = size 40%~60%
- "Level -4" = size 60%~80%
- "Level -5" = size 80%~100%

Label	Interval	Number of Instance
In flow level -1	(-0.088,17.2)	1186
In flow level -2	(17.2,35.2)	192
In flow level -3	(35.2,52.8)	82
In flow level -4	(52.8,70.4)	24
In flow level -5	(70.4,88.0)	4

Table1 discretization of In- flow using equi sized approach

Label	Interval	Number of Instance
Out flow level -1	(-0.086,17.2)	1195
Out flow level -2	(17.2,34.4)	162
Out flow level -3	(34.4,51.6)	85
Out flow level -4	(51.6,68.8)	40
Out flow level -5	(68.8,86.0)	6

Table2 discretization of out- flow using equi sized approach

Second approach using equi-depth where I partition the data values into intervals with equal size along the ordering of the data. The labels are defined as follows:

- "zero" = quantile 0~20
- "extreme-low" = quantile 20~40
- "low" = quantile 40~60
- "medium" = quantile 60~80
- "high" = quantile 80~100

Label	Interval	Number of Instance
In flow zero	(-0.001,1.0)	386
In flow extreme-low	(1.0,4.0)	243
In flow low	(4.0,9.0)	274
In flow medium	(9.0,18.0)	304
In flow high	(18.0,88.0)	281

Table 3 discretization of In- flow using equi depth approach

Label	Interval	Number of Instance
In flow zero	(-0.001,1.0)	421
In flow extreme-low	(1.0,4.0)	230
In flow low	(4.0,8.0)	249
In flow medium	(8.0,17.0)	295
In flow high	(17.0,86.0)	293

Table 4 discretization of Out- flow using equi depth approach

ii) Association Rule Mining:

As I mentioned earlier, I will use **Apriori** and **FP-Growth** algorithms, we want to discover the relationship between in-flow counts and out-flow counts of station 519.
The support threshold and confidence threshold are determined by the quality and quantity of rules found. That is, number of rules found should not be too small or too large and the rules found should have confidence as high as possible. As my goal is to compare the difference between both algorithms, I use the support/confidence threshold to mine rules in transaction datasets with different discretization approach.

First, we mine the association rules of the transaction dataset that is discretized with equi-sized approach.

- support threshold = 0.1
- confidence threshold = 0.2

equi-sized (in. level-1) <==> (out. level-1)
==> 0.961; <== 0.954

```
Apriori
********
Rules:
------
(in.level-1) ==> (out.level-1)  confidence = 0.961
(out.level-1) ==> (in.level-1)  confidence = 0.954

FP-Growth
*********
Rules:
------
(in.level-1) ==> (out.level-1)  confidence = 0.961
(out.level-1) ==> (in.level-1)  confidence = 0.954
```

Figure 19 Results of both algorithms using equi sized approach discretization

Next, we mine the association rules of the transaction dataset that is discretized with equi-depth approach.

- support threshold = 0.1
- confidence threshold = 0.2

equi-depth (in. high) <==> (out. high)
==> 0.847; <== 0.812
equi-depth (in. zero) <==> (out. zero)
==> 0.793; <== 0.727

```
Apriori
********
Rules:
------
(in.high) ==> (out.high)  confidence = 0.847
(out.high) ==> (in.high)  confidence = 0.812
(in.zero) ==> (out.zero)  confidence = 0.793
(out.zero) ==> (in.zero)  confidence = 0.727

FP-Growth
*********
Rules:
------
(in.high) ==> (out.high)  confidence = 0.847
(out.high) ==> (in.high)  confidence = 0.812
(in.zero) ==> (out.zero)  confidence = 0.793
(out.zero) ==> (in.zero)  confidence = 0.727
```

Figure 20 Results of both algorithms using equi depth approach discretization

iii) Observations

The following table shows the rules found for both discretization methods, sorted by confidence.

Discretization Method	Rules Found	Confidence
Method (1). Equi-sized	(in. level-1) <==> (out. level-1)	==> 0.961; <== 0.954
Method (2). Equi-depth	(in.high) <==> (out. High)	==> 0.847; <== 0.812
Method (2). Equi-depth	(in.zero) <==> (out.zero)	==> 0.793; <== 0.727

Table 5 The rules found between in-flow and out-flow for station 519

Using equi-sized approach, we can find just two rules, which can be conclude as:

- In every 30 minutes, in-flow count of 0~17 indicates out-flow count of 0~17, and vice versa. (confidence > 95%)

Using equi-depth approach, we can find 4 rules, which can be conclude as

- In every 30 minutes, in-flow count of 18~88 indicates out-flow count of 17~86, and vice versa. (confidence > 81%)
- In every 30 minutes, in-flow count of 0 indicates out-flow count of 0, and vice versa. (confidence > 72%)

iv) Comparisons

The rules found by equi-depth approach cover 2 ranges of the flow (zero and high range). However, the top 1 rule found by 2 discretization approach are actually very different. The top 1 rule found with equi-sized approach says that low in-flow tends to co-occur with low out-flow, while and the top 1 rule found by equi-depth approach says that high in-flow tends to co-occur with high out-flow.

Top Rules	Discretization Method (1). equi-sized approach	Discretization Method (2). equi-depth approach
confidence > 90%	in-flow count of 0~17 <==> out-flow count of 0~17	
confidence > 80%		in-flow count of 18~88 <==> out-flow count of 17~86
confidence > 70%		in-flow count of 0 <==> out-flow count of 0
confidence > 60%		
confidence > 50%		

Table 6 Comparison between confidence level for both approaches

- Task 2: Find Rules between Time of a Day and Flows of station 519

The second task is to find that if there are any association rules between the time of a day and the flow counts of station 519. For example,

{Morning} ➔{High flow count}

I will follow the same steps or structure as task one.

i) Define Transactions

A transaction consists of : time of a day, flow count of station 519 (sum up in-flow count and out-flow count) I would like to make it clear here that time of a day means which time segment which we have defined earlier in the preprocessing section. The following table is just the first 5 rows from the data defined in the transaction of this task.

time	flow_count
00:00	2.00000
10:00	24.00000
10:30	18.00000
11:00	23.00000
11:30	22.00000

Figure 21 sample from transaction of task 2

Discretization method 1: I decided to split the day into 12 intervals (a bucket is of size = 2hours) , which I choose equi - depth approach to deal with the flow counts , define intervals for low medium and high flow count on station 519.

Label	Interval	Number of Instance
Flow count low	(0.001,5)	518
Flow count medium	(5,22)	510
Flow count high	(22,165)	460

Table 7 discretization method 1 (every 2 hour) using equi-depth

Discretization Method (2). It is method for discretization I have split a single day into 6 intervals which are defined as: Morning, Noon, Afternoon, Evening, or Night.

- 00:00 ~ 06:00: Night
- 06:00 ~ 11:00: Morning
- 11:00 ~ 13:00: Noon
- 13:00 ~ 16:00: Afternoon
- 16:00 ~ 22:00: Evening
- 22:00 ~ 24:00: Night

In order to implement this method of discretization I used the equi-depth approach again to deal whith the flow count like in method 1 into three intervals low medium and high.

Label	Interval	Number of Instance
Flow count low	(0.001,5)	518
Flow count medium	(5,22)	510
Flow count high	(22,165)	460

Table 8 discretization method 2 (part of the day) using equi-depth

ii) Association Rule Mining

Using Apriori and FP-Growth algorithms, we want to discover the relationship between which **time of a day and flow counts of station 519.**

I will apply here also the support threshold and confidence like in the first task.To compare the differences, I use the same support/confidence threshold to mine rules in transaction datasets with different discretization approach.

First, we mine the association rules of the transaction dataset that is discretized with Every 2 hours approach.

- support threshold = 0.05
- confidence threshold = 0.6

```
Apriori
********
Rules:
------
(00:00~02:00) ==> (low)    confidence = 0.927
(02:00~04:00) ==> (low)    confidence = 1.0
(04:00~06:00) ==> (low)    confidence = 0.823
(08:00~10:00) ==> (high)   confidence = 0.742
(10:00~12:00) ==> (medium)   confidence = 0.613
(12:00~14:00) ==> (medium)   confidence = 0.766
(16:00~18:00) ==> (high)   confidence = 0.758
(18:00~20:00) ==> (high)   confidence = 0.694
(20:00~22:00) ==> (medium)   confidence = 0.661
(22:00~24:00) ==> (low)    confidence = 0.613

FP-Growth
*********
Rules:
------
(02:00~04:00) ==> (low)    confidence = 1.0
(22:00~24:00) ==> (low)    confidence = 0.613
(04:00~06:00) ==> (low)    confidence = 0.823
(00:00~02:00) ==> (low)    confidence = 0.927
(20:00~22:00) ==> (medium)   confidence = 0.661
(12:00~14:00) ==> (medium)   confidence = 0.766
(18:00~20:00) ==> (high)   confidence = 0.694
(16:00~18:00) ==> (high)   confidence = 0.758
(08:00~10:00) ==> (high)   confidence = 0.742
(10:00~12:00) ==> (medium)   confidence = 0.613
```

Figure 22 Results of two association algorithms for every 2-hour

Next, we mine the association rules of the transaction dataset that is discretized with **Morning/Afternoon/Evening…** approach.

- support threshold = 0.05
- confidence threshold = 0.6

```
Apriori
********
Rules:
------
(Afternoon) ==> (medium)   confidence = 0.618
(Night) ==> (low)   confidence = 0.841
(low) ==> (Night)   confidence = 0.818
(Noon) ==> (medium)   confidence = 0.726

FP-Growth
*********
Rules:
------
(low) ==> (Night)   confidence = 0.818
(Afternoon) ==> (medium)   confidence = 0.618
(Noon) ==> (medium)   confidence = 0.726
```

Figure 23 Results of two association algorithms for each day interval

method (2). (Night) <==> (low)

==> 0.841; <== 0.818

iii) Observations

The following table shows the rules found for both discretization methods, sorted by confidence.

Discretization Method	Rules Found	Confidence
Method(1), Every 2 hours	(02:00~04:00) ==> (low)	1.0
Method(1), Every 2 hours	(00:00~02:00) ==> (low)	0.927
Method(2) Morning/noon/…	(Night) ==> (low)	0.841
Method (1), Every 2 hours	(04:00~06:00) ==> (low)	0.823
Method(2) Morning/noon/…	(low) ==> (Night)	0.818
Method (1), Every 2 hours	(12:00~14:00) ==> (medium)	0.766
Method (1), Every 2 hours	(16:00~18:00) ==> (high)	0. 758
Method (1), Every 2 hours	(08:00~10:00) ==> (high)	0.742
Method(2) Morning/Noon/…	(Noon) ==> (medium)	0.726
Method (1), Every 2 hours	(18:00~20:00) ==> (high)	0.694
Method (1), Every 2 hours	(20:00~22:00) ==> (medium)	0.661
Method(2) Morning/Noon/…	(Afternoon) ==> (medium)	0.618
Method (1), Every 2 hours	(22:00~24:00) ==> (low)	0.613
Method (1), Every 2 hours	(10:00~12:00) ==> (medium)	0.613

Table 8 The rules found between time of the day and flow

Using "Every 2 hours" approach, we can find 10 rules, which can be conclude as

- At 00 ~ 06 o'clock, the flow counts tend to be lower than 5 in every half hour. (confidence > 82%)
- At 12 ~ 14 o'clock, the flow counts tend to be in the interval (5, 22] in every half hour. (confidence > 75%)
- At 16 ~ 20 o'clock, the flow counts tend to be more than 22 in every half hour (confidence > 68%), especially at 16 ~ 18 o'clock (confidence > 75%).
- At 08 ~ 10 o'clock, the flow counts tend to be in the interval more than 22 in every half hour. (confidence > 67%)

Using "Morning/Afternoon/..." approach, we can find 3 rules, which can be conclude as

At night (22 ~ 24 and 00 ~ 06 o'clock), the flow counts tend to be lower than 5 in every half hour, and vice versa. (confidence >80%)
At noon (11 ~ 13 o'clock), the flow counts tend to be in the interval (5, 22] in every half hour. (confidence > 60%)

iv) Comparisons

With "Every 2 hours" approach, we can find more rules. I think it is because with this approach, we've got more buckets and it happens to match the scenario that flow counts change rapidly at the scale of hour.

Top Rules	Discretization Method (1). "Every 2 hours" approach	Discretization Method (2). "Morning/Afternoon/..." approach
confidence > 90%	(02:00~04:00) ==> less than 5 flow counts per 30 minutes (00:00~02:00) ==> less than 5 flow counts per 30 minutes	
confidence > 80%	(04:00~06:00) ==> less than 5 flow counts per 30 minutes	(Night) ==> less than 5 flow counts per 30 minutes
confidence > 70%	(16:00~18:00) ==> more than 5 flow counts per 30 minutes	less than 5 flow counts per 30 minutes ==> (Night)
confidence > 60%	(12:00~14:00) ==> 5 ~ 5 flow counts per 30 minutes (18:00~20:00) ==> more than 5 flow counts per 30 minutes (08:00~10:00) ==> more than 5 flow counts per 30 minutes	(Noon) ==> 5 ~ 5 flow counts per 30 minutes

Table 10 Comparison between two approaches

If we check the rules carefully, you would find that the rules found by Method (1) almost cover the rules found by Method (2).

- Task 3: Task 3: Find Rules between Station Locations and Their Daily Flows:

The last task is to find if there are any association rules between the location of a station and its daily flow counts. For example,

{70 <longitude≤75.5} ➔ {High daily flow count}

i) Define a transaction

A transaction in this task as I used to consist of

- Station latitude
- Station longitude
- Daily flow counts (this is different on the feature flow count that I used in the previous task)

This feature can be defined as the sum up every day in-flow and out-flow for each station.

	station latitude	station longitude	flow_count
0	40.6554	-74.010628	14.0
1	40.6554	-74.010628	12.0
2	40.6554	-74.010628	8.0
3	40.6554	-74.010628	18.0
4	40.6554	-74.010628	17.0

Figure 24 sample from the transaction of the task 3

I have used this new variable or quantity here to avoid generating only rules for low flow count.

Discretization Method (1).

Using equi-sized approach to partition the continuous domain of the station location into intervals with equal length. (this is applied for both longitude and latitude features)

Interval	Number of Samples
(40.688, 40.721]	7657
(40.721, 40.753]	5983
(40.753, 40.786]	5518
(40.655, 40.688]	4309
(40.786, 40.818]	2821

Table 10 station latitude in method1 discretization using equi-sized approach

Interval	Number of Samples
(-73.999, -73.968]	10075
(-73.968, -73.936]	9145
(-73.968, -73.936]	3596
(-74.031, -73.999]	3224
(-74.063, -74.031]	248

Table 11 station longitude after discretization method 1 using equi-sized approach

To deal with flow counts, I have used the equi-depth approach Intervals for flow counts are as the following: Extreme-low, low, medium, high and extreme high

Label	**Interval**	**Number of Instance**
daily flow count extreme-low	(-0.001,30.0)	5440
daily flow count low	(30.0, 65.0]	5179
daily flow count medium	(65.0, 128.0]	5172
daily flow count high	(128.0, 274.0]	5241
daily flow count extreme high	(274.0, 1609.0]	5256

Table 11 Daily flow count method1 discretization using equi-depth

Discretization Method (2). equi-depth approach. I partition the data values into intervals with equal size along the ordering of the data.

Interval	Number of Samples
(40.654, 40.692]	5270
(40.692, 40.715]	5270
((40.715, 40.743]	5239
(40.743, 40.769]	5270
(40.769, 40.818]	5239

Table 12 station latitude in method (2) discretization using equi-depth approach

Interval	Number of Samples
(-74.063, -73.993]	5270
(-73.993, -73.979]	5270
(-73.979, -73.959]	5239
(-73.959, -73.943]	5270
(-73.943, -73.904]	5239

Table 13 station longitude after discretization method 2 using equi-depth approach

Label	Interval	Number of Instance
daily flow count extreme-low	(-0.001,30.0)	5440
daily flow count low	(30.0, 65.0]	5179
daily flow count medium	(65.0, 128.0]	5172
daily flow count high	(128.0, 274.0]	5241
daily flow count extreme high	(274.0, 1609.0]	5256

Table 14 Daily flow count discretization method 2 using qui-depth

ii) Association Rule Mining:

Using Apriori and FP-Growth algorithms, we want to discover the relationship between station locations and their daily flow counts. To compare the differences, I use the same support/confidence threshold to mine rules in transaction datasets with different discretization approach. First, we mine the association rules of the transaction dataset that is discretized with equi-sized approach.

- support threshold = 0.08
- confidence threshold = 0.4

Rules using apriori :

(extreme-high) ==> (latitude = (40.721, 40.753]) confidence = 0.585
(latitude = (40.721, 40.753]) ==> (extreme-high) confidence = 0.514
(extreme-high) ==> (longitude = (-73.999, -73.968]) confidence = 0.693
(extreme-low) ==> (latitude = (40.688, 40.721]) confidence = 0.431
(extreme-low) ==> (longitude = (-73.936, -73.904]) confidence = 0.412
(longitude = (-73.936, -73.904]) ==> (extreme-low) confidence = 0.623
(high) ==> (longitude = (-73.999, -73.968]) confidence = 0.536
(low) ==> (longitude = (-73.968, -73.936]) confidence = 0.515
(medium) ==> (longitude = (-73.968, -73.936]) confidence = 0.484
(medium) ==> (longitude = (-73.999, -73.968]) confidence = 0.408(
(extreme-high) ==> (longitude = (-73.999, -73.968]) ,latitude = (40.721, 40.753]) confidence = 0.63

Rules using FP-Growth :

(low) ==> (longitude = (-73.968, -73.936]) confidence = 0.515
(longitude = (-73.936, -73.904]) ==> (extreme-low) confidence = 0.623
(extreme-low) ==> (longitude = (-73.936, -73.904]) confidence = 0.412
(high) ==> (longitude = (-73.999, -73.968]) confidence = 0.536
(latitude = (40.721, 40.753]) ==> (extreme-high) confidence = 0.514
(extreme-high==> (latitude = (, longitude = (-73.999, -73.968]) ,40.721, 40.753]) confidence = 0.63

Next, we mine the association rules of the transaction dataset that is discretized with equi-depth approach.

- support threshold = 0.08
- confidence threshold = 0.4

Rules using apriori:

(extreme-high) ==> (latitude = (40.715, 40.743]) confidence = 0.461
(latitude = (40.715, 40.743]) ==> (extreme-high) confidence = 0.462
(extreme-high) ==> (longitude = (-73.993, -73.979]) confidence = 0.415
(longitude = (-73.993, -73.979]) ==> (extreme-high) confidence = 0.414
(extreme-low) ==> (longitude = (-73.943, -73.904]) confidence = 0.553
(longitude = (-73.943, -73.904]) ==> (extreme-low) confidence = 0.574

Rules using FP-Growth:

(longitude = (-73.993, -73.979]) ==> (extreme-high) confidence = 0.414
(latitude = (40.715, 40.743]) ==> (extreme-high) confidence = 0.462
(extreme-high) ==> (longitude = (-73.993, -73.979]) confidence = 0.415
(extreme-low) ==> (longitude = (-73.943, -73.904]) confidence = 0.553
(longitude = (-73.943, -73.904]) ==> (extreme-low) confidence = 0.574

iii) Observations

The following table below shows the rules found, sorted by their confidence.

Discretization Method	Rules Found	Confidence
Method (1). equi-sized	(extreme-high) ==> (longitude = (-73.999, -73.968]) ,latitude = (40.721, 40.753])	0.63
Method (1). equi-sized	(extreme-high) ==> (latitude = (40.721, 40.753])	0.585
Method (2). equi-depth	(longitude = (-73.943, -73.904]) <==> (extreme-low)	==> 0.574; <== 0.553
Method (1). equi-sized	(high) ==> (longitude = (-73.999, -73.968])	0.536
Method (1). equi-sized	(low) ==> (longitude = (-73.968, -73.936])	0.515
Method (1). equi-sized	(medium) ==> (longitude = (-73.968, -73.936])	0.484
Method (2). equi-depth	(extreme-high) <==> (latitude = (40.715, 40.743])	==> 0.461; <== 0.462
Method (1). equi-sized	(extreme-low) ==> (latitude = (40.688, 40.721)	0.431
Method (2). equi-depth	(extreme-high) <==> (longitude = (-73.993, -73.979])	==> 0.414; <== 0.415
Method (1). equi-sized	(extreme-low) ==> (longitude = (-73.936, -73.904])	0.412

Using equi-sized approach, we can find 7 rules, which can be conclude as

- Extreme-high daily flow count (more than 274 per day) tend to occur at the area of latitude 40.721~ 40.753 and longitude -74.012 ~ -73.985. (confidence >63%)
- High daily flow counts (128.0~274.0per day) tend to occur at the area of latitude longitude = (-73.999, -73.968]) . (confidence >50%)
- Medium and low daily flow counts (30~ 128 per day) tend to occur at the area of longitude -- 73.968~-73.936. (confidence ≃50%)

Using equi-depth approach, we can find 6 rules, which can be conclude as

- Stations at latitude (40.715, 40.743] tend to have extreme-high daily flow count (more than 274 per day) (confidence ≃45%), and at longitude = (-73.993, -73.979] . tend to have extreme-high daily flow count (more than 274 per day) (confidence ≃40%).

- Stations at (longitude = (-73.968, -73.936]) tend to have extreme-low daily flow count (more than 274 per day). (confidence >55%)

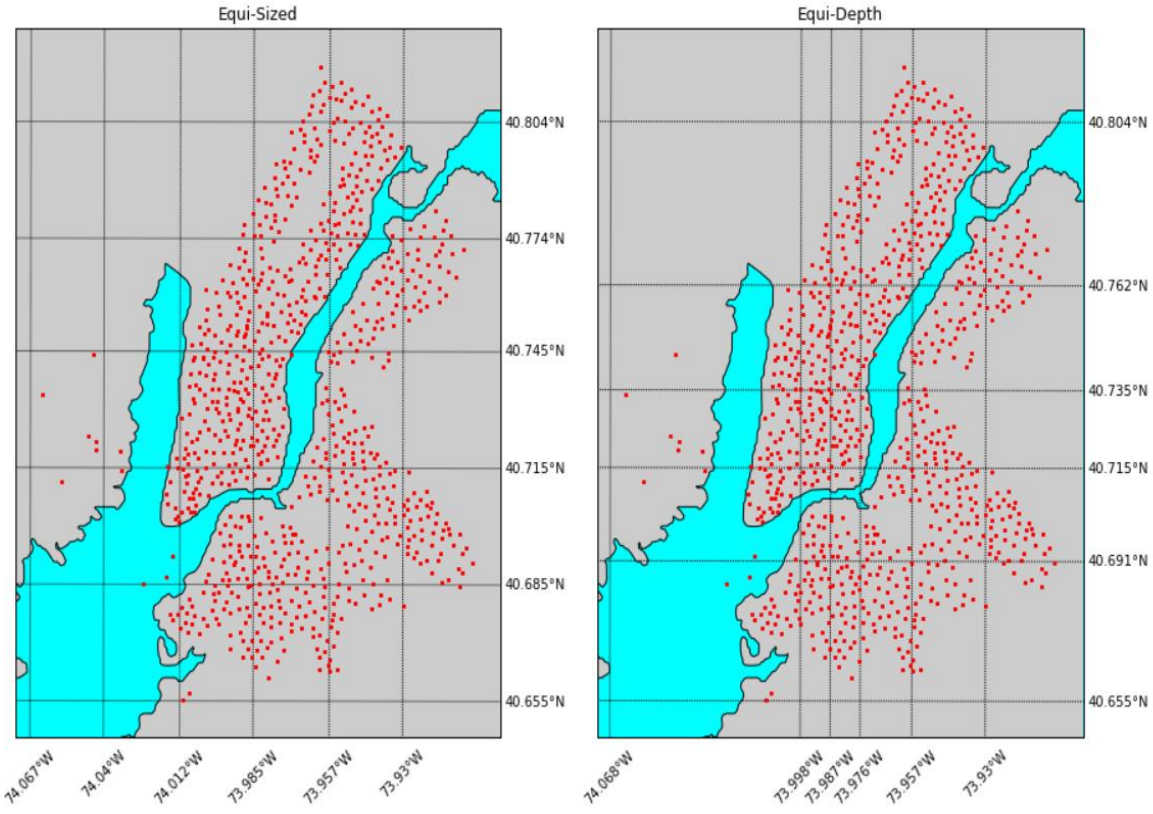


Figure 25 Discretization results on NYC map

With equi-sized approach, we can find more rules and higher confidence. I think it is because using equi-depth approach would make all split areas have the same density of stations, while a station is actually built according to the population or say, estimated flow count, of that area. As the result, the number of rules mined with equi-depth approach and their confidence would likely to be low.

iv) Comparisons

Top Rules	Discretization Method (1). equi-sized approach	Discretization Method (2). equi-depth approach
confidence > 60%	more than 274 flow counts per day ==> (longitude = (-74.012, -73.985])	
confidence > 55%		Less than 30 flow count per day <==> (longitude = (-73.943, -73.904])
confidence > 50%	Between 128 and 274 daily flow counts ==> longitude = (-73.999, -73.968] More than 30 and less than 65 daily flow counts per day ==> (longitude = (-73.968, -73.936])	(latitude = (40.736, 40.762]) ==> more than 286 flow counts per day
confidence > 45%		(latitude = (40.715, 40.743<==> more than 274 flow counts per day
confidence > 40%	Less than 30 flow counts per day ==> (latitude = (40.688, 40.721)	more than 274 flow counts per day <==>longitude = (-73.993, -73.979]

4. Conclusion and future work:

In this project, the data collection, data cleaning and the feature extraction takes most of the time, which I believe is true to most of the data analysis projects. Only after those preparation can you make sure that this project is feasible and meaningful. The data pre-processing procedure is full of unexpected errors and cause a lot of problem to me. The data provided by Citi bikes was a clean data that don't contains any missing values.

For some future work which in fact was supposed to be done in parallel with mining association rules is to turn data into different clusters and explain the meaning of them which includes temporal and spatial clustering techniques.

Understanding demand for bike services is important for logistics matters and for environmental reasons. Today, more and more cities try to implement sharing bike systems and have to deal with increasing demand for this service.
Here, we tried to solve part of this problem: by showing ho many users will use the service, we can better schedule demand and define the best dates for maintenance of bikes that are not in use.

What we found today is that some hours are busier than others. Some seasons demand more or less bikes on the streets and some other variables seem to have none or little effect over the problem.
Is it possible to get better results?