Project Report

Predicting Station Rentals in Bike Sharing Systems

April 2025

Research Task in Statistics, Winter Semester 2024/25
Faculty of Transport and Traffic Sciences, TU Dresden

Bhavay Singhal
Student — Msc Transportation Economics
Matriculation number: 5126210

Supervisor: Ms Jing Zou, MSc
Supervising Professor: Prof Dr Ostap Okhrin

Table of Contents

Introduction

Background

Bike sharing systems play a significant role in enhancing urban mobility and facilitating transportation. By providing an alternative to cars and personal mode of transportation, bike-sharing helps alleviate traffic congestion, especially for short to medium distances (Hamilton & Wichman, 2017). This leads to smoother traffic flow and reduced travel times. Even where there is an extensive public transit network in place, there are still gaps in the network that might make certain areas of regional space inaccessible (Griffin & Sener, 2017). By functioning as an instrument for first and last mile connectivity, bike-sharing systems can work as linkage and reduce these gaps to improve accessibility, making it easier for people to reach destinations that would otherwise not be possible (Kabra et al, 2018).
In addition to urban mobility, bike share can help prevent harm to the environment and help combat climate change. Since bikes are a zero-emission mode of transport, they contribute towards cleaner air and reducing carbon footprint (Hallisey, 2022). Research on New York's Citi Bike system has showed the promising potential of bike sharing systems in this context. In 2024, it was estimated that this programme had reduced carbon emissions by 1,800,000 pounds or 816 metric tons per month (Adamjee, 2024). Similarly, it was found that bike sharing in Shanghai saved 8,358 tonnes of petrol and reduced carbon dioxide emissions and NOX emissions by 25,240 tonnes and by 64 tonnes respectively, in 2016 (Zhang & Mi, 2018). In an effort to lower carbon emissions, it’s essential that at least some motorized trips are replaced with a greener alternative such as shared-bikes. According to a report by Nextbike-UK (2017), studies on implementing bike share found that motorized trips fell by 7% (in Lyon, France) up to more than 50% (in Mexico City, Mexico and Lisbon, Portugal), after such systems were implemented.

While the benefits of implementing a bike sharing system are visible when compared to some other modalities, there are considerable improvement areas that can facilitate successful adoption and allow the mentioned benefits to be better realized. Considering climate change, a life cycle assessment of a bike sharing system in 2024 has revealed that the manufacturing phase of shared bicycles produces the largest amount of carbon emissions, followed closely by fleet management and maintenance (Chen et al, 2024). Focusing on the second component in particular, the assessment highlights the importance of efficiently redistributing bikes across the network continuously, for accurate station occupancy (number of bikes to be positioned at docking stations) and maximizing usage for each bike by accurately foreseeing future spatial demand per station (Salah et al, 2021). This would allow the produced emissions throughout the entire life-cycle to be optimized, leading to a ‘net’ reduction (Chen et al, 2024).
A more immediate problem in a bike sharing system arises due to unreliable bike access. These systems often experience imbalances because user demand varies depending on location and time of day (Kim, 2023). For example, many people might ride bikes downhill to a certain area in the morning, leaving those stations full and others empty. This often leads to problems such as users being unable to find a bike when they need one or they can’t return one because the station is already full. It is commonly found that in such systems, a lack of effective re-balancing leads to high failure rates during peak times, limiting availability and frustrating users (Chiariotti et al, 2018). Ultimately, such tendencies result in lost demand caused by bike shortages at stations, lowering both system utilization and user satisfaction (Affonso, 2021). Ensuring the right number of bikes are available where and when they're needed would reduce the likelihood of users

encountering empty stations when they need a bike, and minimize instances of full stations, allowing users to return bikes without inconvenience. However, for quick and reliable bike access, which requires frequent re-balancing during peak times, reliable short term demand prediction at high spatial resolution is crucial (Liang et al, 2022). Short term demand prediction can also be helpful in planning maintenance schedules more effectively, ensuring bikes from a given station are serviced during periods not pertaining to peak demand (Chandwani et al, 2024). However, this requires estimation of demand patterns that is more granular than daily demand, with a location component to accommodate for spatial variation in usage pattern.

New York City, in May 2013, launched the largest bike share program in the United States known as Citi Bike. Strong connection to public transportation hubs and covering targeted areas with high station density have been critical factors in high ridership for this programme in the city (Kaufman, 2018). In particular, the lower half of Manhattan (Lower and Midtown Manhattan) and pockets of northwest Brooklyn (including Downtown Brooklyn) were targeted as initial deployment areas. The selection of these operating areas was strategic for the following reasons:

- **High population density:** These areas have a high concentration of residents, workers, and visitors, creating a large potential ridership base.
- **Commercial and residential mix:** The combination of business districts (such as the Financial district) and residential neighborhoods generates demand for bike-sharing for commuting, errands, and leisure.
- **Transit connectivity:** These areas have good subway and bus access, allowing Citi Bike to integrate with existing public transportation and provide a "last mile" solution.

These reasons ensure that ridership demand exhibit high usage patterns and is consistent, especially during peak times. This has been the reason for many machine learning based demand prediction studies to have traditionally concentrated on high-traffic regions belonging to the initial launch areas (Choi et al, 2020, O’Mahony, 2015). Meanwhile, the programme’s biggest challenges have been to expand further from the initially launched areas as well as lost demand due to rebalancing issues (Kaufman, 2018). Even within the initial operational areas, Citi Bike has struggled to maintain its bikes. Since 2017, the service has gradually expanded into other parts of the city, including outer boroughs and upper Manhattan. However, compared to the areas in which the service was initially launched, these areas are likely to have discrete demand patterns because of lower population densities, smaller or fewer business districts and less comprehensive transit access in some areas. These reasons can result in fewer trips generated in a given time period, irregular demand or lacking of network effects seen in high-density regions *i.e. proximity to other stations boosts usage* (Ranaiefar et al, 2016), making it more challenging for machine learning models to generalize demand patterns for future prediction (Uddin et al, 2023). Hence, a machine learning study that models usage patterns with focus on these less understood areas, possessing unique characteristics, ought to be conducted.

Project’s Goal and Scope

Compared to simpler machine learning methods, deep learning techniques have become very popular, achieving very high accuracy by modeling complex patterns. The ability to automatically extract features from raw data eliminates the need for raw feature engineering (Dataiku, 2025). This is particularly valuable for dealing with unstructured data such as text or images. However, when dealing with tabular or structured data, traditional methods such as tree-based ensembles still hold importance for prediction tasks and have some advantages over deep learning models:

- **Less Data Requirement:** Tree-based ensembles generally perform well with smaller to medium-sized datasets, whereas deep learning models typically require very large datasets to train effectively and avoid overfitting (Grinsztajn et al., 2022).
- **Interpretability:** Tree-based models are inherently more interpretable than deep learning models. Techniques like feature importance and partial dependence plots can easily explain how each feature influences the model's predictions. Deep learning models are often considered "black boxes" due to their complexity, making it difficult to understand the decision-making process (Rane et al., 2024).
- **Training Speed:** Tree-based ensembles usually train much faster than deep learning models. Deep learning models, especially complex architectures, can take significant time and computational resources to train (Grinsztajn et al., 2022).
- **Computational Resources:** Tree-based models require less computational power and memory compared to deep learning models, making them suitable for deployment on resource-constrained devices (Grinsztajn et al., 2022).
- **Less Hyperparameter Tuning:** Tree-based models often have fewer hyperparameters to tune compared to deep learning models. This simplifies the model development process and reduces the risk of overfitting (Roßbach, 2023).
- **Robustness to Outliers:** Tree-based models are generally more robust to outliers in the data than deep learning models. The tree structure inherently isolates outliers, reducing their impact on the overall model (Obermiller, 2025). On the other hand, neural networks are much more susceptible to the influence of the outliers (Kennedy, 2025).
- **Simpler Implementation:** Implementing tree-based models is generally simpler and requires less specialized knowledge compared to deep learning models, making them more accessible (Roßbach, 2023).

In practice, ensembles of decision trees (such as random forests and gradient boosting machines) and multilayered neural networks are most applicable across a vast majority of datasets and tend to give similar results for many kinds of structured data, in terms of accuracy in prediction tasks (Howard & Gugger, 2020). However, keeping the above mentioned advantages of tree-based approaches in mind, this project aims to perform a comparative study between different tree-based ensembles for short-to-medium-term demand prediction on bike-station rentals, through a self-generated dataset based on Citi Bike’s trip data (origin-destination data) in New York City. It also known that high quality relevant features can improve model performance, sometimes being more impactful than the choice of algorithm itself (Shaik, 2024). So, the goal of the project is to develop relevant and high quality features that can sufficiently accommodate demand variation across space and time, i.e., feature engineering using spatial and temporal information.

It is important to make a distinction between short term demand prediction and medium term demand prediction to highlight the difference in why each of them could be useful for bike sharing systems. Short-term forecasting typically focuses on predicting demand 15 minutes to a few hours ahead (Dastjerdi & Morency, 2022) for capturing immediate fluctuations in demand. Since this approach involves modeling rapid demand fluctuations and complex non-linear relationships, the volume of data required is larger, in addition to it being more granular. This is why complex algorithms are more appropriate to achieve high accuracy in capturing the demand changes (Li et al., 2022). Short term predictions at fine temporal resolutions are highly effective in addressing immediate operational challenges. They allow operators to redistribute bikes in real time, i.e. dynamic rebalancing, to avoid empty or overcrowded stations (Jia et al., 2022).
On the other hand, medium-term forecasting generally looks at daily or weekly demand patterns (Guido et al, 2019) for identifying broader trends. Since data is aggregated on daily or weekly basis, the focus is more on identifying recurring usage patterns over longer periods, rather than modeling fine-grained fluctuations. This makes data requirement to be less extensive and makes simpler, low-dimensional approaches (eg. regression models such as ARIMA/Ridge Regression) more appropriate for medium-term forecasting (Wang, 2024). When it comes to strategic planning, medium-term forecasts can help plan static overnight rebalancing operations and maintenance scheduling by identifying recurring patterns (Torres et al., 2024). Unlike dynamic rebalancing, which reacts to real-time fluctuations, medium-term forecasts allow operators to proactively address imbalances before they occur (Liang et al., 2024).

This projects aims to bridge the gap between short-term and medium-term forecasting, so that finer demand fluctuations can be captured at a granular level while also preserving the ability to plan for fleet management (eg. station inventory optimization and maintenance scheduling) for the upcoming days. This approach is intended to cater to both static and dynamic rebalancing, so that instances of lost demand can be reduced. While there are many different ways to address the rebalancing and maintenance scheduling problem, or solve broader issues in fleet management, the scope of this project is limited to performing short-to-medium term demand prediction using tree-based ensembles. In particular, demand prediction is meant to be prediction of future rentals only at operational stations for the bike sharing system, i.e. Citi Bike.

Dataset Preparation

Citi Bike System Data

The system data includes trip data (origin-destination data) of its riders in a csv format, beginning from 2013. The data is refined to exclude trips made by staff for system maintenance and inspections, trips to or from designated "test" stations (which were more frequently used in June and July 2013), and trips lasting less than 60 seconds, as these might represent false starts or attempts to re-dock a bike securely. For this project, only the data for the first four months of 2024 has been used. During this time period, the bike sharing system had over 2,000 stations operating in New York City, US. According to Citi Bike's official website, the bikes can be unlocked from one station and returned to any other station in the system. The following details are available in the system generated data:

- Ride ID (unique trip identifier)
- Rideable type (electric bike versus classic bike)
- Started at (temporal information — contains both date and timestamp, accurate to milliseconds)
- Ended at (temporal information — contains both date and timestamp, accurate to milliseconds)
- Start station name (including street name and number)
- Start station ID (Citi Bike's own assignment of station IDs)
- End station name (including street name and number)
- End station ID (Citi Bike's own assignment of station IDs)
- Start latitude (spatial information)
- Start longitude (spatial information)
- End latitude (spatial information)
- End Longitude (spatial information)
- Member or casual ride (membership type)

Data Pre-processing

Using the Citi Bike system data, a new dataset is generated that is more appropriate for applying machine learning methods. The data pre-processing steps discussed in this section.

From the features provided by the system generated data, only the following are retained:

- Started at (temporal information — contains both date and timestamp, accurate to milliseconds)

- Start latitude (spatial information)

- Start longitude (spatial information)

In some cases, the same station has been assigned multiple IDs by Citi Bike. In addition, some station IDs are stored as integer values while other are stored as a string. This makes the station ID variables unreliable to uniquely identify stations in the distribution network. The stations are assigned new IDs so that they can be used as a unique identifier for a given station. Station names were not used as a unique identifier because for some stations, the same name corresponded with slightly different location coordinates. For example, station named Amsterdam Ave & W 119 St has two sets of coordinates: (-73.959586, 40.808632) and (-73.959621, 40.808625). With station IDs uniquely assigned, station names are unnecessary, hence removed.

For electric bike as rideable type, the dataset contains a lot of missing values related to the station from where the trip began. Since this information is essential to identify rental demand for a given station and this rideable type is in clear minority, this rideable type is removed from the dataset and only 'classic bikes' is considered.

Since data is only available at trip level and not at station level, demand at a station is calculated by aggregating the number of trips that originated from that station, within a specified time window. This means that the information about the number of bikes being deposited at a station after use, at any given time, is left out. The feature specifying the type of membership is not retained, since trips are being aggregated for a station irrespective of whether it took place for a member or a casual rider. This could be a limitation since usage pattern can depend on membership type.

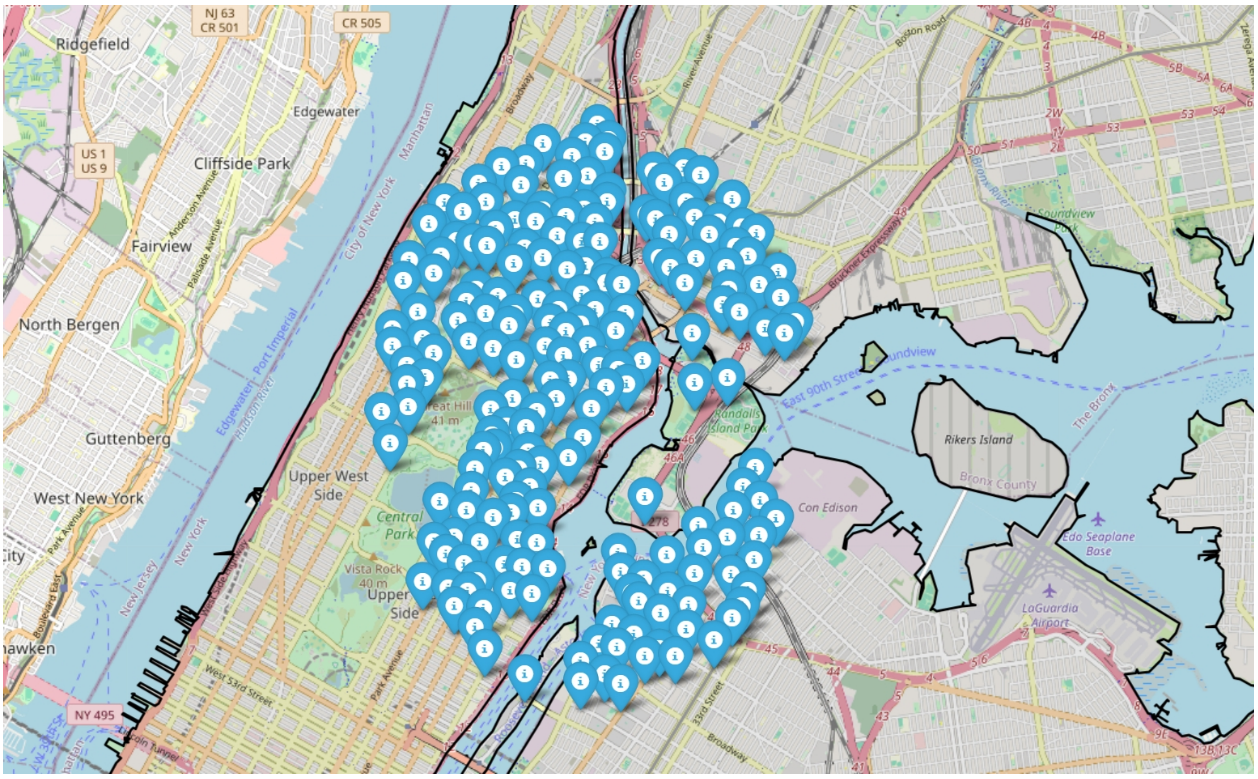
Citi Bike stations are functional throughout the day. However, it is found that rentals typically take place between 08:00 to 22:00 hours. Hence, instead of the entire day, only this time duration is considered. For this duration, specific time windows are generated, within which rental demand is calculated for a station. For each timestamp date, the duration (from 08:00 to 22:00 hours) is broken down into seven chunks of two-hour windows as follows:

- 08:00:00.000 (denotes time window: 08.00 – 10.00)
- 10:00:00.000 (denotes time window: 10.00 – 12.00)
- 12:00:00.000 (denotes time window: 12.00 – 14.00)
- 14:00:00.000 (denotes time window: 14.00 – 16.00)
- 16:00:00.000 (denotes time window: 16.00 – 18.00)
- 18:00:00.000 (denotes time window: 18.00 – 20.00)
- 20:00:00.000 (denotes time window: 20.00 – 22.00)

This time resolution is not extremely granular for real-time demand prediction (such as predicting demand 15 minutes into the future), but is granular enough to determine significant demand fluctuations throughout the major portion of the day (from 08:00 to 22:00 hours) and perform planning tasks for the upcoming days. This measure was also taken so that the size of the generated data can be bounded, to control training time while applying tree-based ensembles. The available temporal information from the provided timestamp is also broken down into individual temporal features such as: *year, month, day, hour.* Note that the hour variable in the developed dataset can only have any of the seven “hour” values highlighted above, and not all hours present in a day. Hence, so far, nine features have been developed from the system data:

starting latitude, starting longitude, starting station ID, start trip timestamp (datetime), aggregated number of rentals, year, month, day, hour.

Throughout the four month time period, from January 2024 to April 2024, 2088 stations that were in common for all four months have been considered. Since the entire four month dataset for 2088 stations is very huge, only 200 stations that are in close vicinity to the station *1 Ave & E 110 St* have been chosen. These 200 stations (about 10% of total stations) contain a mix of initial launch areas (Mid town Manhattan), as well areas of The Bronx, Queens and Upper Manhattan with relatively low expected usage. The distribution of bike stations can be seen in the figure below.



Distribution of the 200 selected Citi Bike stations

Methodology & Analysis

Feature Engineering

As a baseline measure of performance, a random forest learner in python's sickit-learn library is used, using the default hyperparamaters. The model is trained separately for the first 25 days of February 2024 and for the first 25 days of March 2024, using seven features: *aggregated number of rentals, year, month, day, hour, starting station ID and rentals lagged by one time period (explained below)* as baseline features. This length of training data is considered so that the most recent data is used for prediction, while also limiting data size to reduce training time. The length of 25 days is meant to be large enough to provide a rough idea about prediction performance on test set. A more elaborate discussion about the length of training data is included further in the next section of the paper. The performance is evaluated on the remaining days of each month (test set), separately. It is found that for the baseline,

the MSE scores are 2.722 (on test data for Feb 2024) and 3.915 (on test data for Mar 2024) and,

the r2 scores are 0.256 (on test data for Feb 2024) and 0.444 (on test data for Mar 2024)

To evaluate the suitability of new features created, the models are trained again in the same manner with new features added, and the resulting performance over the test set of each month is averaged, so that it can be compared with the baseline.

Many features used or developed in this project, such as 'station ID' or 'name of day' (explained below), have unordered categorical data. Since Random Forests are ensembles of decision trees, they make splits based on feature values. However, if a categorical feature is numerically encoded, the trees can create splits like "value < 23," which has no meaningful interpretation for unordered categories. With a numerical representation, the model might incorrectly assume that one category is "greater than" or "lesser than" the other category, which is not the case with unordered data (Smith et al, 2022). The issue of creating an artificial order is resolved with one-hot encoding where each category is transformed into a separate binary feature, i.e. dummy. With dummies, splits are now made on presence or absence of a category (True/False values), although it can lead to performance reduction if data dimensionality increases dramatically due to presence of 'too many' categories. Some categorical features explained below were treated with one-hot encoding because performance advantage in terms of lower MSE and higher r2 score was found, while that feature was individually being added. While one-hot encoding can be beneficial if cardinality is not too high, it is generally found that alternative forms of encoding such as categorical handling and target encoding are better suited for random forests (Zhu et al, 2024). The new features developed are as follows:

- Weather information

  Data from Python's Meteostat library is used to get weather related features. Information is retrieved using the location information from individual station coordinates. The five weather variables used are as follows: *temperature (in Celsius), relative humidity, precipitation, wind speed* and *weather condition code (abbreviated as "coco")*. According to Meteostat Developers' website the variable 'coco' has 27 categories in total, out of which 16 categories are present in the data. Each code (integer value) represents a weather condition category, such as Clear (code:1), Fair (code:2), Cloudy (code:3), etc. Only the coco variable is treated using one-hot encoding, so that all categories are represented by a dummy. On average, MSE dropped by 0.436 and r2 score improved by 0.1.

- Rentals lagged by 1 time unit

  For a given station and date at hour "t", the aggregated number of rentals made from that station on the same date in the previous time window (i.e. at hour "t-2") is considered as an explicit feature. In simple words, it's the rental demand from the previous time unit. Since the first time period for any day is 08:00:00.000 (denoted time window: 8.00 – 10.00), there is no value available since no previous time unit exists. Hence, for this time period, the lagged value is NaN. This feature is included as part of the baseline model. For the February data, MSE: 2.722 and r2 score: 0.256, and for the March data, MSE: 3.915 and r2 score: 0.444.

- Rentals lagged by 2 time units

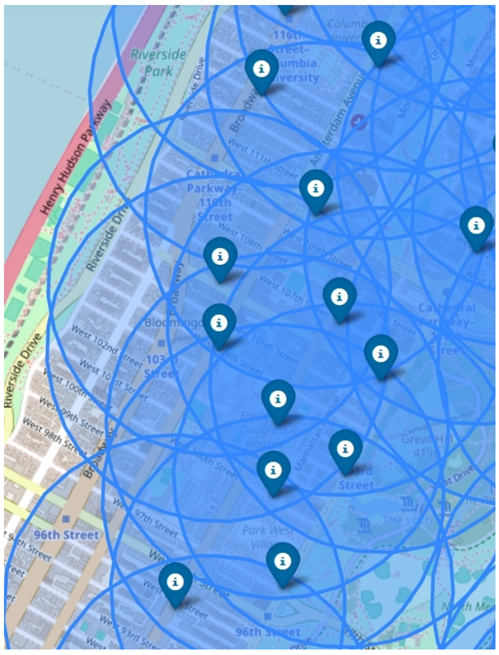
  Similar to the feature above, aggregated rentals are considered from the time period that exists before the previous time period. For any day, 08:00:00.000 (denoted time window: 8.00 – 10.00) and 10:00:00.000 (denoted time window: 10.00 – 12.00), both periods are assigned NaN values for this feature, since the two prior time period doesn't exist for either of them. On average, MSE dropped by 0.13 and r2 score improved by 0.03.

- Weekend Indicator

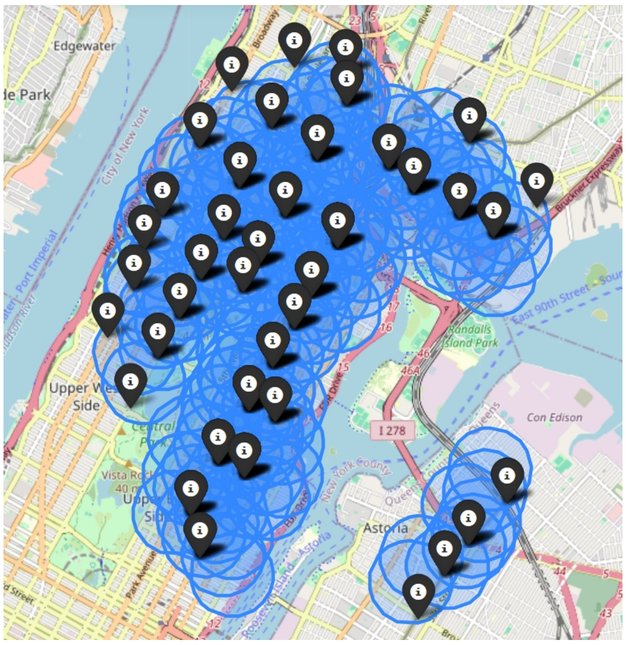
  This feature takes the value of "1", if the day is either 'Saturday' or 'Sunday', else "0". It is meant to capture usage patterns on the weekend, since weekends are expected to have different usage patterns compared to weekdays. On average, MSE dropped by 0.34 and r2 score improved by 0.08.

- Number of Subway stations nearby

  It is mentioned that 74% of Citibike stations are within 400m of any subway entrance in NYC, for the areas in which the service was initially launched (Kaufman, 2015). Keeping some margin of error, a buffer zone of *800m* is created (blue circles) as shown in the figure on the right for each Citibike station (dark blue pin), to calculate the number of subway stations lying within this proximity. An extended buffer is considered because Upper Manhattan area as well as parts of Bronx and Queens are seen to have less dense coverage for Citibike stations, and subway stops are farther apart



compared to areas such as Lower Manhattan. The buffer of 800m was also considered because it provides a better range of values for the number of stations in the vicinity. According to Wang & Noland (2021), the number of subway stations lying in proximity to bike-sharing stations has positive correlation with rentals. The 'number of subway stations in proximity' ranged from 0 to 4 with the distribution as follows:



- Bike stations with no subway in proximity: 49
- Bike stations with only one subway st. in proximity: 68
- Bike stations with two subway sts. in proximity: 77
- Bike stations with three subway sts. in proximity: 5
- Bike stations with four subway sts. in proximity: 1

The figure on the right shows the subway stations marked by black pins and the original 800m buffer zones by blue circles. The number of black pins within each blue circle is calculated. The above distribution suggests that there are 151 Citi bike stations that have at least one subway station in their vicinity of 800m. This variable is not treated using one-hot encoding, because the original numeric form yielded better performance. Using this feature, MSE dropped by 0.04 and r2 score improved by 0.01, on average.

- Day of Week (Name of Day)

  The feature is used as dummy variable using one-hot encoding, with the seven dummies corresponding to the day of the week. On average, MSE dropped by 0.38 and r2 score improved by 0.1.

- Previous Day Rentals

  The aggregated rentals today at a given time, for a given station, might have some relationship with rentals the previous day for the same time and station. This information is captured in the *prev_day* variable. On average, MSE dropped by 0.16 and r2 score improved by 0.04.

- Station ID (manually created; unique station identifier)

  The feature is used as dummy variable using one-hot encoding, with each dummy corresponding to one of the 200 unique station IDs. This feature is included as part of the baseline model. Treating with one-hot encoding led to drop in MSE by 0.22 and improvement in r2 score by 0.05, on average.

- Rolling average, over past 7 days

For a given hour of day, an average value of rentals over the entire 7 days (for the same hour) prior to the current date is calculated, for a given station. This is considered a rolling average since the value is ‘rolling’ or moving with date. On average, MSE dropped by 0.215 and r2 score improved by 0.11. The improvement was particularly seen for the February data. While developing this feature, some NaN value were created for the first week of a month. Replacing the missing values with appropriate values should yield even better performance.

- Rush Hour Indicator

  For all of the seven time windows, the corresponding mean value of rentals made in all stations are as follows:

  1.676694214876033 (hour 8-10)
  1.3396280991735536 (hour 10-12)
  1.5396694214876032 (hour 12-14)
  1.7802892561983472 (hour 14-16)
  2.0058677685950412 (hour 16-18)
  1.448099173553719 (hour 18-20)
  0.6651239669421488 (hour 20-22)

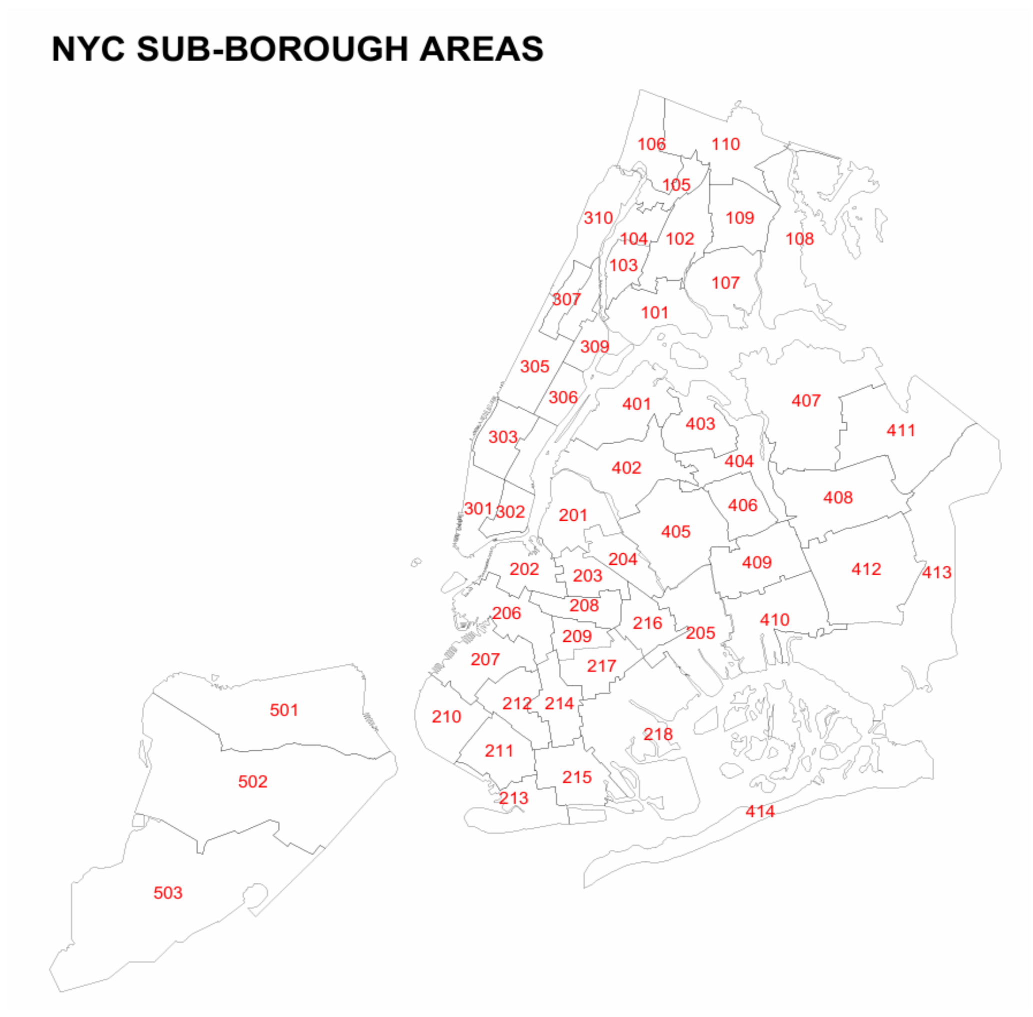
  the corresponding max and median values are as follows (max, median):

  31, 1 (hour 8-10)
  34, 1 (hour 10-12)
  48, 1 (hour 12-14)
  78, 1 (hour 14-16)
  67, 1 (hour 16-18)
  37, 1 (hour 18-20)
  14, 0 (hour 20-22)

  From the basic analysis it looks like there are 3 periods with a lot of rush hour. For hours 8-10, the max isn't very high but the mean is relatively high. This conveys that considerable rentals are made in this time window consistently. For hours 14-16 and 16-18, it looks like both max and mean are high. This also suggests peak demand taking place in these time windows. Hence, the *rush hour indicator* is “1”, if time of day (hour) falls within any of the three time windows, else “0”. Using this indicator as a feature resulted in drop in MSE by 0.04 and an increase in the R2 score by 0.008, on average.

- Adding Borough and Sub-Borough information in a single consolidated variable (Map ID)

  The dataset consists of three boroughs, Manhattan, The Bronx and Queens. However, each borough contains many sub-boroughs. A map illustrating sub-boroughs, as codes in New York City, can be seen below (infoshare.org, Community Studies of New York):



In addition to individual location coordinates of a station, a variable that can contain information about borough and sub-borough is likely to capture usage patterns that are more localized to a sub-borough, so that all the stations that belong to that sub-borough could be identified. The "MapID" variable is created for this purpose, with a three digit code (as illustrated in the map above) that is unique to each sub-borough. The first digit is meant to denote the borough and the remaining two digits are meant to denote the sub-borough area. For example, in code 109, "1" denotes the borough "Bronx" and "09" denotes the "Pelham Parkway" sub-borough area. A complete list of sub-boroughs with their corresponding codes can be found here. All the codes were treated using one-hot encoding using dummy variables. The MSE reduced by 0.5, on average.

- Lagged Rentals from nearby stations, by one time period

Here, instead of taking the aggregated rentals, lagged by one time period, from the same station, the lagged rentals are taken from the stations in the vicinity of a given station. Multiple approaches were tried to see what works best. First, by taking lagged rentals from the nearest, second nearest and third nearest stations (lagged by one time period in each case). In this approach the problem to consider was, how many nearby stations should be considered for best performance (On average, MSE dropped by 0.295). Second, by using a weighted average where the nearest station is manually assigned highest weight and the two farther stations are assigned lesser weight, depending upon the distance from the station being considered. In this approach the problem to consider was, what should the ideal weight assignment be, since the distance of first, second and third nearest station is varying, depending on a given station (For February data, MSE dropped by 0.12). Third, an absolute average was considered where equal weights are assigned to each of the three stations. However, this led to relative poor performance and dis-proportionally higher importance to a far-away station (On average, MSE dropped by 0.215).

To address the manual weighting problem, we can consider all stations and give them preferential treatment based on their distance from the station in consideration. This was done using the Gaussian Kernel (McCormick, 2013), which assign weights automatically using the mentioned formula below. This kernel is used to assign spatial weights based on distance of all other stations with respect to a particular station. Higher weight (greater importance) is assigned to stations with shorter distances because of its spatial proximity to the station in consideration. The formula for the Gaussian function:

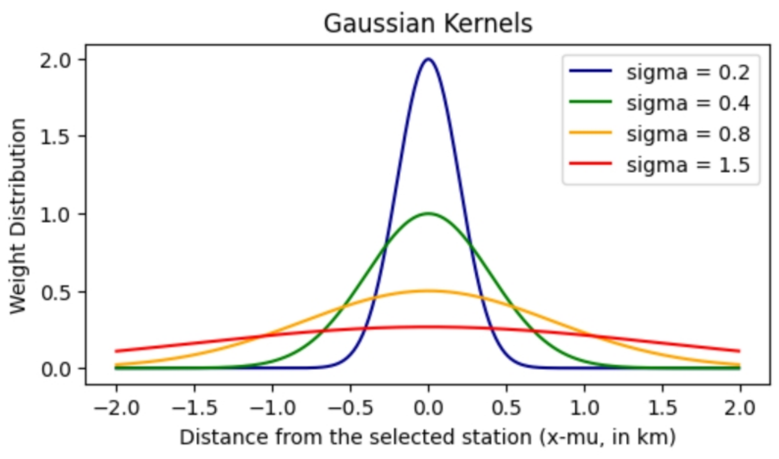
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where,

x-mu: projected distance between other stations and the station in consideration and,

sigma: parameter that controls the width of the Gaussian window, lower value corresponds to narrower window around mean (mu); faster weight decay.

The different widths of the Gaussian windows (depending on the value of sigma) are illustrated in the figure below:



For cities such as New York with dense urban neighborhoods, it is found that potential riders prefer to walk about 1,000 feet or 300 meters and considerable drop-off in the desire to rent a bike begins appearing when the distance is more than 0.25 mile or 405 meters (NACTO, 2015). If someone is more than 500 meters away from the station, the likelihood of bike rental becomes close to none. Since in this paper, the areas being covered are less dense than the most dense areas in the city, such as business districts of lower Manhattan, have relatively wider streets, and have less dense coverage of CitiBike stations, the critical distance is increased from 300 meters to 500 meters. Based on this information and accommodating for some margin of error, usage patterns of bike stations that are within 500 meters of a particular bike station should strongly capture the localized demand fluctuations of the region and behave as a useful feature in explaining the future rentals from that particular station. Stations that are lying a 1000 meters away should carry no significant information that would be useful to predict future rentals. The most appropriate sigma value was found to be 0.4, as it significantly tapered weights by distance for stations lying 500m afar while prioritizing those that are lying within 500m of distance. Hence, the weights generated from this kernel are now used to reach at a weighted average by treating rental values from all other stations with assigned weights. For every station, a unique weight matrix is created that carries the weight information for all the other stations.

Using this approach, the total rentals made only in the previous time window, for every other station, were averaged to arrive at a weighted average that is calculated for ever station. The MSE dropped by 0.313 and r2 score improved by 0.116. Hence, compared to the previous three weighting approaches, assigning weights using the Gaussian Kernel is clearly superior in terms of performance.

- Rentals from previous day, from nearby stations

  Rentals are considered similar to the 'Previous Day Rentals' feature explained before. However, instead of taking values from the same station, a weighted average of values (for previous day, same time) from all other stations is taken using the Gaussian Kernel as explained previously. In simple words, weighted average over lagged rentals from all other stations is performed, where values are lagged by an entire day. On average, MSE dropped by 0.244 and r2 score improved by 0.06.

- Rentals from rolling average, from nearby stations

  Rentals are considered similar to 'rolling average for past 7 days' feature as explained before. However, instead of taking values from the same station, a weighted average of values (averaged over 7 days prior, same time) from all other stations is taken using the Gaussian Kernel as explained previously. On average, MSE dropped by 0.21 and r2 score improved by 0.05.

- Lagged Rentals from nearby stations, by two time periods

  Rentals are considered similar to 'Lagged Rentals from nearby stations, by one time period' feature as explained before using the Gaussian Kernel. The only difference is that the values are now lagged by two time units instead of one. On average, MSE dropped by 0.17 and r2 score improved by 0.04.

Altogether, there are 13 additional features that have been developed on top of the baseline model. While many features have been generated in this section, these features are standalone. Literature suggests that creating interactive terms by combining two or more features can also yield good performance, however this approach is not covered in this paper and can be seen as a limitation of the study. Arranging the features solely on the basis of average drop in MSE, the feature importance can be ranked as follows, in descending order of importance:

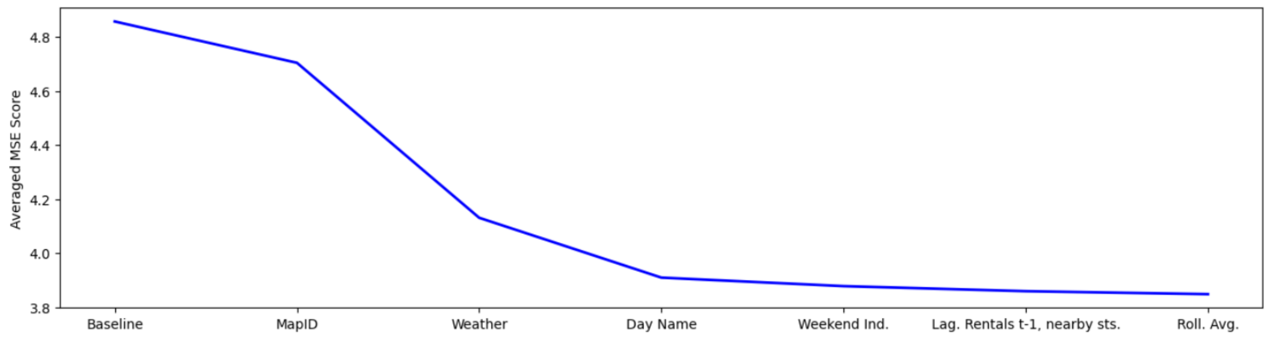
1. Borough and Sub-borough Information (i.e. MapID): MSE dropped by 0.5
2. Weather Information: MSE dropped by 0.436
3. Day of week (name of day): MSE dropped by 0.38
4. Weekend Indicator: MSE dropped by 0.34
5. Lagged Rentals from nearby stations, by one time period (Gaussian Kernel): MSE dropped by 0.313
6. Rentals from previous day, from nearby stations (Gaussian Kernel): MSE dropped by 0.244
7. Rolling average, over past 7 days: MSE dropped by 0.215
8. Rentals from rolling average, from nearby stations (Gaussian Kernel): MSE dropped by 0.21
9. Lagged Rentals from nearby stations, by two time periods (Gaussian Kernel): MSE dropped by 0.17
10. Previous Day Rentals: MSE dropped by 0.16
11. Rentals lagged by 2 time units: MSE dropped by 0.13
12. Number of subway stations nearby: MSE dropped by 0.04
13. Rush Hour Indicator: MSE dropped by 0.04

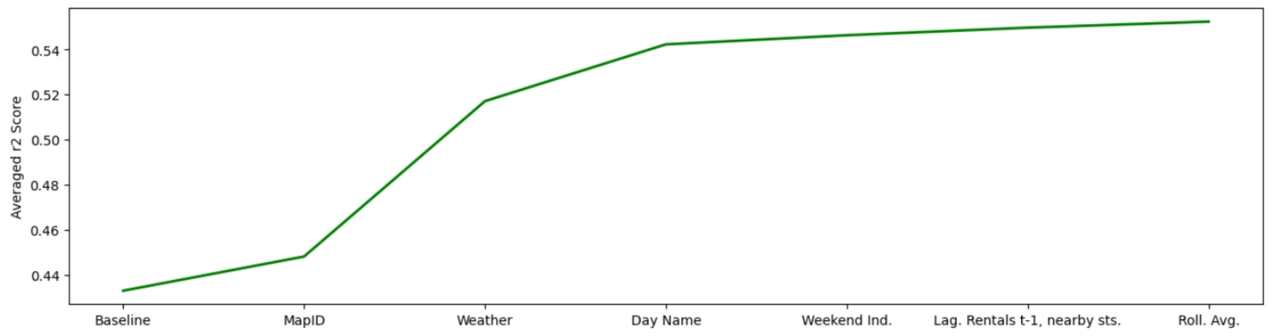
Data Segmentation and Feature Selection

All 13 of the developed features may not contribute to the overall model's performance in predicting future rentals. Using the information of feature importance that was seen in the last section, features are added incrementally (in order of importance) to the baseline model until adding a feature leads to no increase in performance. Unlike the previous section though, more data will be used for training, while still preserving the same learner (i.e. random forest with default hyperparameters).

To accommodate for hyperparameter tuning in the later part of this paper, the entire dataset is divided into 3 parts: training set, validation set and test set. There are two test sets generated, one for the last week of March and the other for the last week of April. The week previous to the last week of March is taken as the validation set. This set will be used for hyperparameter tuning. The 6 weeks prior to the time period considered for the validation set, is taken to be the first training set. The second training set is the 6 weeks prior to (and excluding) the third week of April. The third week is not included in training so that it can be used later for tuning hyperparameters, in case that is needed. The test set for both months remain unchanged. The data for both months has been divided into three sets, each, in such a manner that the temporal order is preserved.

Hence, the performance is evaluated using two different models that are using the same learner (and hyperparameters) i.e. random forest, but are trained on the most recent 6 weeks data (excluding the validation set period). Training is not performed for more than six weeks to limit the computational requirement. The performance is evaluated by averaging the MSE for the two test sets. The performance change on test set for feature selection can be seen in the two plots below:





In both the plots, for every feature that is added incrementally to the models, the performance change is significant for the initial set of features added to the baseline but plateaus towards the end. Adding any other feature does not result in a performance increase, but actually worsens the performance in some cases. Hence, out of the 13 generated features, only six features added on top of the baseline result in a considerable performance gain. The plots for performance metrics of individual months can be found in the appendix.

For most of the features, they are added according to the feature importance ranking that was shown in the previous section. However, the sixth feature “weighted average of previous day rentals, from nearby stations only” led to an overall reduction in performance. Adding the successive feature i.e. “Rolling average of rentals, from the last 7 days” on top of this feature did not fix the problem. Hence, the feature “weighted average of previous day rentals, from nearby stations only” was completely removed. It was found that after the fifth feature i.e. “rentals lagged by one time period, for nearby stations only” is added, adding the feature “Rolling average of rentals, from the last 7 days” did lead to an increase in performance. Hence, this feature was kept as the last feature, replacing “weighted average of previous day rentals, from nearby stations only”.

Another deviation from the previous section is that this time "coco" variable (part of weather information) is added without any one-hot encoding. This is because the categories in the training and the test set differs. Also, the categories of 'coco' follow a logical order, hence dummy variables are not strictly required. Adding dummies for this variable would unnecessarily increase dimensionality, which could penalize performance.

For the baseline model, where features used are the same as baseline in the previous section i.e.: *aggregated number of rentals, month, day, hour, starting station ID and rentals lagged by one time period,* the average MSE is 4.85 and the average r2 score is found to be 0.432. After the six generated features are added on top of the baseline model, the average MSE is 3.84 and the average r2 score is found to be 0.55, with default hyperparameters using random forest.

Model Selection & Tuning

- Random Forest

  For random forest learner, random search method is used to find the best set of hyperparameter values. The performance of the changed hyperparameter values is evaluated on the validation set of March data (i.e. the third week of March) to determine which set of hyperparameters are best. Since it is a time-series problem, cross-validation has not been used so that data is not shuffled and the temporal order is preserved. Out of all the hyperparameters, the focus has been on the four main hyperparametes for random forest learner:

  - ➢ *n_estimators (number of trees),*
  - ➢ *max_depth (max depth allowed for a tree),*
  - ➢ *min_samples_split (minimum number of samples required to split an internal node),*
  - ➢ *min_samples_leaf (minimum number of samples required to be a leaf node)*

  However, tuned model led to poorer performance on test (average MSE was 3.92) compared to the default hyperparameters (average MSE was 3.84). Hence, the default hyperparameter values were preserved. Performance for different hyperparameter values for random forest learner on the validation set is mentioned in the appendix.

- Histogram-based Gradient Boosting Trees (HGBT)

  For this learner, random search is used as with the previous case. The model, using default hyperparameters, provided slightly better performance on test set compared to Random Forest. Average MSE = 3.74, average r2 score = 0.56.

  The performance of the changed hyperparameter values is evaluated on the same validation set to determine which set of hyperparameters are best. Cross-validation isn't used. Out of all the hyperparameters, the focus has mainly been on two main hyperparameters:

  - ➢ *learning_rate*
  - ➢ *max_iterations*

The advantage of using this model is that it's not very sensitive to different hyperparameter values and can provide good performance with minimal tuning. Performance for different hyperparameter values for HGBT learner on the validation set is mentioned in the appendix. After tuning, a slight improve in performance compared to default hyperparameters was observed. Average MSE = 3.70, average r2 score = 0.565.

- Light Gradient Boosting Machine

  For this learner, random search is used as with the previous case. The model, using default hyperparameters, provided slightly better performance on test set compared to Random Forest. Average MSE = 3.75, average r2 score = 0.561. However, this performance was poorer compared to HGBT with default hyperparameters.

  The performance of the changed hyperparameter values is evaluated on the same validation set to determine which set of hyperparameters are best. Cross-validation isn't used. Out of all the hyperparameters, the focus has mainly been on six main hyperparameters:

  - *n_estimators (number of trees to fit)*
  - *learning_rate*
  - *num_leaves (maximum tree leaves)*
  - *max_depth (maximum tree depth)*
  - *reg_alpha (L1 regularisation)*
  - *reg_lambda (L2 regularisation)*

  The model can provide more accurate performance compared to the previous learners. However, it's very sensitive to different hyperparameter values and requires careful tuning. Performance for different hyperparameter values for LGBM learner on the validation set is mentioned in the appendix. After tuning, improvement in performance compared to default hyperparameters was observed. Average MSE = 3.67, average r2 score = 0.57. This is the best performance observed so far, although the improvement did not take place by a large margin.

- Xgboost Regressor

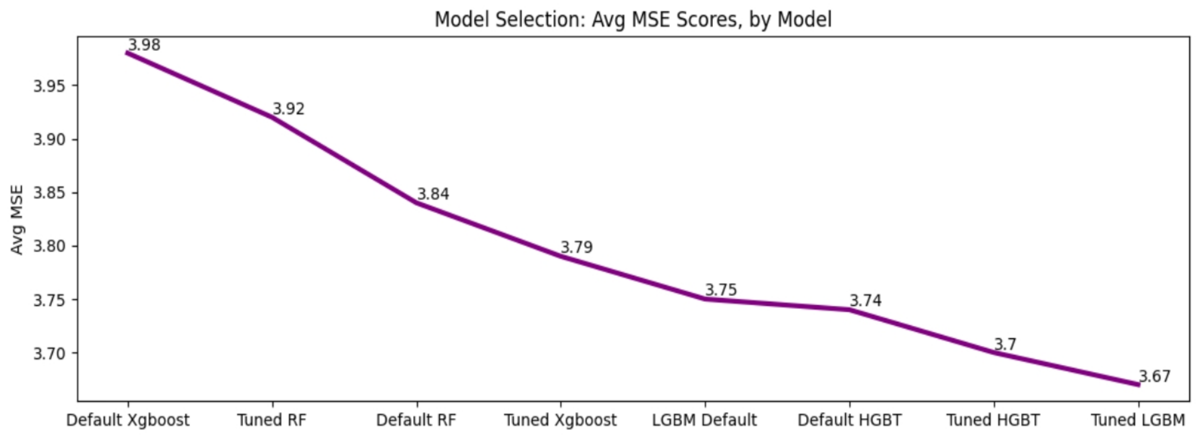
  For this learner, random search is used as with the previous case. The model, using default hyperparameters, provided poorer performance on test set compared to Random Forest. Average MSE = 3.98, average r2 score = 0.53. The performance was naturally also poor compared to HGBT and LGBM, both with default hyperparameters. Hence, with default hyperparameters, it's the worst performing learner.

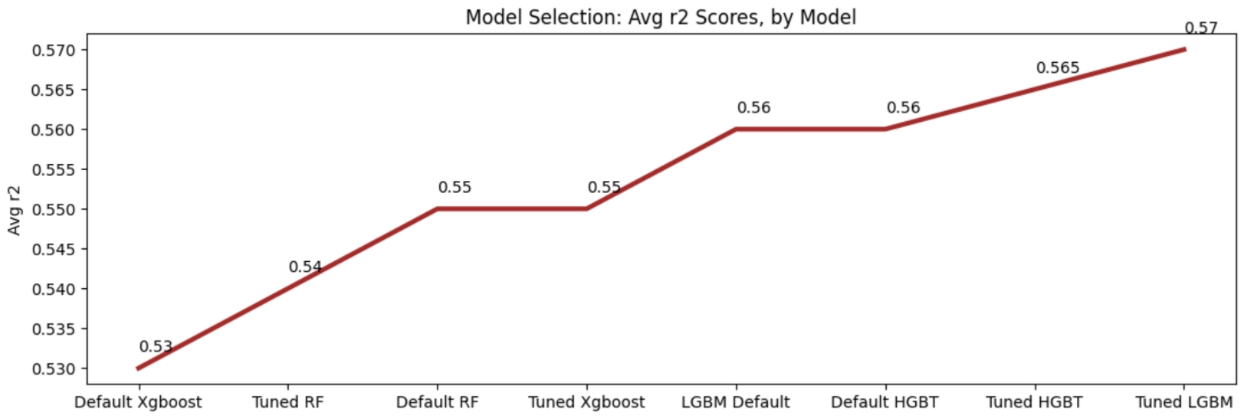
  The performance of the changed hyperparameter values is evaluated on the same validation set to determine which set of hyperparameters are best. Cross-validation isn't used. Out of all the hyperparameters, the focus has mainly been on five main hyperparameters:

  - *n_estimators (number of trees)*
  - *learning_rate*
  - *max_depth (max tree depth allowed)*
  - *subsample (portion of samples/data per tree)*
  - *reg_lambda (L2 regularization)*

The learner can provide very accurate performance compared to random forest and HGBT. However, it's very sensitive to different hyperparameter values and requires careful tuning. Performance for different hyperparameter values for Xgboost learner on the validation set is mentioned in the appendix. After tuning, improvement in performance compared to default hyperparameters was observed. Average MSE = 3.79, average r2 score = 0.55. The performance is now better than RF, but still lags behind LGBM and HGBT.

The performance on the test set for all the four learners is illustrated in the graphs below:





Conclusion & Limitations

In this study, the most useful features in predicting future rentals at individual bike stations were identified and ranked in importance. These features are helpful in capturing usage patterns that vary with location and time. The paper also illustrates using a Gaussian Kernal as means to assign spatial weights to nearby stations, and its benefit over taking the absolute average or manually figuring out weights. Light Gradient Boosting Machine was identified as the most superior learner in terms of performance, however it is closely followed by Histogram-based Gradient Boosted Trees. But how can the goodness of MSE values from these value be determined?

In the test set, if the 'mean' of rentals is always taken to be the predicted value, the MSE (i.e. variance) comes out to be 8.59. This means a RMSE or standard deviation of approximately 2.93 bikes. In simple words, an error of about 3 bikes, on average, would be made on the test set if mean is always taken to be the predicted value. The RMSE with tuned LGBM model comes out to be 1.9 bikes. So, the error now reduces from 3 bikes to 2 bikes, on average, on the test set. While this improvement is positive, a DNN based learner might be required for achieving better performance.

This study only focused on random search method for identifying hyperparameters. Random search requires less computational resources and is faster to perform, however it doesn't exhaustively search the entire parameter space. Advanced tuning methods (such as Bayesian Optimisation or grid search) might identify even better hyperparameter values. Additionally, to preserve temporal order, this study used very simple train-validation-test splits. Cross validation methods (such as TimeSeriesSplit in Python's sk-learn library) that are dedicated to time-series data can improve the tuning process.

References

- Hamilton, T. L., & Wichman, C. J. (2017). Bicycle infrastructure and traffic congestion: Evidence from DC's Capital Bikeshare. *Journal of Environmental Economics and Management*, *87*, 72–93. https://doi.org/10.1016/j.jeem.2017.03.007
- Griffin, G. P., & Sener, I. N. (2016). Planning for Bike Share Connectivity to Rail Transit. *Journal of public transportation*, *19*(2), 1–22. https://doi.org/10.5038/2375-0901.19.2.1
- Kabra, A., Belavina, E., & Girotra, K. (2018). Bike-Share systems: accessibility and availability. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2555671
- Department for Transport, Rabl, A., De Nazelle, A., Steer Davies Gleave, Department for Transport, Department for Transport, Street Davies Gleave, & Department for Transport. (2017). *Bike Sharing: It's about the community.* https://cyclingindustries.com/fileadmin/content/documents/170707_Benefits_of_Bike_Sharing_UK_AI.pdf
- Ranaiefar, F., & Rixey, R. A. (2016). *Bike Sharing Ridership Forecast using Structural Equation Modeling*. https://www.semanticscholar.org/paper/Bike-Sharing-Ridership-Forecast-using-Structural-Ranaiefar-Rixey/bf603a67d91ffc191b5d9f0180367d688703ecfd
- Moura, F., Valença, G., Félix, R., & Vale, D. S. (2023). The impact of public bike-sharing systems on mobility patterns: Generating or replacing trips? *International Journal of Sustainable Transportation*, *17*(11), 1254–1263. https://doi.org/10.1080/15568318.2022.2163209
- *Hallisey, K. (2022). How riding a bike benefits the environment*. Transportation. https://transportation.ucla.edu/blog/how-bike-riding-benefits-environment#:~:text=Cleaner%20Air&This%20affects%20our%20health%20and,add%20pollution%20to%20the%20atmosphere
- Mahajan, S., Argota Sánchez-Vaquerizo, J. Global comparison of urban bike-sharing accessibility across 40 cities. *Sci Rep* 14, 20493 (2024). https://doi.org/10.1038/s41598-024-70706-x
- Adamjee, L. (2024, May). *Citi Bike's impact on New York City transportation*. The Science Survey. https://thesciencesurvey.com/news/2024/05/14/citi-bikes-impact-on-new-york-city-transportation/
- Zhang, Y., & Mi, Z. (2018). Environmental benefits of bike sharing: A big data-based analysis. *Applied Energy*, *220*, 296–301. https://doi.org/10.1016/j.apenergy.2018.03.101
- Chen, Y., Zeng, D., Deveci, M., & Coffman, D. (2024). Life cycle analysis of bike sharing systems: A case study of Washington D.C. *Environmental Impact Assessment Review*, *106*, 107455. https://doi.org/10.1016/j.eiar.2024.107455
- Cao, J., Xu, W., & Wang, W. (2022). Bike-Sharing fleet allocation optimization based on demand gap and cycle rebalancing strategies. *Scientific Programming*, *2022*, 1–14. https://doi.org/10.1155/2022/1892836

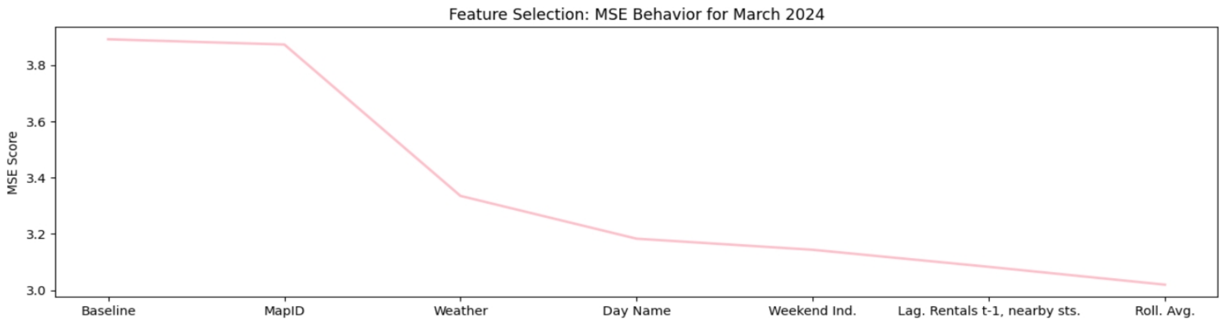
- Salah, I. H., Mukku, V. D., Kania, M., & Assmann, T. (2021). Towards sustainable Liveable city: Management operations of shared Autonomous Cargo-Bike fleets. *Future Transportation*, *1*(3), 505–532. https://doi.org/10.3390/futuretransp1030027
- Y. Liang, G. Huang and Z. Zhao, "Bike Sharing Demand Prediction based on Knowledge Sharing across Modes: A Graph-based Deep Learning Approach," *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, Macau, China, 2022, pp. 857-862, doi: 10.1109/ITSC55140.2022.9922276.
- Kim K. (2023). Discovering spatiotemporal usage patterns of a bike-sharing system by type of pass: a case study from Seoul. *Transportation*, 1–35. Advance online publication. https://doi.org/10.1007/s11116-023-10371-7
- Chiariotti, F., Pielli, C., Zanella, A., & Zorzi, M. (2018). A dynamic approach to rebalancing Bike-Sharing systems. *Sensors*, *18*(2), 512. https://doi.org/10.3390/s18020512
- Affonso, R. C., Couffin, F., & Leclaire, P. (2021). Modelling of User Behaviour for Static Rebalancing of Bike Sharing System: Transfer of Demand from Bike-Shortage Stations to Neighbouring Stations. *Journal of Advanced Transportation*, *2021*, 1–15. https://doi.org/10.1155/2021/8825521
- Chaurey, S., Jodhe, S., Yawalkar, P., Sheikh, S., Mahajan, R., & Mrs.Kamal.S.Chandwani. (2024). Predictive modeling for bike rental demand: Enhancing operational efficiency and user satisfaction in urban mobility. *International Journal for Multidisciplinary Research*, *6*(6). https://doi.org/10.36948/ijfmr.2024.v06i06.30486
- Kaufman, S. M. (2018). Citi Bike: The first two years. *NYU Wagner.* https://wagner.nyu.edu/impact/research/publications/citi-bike-first-two-years
- NEW YORK METROPOLITAN TRANSPORTATION COUNCIL • NEW YORK CITY. (2021). Coordinated Public Transit-Human Services Transportation Plan for NYMTC Region. In *Final* (pp. 3–3). https://www.nymtc.org/portals/0/pdf/CPT-HSP/NYMTC%20coord%20plan%20NYC%20CH03.pdf
- *Zoning: Districts Guide - Commercial Districts - DCP.* (n.d.). https://www.nyc.gov/site/planning/zoning/districts-tools/commercial-districts-c1-c8.page
- Choi, P., & Choi, P. (2022, July 26). Machine learning: The CitiBike station rebalancing issue. *Data Science Blog*. https://nycdatascience.com/blog/student-works/capstone/solving-the-citibike-station-rebalancing-issue-with-python-machine-learning/
- O'Mahony, E., & Shmoys, D. (2015). Data analysis and optimization for (CITI)Bike Sharing. *Proceedings of the AAAI Conference on Artificial Intelligence*, *29*(1). https://doi.org/10.1609/aaai.v29i1.9245
- Uddin, M., Hwang, H. L., & Hasnine, M. S. (2023). An interpretable machine learning framework to understand bikeshare demand before and during the COVID-19 pandemic in New York City. *Transportation Planning and Technology*, *46*(4), 482–498. https://doi.org/10.1080/03081060.2023.2201280

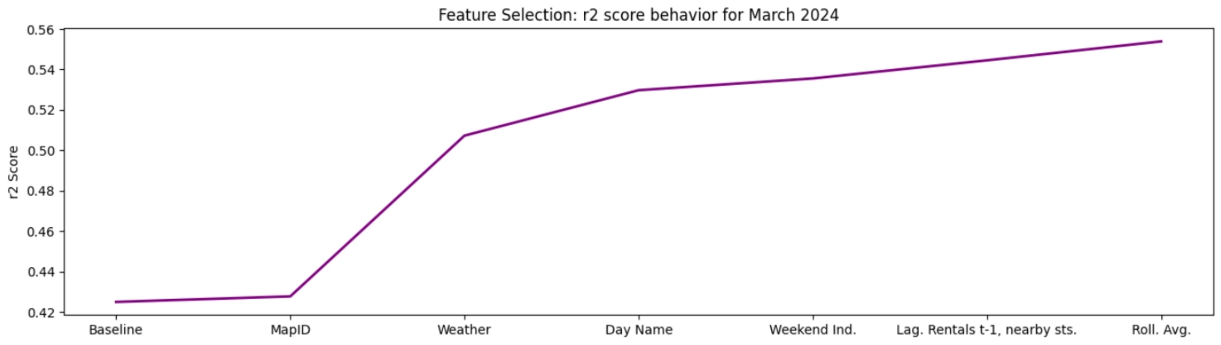
- Dataiku. (2025, January 3). *A Deeper understanding of deep learning | Dataiku*. https://www.dataiku.com/stories/detail/a-deeper-understanding-of-deep-learning/
- Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on tabular data? *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2207.08815
- Roßbach, P. (2023, June 27). *Neural Networks vs. Random Forests – Does it always have to be Deep Learning?* Frankfurt School Blog. https://blog.frankfurt-school.de/neural-networks-vs-random-forests-does-it-always-have-to-be-deep-learning/
- Obermiller, D. (2025). *Outliers and Tree-Based Models: Should we be concerned?* (2025, January 8). https://communities.sas.com/t5/SAS-Communities-Library/Outliers-and-Tree-Based-Models-Should-We-Be-Concerned/ta-p/955438
- Kennedy, W. B. (2025, February 3). *Deep learning for outlier detection on tabular and image data*. Towards Data Science. https://towardsdatascience.com/deep-learning-for-outlier-detection-on-tabular-and-image-data-90ae518a27b3/
- Rane, N., Choudhary, S. P., & Rane, J. (2024). Ensemble deep learning and machine learning: applications, opportunities, challenges, and future directions. *Studies in Medical and Health Sciences.*, *1*(2), 18–41. https://doi.org/10.48185/smhs.v1i2.1225
- Howard, J., & Gugger, S. (2020). *Deep Learning for Coders with fastai and PyTorch: AI Applications Without a PhD*. https://openlibrary.telkomuniversity.ac.id/home/catalog/id/165604/slug/deep-learning-for-coders-with-fastai-and-pytorch-ai-applications-without-a-phd.html
- Shaik, N. (2024, June 1). *Elevating Machine learning performance: the power of feature engineering*. JNRID.org. https://tijer.org/jnrid/viewpaperforall.php?paper=JNRID2406007
- Dastjerdi, A. M., & Morency, C. (2022). Bike-Sharing Demand Prediction at Community Level under COVID-19 Using Deep Learning. *Sensors*, *22*(3), 1060. https://doi.org/10.3390/s22031060
- C. Guido, K. Rafał and A. Constantinos, "A low dimensional model for bike sharing demand forecasting," *2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, Cracow, Poland, 2019, pp. 1-7, doi:10.1109/MTITS.2019.8883283
- Li, X., Xu, Y., Zhang, X., Shi, W., Yue, Y., & Li, Q. (2022, February 9). *Improving short-term bike sharing demand forecast through an irregular convolutional neural network*. arXiv.org. https://arxiv.org/abs/2202.04376
- Wang, Y. (2024). Forecasting demand of shared bikes based on ARIMA model. *SCITEPRESS – Science and Technology Publications, Lda*, 659–664. https://doi.org/10.5220/0012961400004508
- Jia, R., Chamoun, R., Wallenbring, A., Advand, M., Yu, S., Liu, Y., & Gao, K. (2022). A spatio-temporal deep learning model for short-term bike-sharing demand prediction. *Electronic Research Archive*, *31*(2), 1031–1047. https://doi.org/10.3934/era.2023051

- Torres, J., Jiménez-Meroño, E., & Soriguera, F. (2024). Forecasting the Usage of Bike-Sharing Systems through Machine Learning Techniques to Foster Sustainable Urban Mobility. *Sustainability*, *16*(16), 6910. https://doi.org/10.3390/su16166910
- Liang, J., Jena, S. D., & Lodi, A. (2024). Dynamic rebalancing optimization for bike-sharing systems: A modeling framework and empirical comparison. *European Journal of Operational Research*, *317*(3), 875–889. https://doi.org/10.1016/j.ejor.2024.04.037
- *Citi Bike System Data | Citi Bike NYC*. (n.d.). Citi Bike NYC. https://citibikenyc.com/system-data
- *The Gaussian Kernel · Chris McCormick*. (2013, August 15). https://mccormickml.com/2013/08/15/the-gaussian-kernel/
- *Infoshare online*. (n.d.). http://www.infoshare.org/main/userguide.aspx
- Wang, H., & Noland, R. B. (2021). Bikeshare and subway ridership changes during the COVID-19 pandemic in New York City. *Transport policy*, *106*, 262–270. https://doi.org/10.1016/j.tranpol.2021.04.004
- Smith, H. L., Biggs, P. J., French, N. P., Smith, A. N., & Marshall, J. C. (2022). Lost in the Forest: Encoding categorical variables and the absent levels problem. *bioRxiv (Cold Spring Harbor Laboratory)*. https://doi.org/10.1101/2022.09.12.507676
- Zhu, W., Qiu, R., & Fu, Y. (2024). Comparative Study on the Performance of Categorical Variable Encoders in Classification and Regression Tasks. *ArXiv, abs/2401.09682*.
- NACTO Bike SHARE. (2015b). *walkable Station spacing is key to successful, equitable bike share*. https://nacto.org/wp-content/uploads/2015/09/NACTO_Walkable-Station-Spacing-Is-Key-For-Bike-Share_Sc.pdf

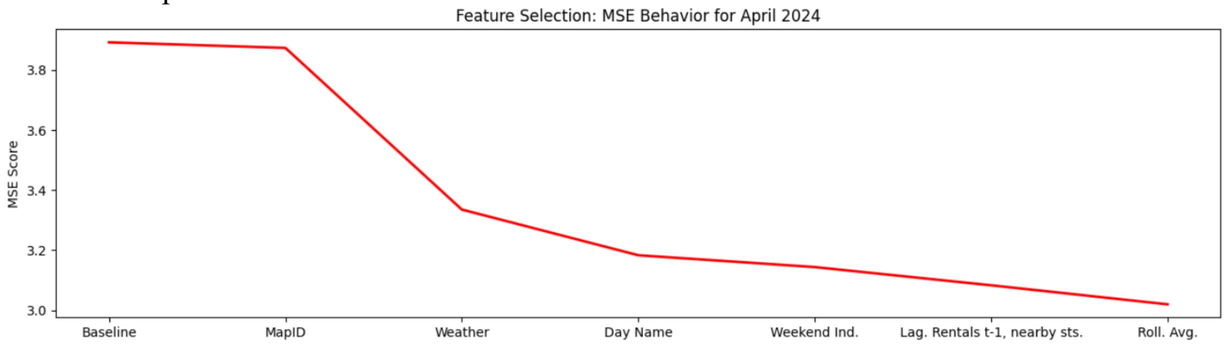
Appendix

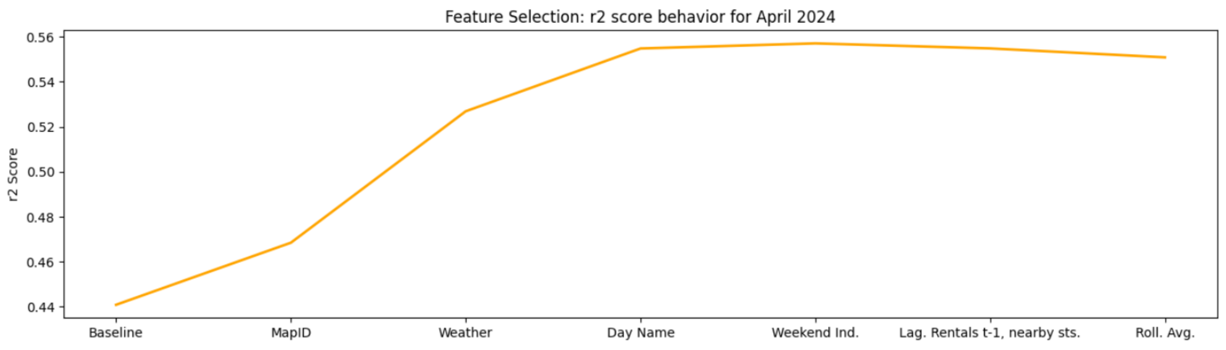
- The performance change on the test set for the month of March 2024, during the feature selection process:





- The performance change on the test set for the month of April 2024, during the feature selection process:





- Hyperparameter Tuning for Random Forest, performance on the validation set for best parameter values found in each trial:

Trial	n_estimators	min_samples_split	min_samples_leaf	max_depth	MSE
1	33	2	3	15	2.503
2	75	2	4	15	2.476
3	120	2	4	15	2.48
4	79	2	4	15	2.478
5	120	2	4	15	2.48

- Hyperparameter Tuning for HGBT, performance on the validation set for best parameter values found in each trial:

Trial	learning_rate	max_iter	MSE
1	0.079	300	2.238
2	0.109	100	2.280
3	0.05	400	2.260
4	0.05	300	2.260
5	0.08	300	2.238

- Hyperparameter Tuning for LGBM, performance on the validation set for best parameter values found in each trial:

Trial	n_estimators	learning_rate	num_leaves	max_depth	reg_alpha	reg_lambda	MSE
1	100	0.089	35	-1	0	0	2.45
2	100	0.119	35	-1	0	0	2.32
3	100	0.102	40	-1	0	0	2.37
4	100	0.116	39	14	0	0	2.30
5	250	0.126	20	11	0	0	2.26
6	250	0.095	25	8	0	0	2.28
7	250	0.099	30	35	0.1	0.5	2.265
8	340	0.1	30	25	0	0.5	2.23
9	335	0.09	32	33	0	0.6	2.29
10	345	0.09	32	32	0	0.6	2.28
11	345	0.09	32	24	0	0.6	2.28
12	365	0.08	140	6	0	0.6	2.22

13	380	0.08	330	6	0	0.8	2.25
14	400	0.08	320	6	0	0.8	2.25

- Hyperparameter Tuning for Xgboost, performance on the validation set for best parameter values found in each trial:

Trial	**subsample**	**reg_lambda**	**n_estimators**	**max_depth**	**learning_rate**	**MSE**
1	0.6	0.7	250	6	0.07	2.38
2	0.6	0.4	200	8	0.05	2.40
3	0.65	0.3	260	7	0.04	2.42
4	0.65	1.2	300	7	0.05	2.33
5	0.6	1.4	240	7	0.07	2.34
6	0.6	1.2	300	7	0.07	2.33
7	0.8	0.3	180	5	0.1	2.47
8	0.7	0.5	360	6	0.04	2.43
9	0.6	0.7	220	6	0.08	2.41
10	0.6	1.4	270	7	0.07	2.34