Final Project

Understanding BIXI users' behaviour based on weather, calendar and geospatial data

By
Gabriella Bincoletto-Montpetit, 11149602
William Désilets, 11285063
Simon Drolet, 11178019

Presented to
Sasha Luccioni

Friday April 16th, 2021

As part of the requirements for MATH8069A – Machine Learning I: Large-scale data analysis and decision making.

HEC MONTRÉAL

Table of contents

Section 1: Introduction

1.1. Context

Despite financial difficulties – including filing for bankruptcy in 2014, resulting in a municipal takeover (Wright, 2014) – the BIXI bike-sharing system has firmly implanted itself within Montreal culture. It's certainly no coincidence that several public investments have been made to keep the company afloat: bike-sharing systems (and cycling in general) have been associated with substantial benefits to travelers' well-being and health issues (Goodman et al., 2014), while improving urban accessibility and reducing the number of cars on the road. Cities implement such systems to "tackle increased expansion of urban mobility, air pollution and changes in urban mobility patterns and behavior" (Albuquerque et al., 2021).

The popularity of such bike-sharing systems (BSS) has been steadily rising in Europe and Asia during the last two decades but has only picked up on this side of the Atlantic recently. As an example, there were around 101 BSS in 125 cities across the world in 2010; among those, only one was located in the US (Zhou, 2015). As of today, according to the Meddin Bike-Sharing World Map, Montreal's BIXI system is still among only 5 North American networks involving 5000 bikes or more. Therefore, our team felt it was only natural to take a closer look at this integral part of our hometown – especially since the company was known to publicly publish its data, perhaps as an effort to crowd source large-scale analysis of eminently complex problems. Indeed, the prediction of bike flow and station-specific demand (and the resulting issue of physical distribution of bikes) are referred to as "challenging" and studies about this topic "leave several issues poorly addressed" (Zhou, 2015). Our team set about finding a way to answer interesting questions and learn more about this topic.

1.2. Research question

Certain studies had already set about predicting spatiotemporal biking patterns (Zhou, 2015) or station-level demand (Lin et al., 2018). Building on these ideas, and banking on complex models' ability to use multi-dimensional data as input, we decided to bring another factor in the mix: weather.

The specified goal of our proposed research is to predict and analyze BIXI users' behavior (within the BIXI season, which usually lasts from April to November), based on historical weather data (temperature, wind, precipitations, etc.), calendar data (time of the day, day of the week, holidays) and geographical location within the island of Montreal. Our proposed approach is split into two specific sub-questions:

1) How can we use weather and calendar data to predict hourly bike demand and average trip duration? (See Section 4)
2) How can we cluster stations together to better understand the bike flow within the city? (See Section 5)

We believe answering these questions could ultimately provide valuable information for demand prediction and physical allocation of bikes, a known challenge for the company. It could also paint a clearer picture of the routes taken by BIXI users, which could facilitate decision making both for operators of the bike-sharing system but also Montreal urban planners.

Section 2: Literature review

As bike sharing systems gain in popularity, the number of studies analyzing flow patterns and predicting demand has also steadily increased. Our team reviewed multiple studies but focused this literature review on two main studies, which were closely tied with our research question. We then provide a general overview of common points found across these studies.

The first study we reviewed focused on understanding spatiotemporal patterns of BSS biking behaviour in Chicago (Zhou, 2015). More specifically, this study aimed to understand how bike flow patterns vary as a result of time, day and user group, as well as analyze the spatiotemporal over-demand for bikes and docks between 2013 and 2014. We mainly focused on the study's first objective, that is, understanding bike flow patterns, as this was aligned with our research project. To answer this question, the researchers identified neighborhood flows for each trip and grouped them into trip clusters. In addition, ride direction was calculated at different time intervals for stations in each cluster.

During the data pre-processing stage, the researchers eliminated trips with identical origin and destination, since these trips were not considered meaningful for flow analysis. Trips with a duration of less than one minute were also removed, as these trips may not be representative of a common use of the bike-sharing system. Finally, the dataset was divided between weekday and weekend trips as well as trips made by subscribers and non-subscribers, in order to analyze potential travel pattern differences depending on the day and user type.

The results indicated that weekend usage was much lower than weekday usage, and that weekday usage peaked at rush hours, that is, around 8am and 6pm. Moreover, it was found that there were dominant inbound flows to the center of Chicago during morning peak hours, and outbound flows from the center of Chicago during afternoon peak hours. It was also discussed that this bike flow pattern may apply to cities with similar concentric zone structures. In fact, we believe this flow pattern could also apply in Montreal, which confirmed our interest in conducting such a project.

The second study we reviewed focused on predicting BSS demand in Thessaloniki, Greece, based on 2013-2018 data (Boufidis, 2020). More specifically, the predictive model that was developed accounted for spatial and temporal data, as well as historical and forecasted weather data. In this study, depending on stations' rental density, three different time intervals for model training were considered, namely 1 hour, 2 hours and 3 hours. Once the time interval was selected, the algorithm counted rentals within this interval to train the model and predict demand. In addition, the models were optimized by using a grid search cross validation, whereby models are trained and cross-validated with different hyperparameters to identify the optimal configuration. The study showed that, although four algorithms were trained (Gradient Boosting, XGboost, Random Forests and a Neural Network), XGboost outperformed the rest by some margin.

To conclude, here are a few key points that emerged from the additional studies we reviewed:

- Temperature and hour were ranked as the most influential variables to predict demand
- Tree-based methods, especially XGboost, give best results for system-level demand prediction
- The most common performance evaluation metrics used in this context are RMSE, MSE, MAE and R^2

Section 3: Data preparation

3.1. Data sources

For this project, data was collected from three main sources.

Our first and main dataset, published by BIXI and available on their Open Data Portal, contained every bike trip taken by users from 2014 to 2020 – which accounted for 29 million rows in total. As the project went on, computational issues and general ease of use led us to subset our dataset and only use trips taken from 2017 to 2020, which still amounted to around 14 million observations. This main dataset was complemented by a second reference dataset, containing the assigned code, intersection name and geographical location (longitude and latitude) of each BIXI station through the 7 years of its existence.

Our second main source of data pertained to meteorological conditions. For this information, a few options were considered. Meteostat.net offered an API which was quite easy to use in Python; however, the lack or reliable data for the Montreal region forced us to set it aside. Data from the Government of Canada was used instead, as three stations located around the island allowed us to obtain a reasonable idea of the meteorological conditions at any given time. Specifically, these three stations were located at YUL International Airport, the McTavish pumping station near downtown Montreal and St-Hubert, nearby on the South shore.

Our final main source came from a dataset provided by the City of Montreal, which was initially destined to define "diffusion areas" for distribution of census documents. However, by using only certain variables, we were able to extract a sub-dataset indicating Montreal neighborhood ("Arrondissement") along with the corresponding longitude and latitudes. Despite its size (3201 observations), we were still able to train our classification model to a very satisfactory degree.

3.2. Data pre-processing

Both BIXI datasets were extremely well-kept (0 missing values) and required little to no cleaning. Six variables were available for the trip data; start and end date/time, start and end station (identified by a station code), duration of trip and finally, a binary variable indicating whether the user was a BIXI member or no. However, since station IDs weren't always maintained from one year to another, we manually created a new KeyID that accounted for the year and code of the stations. We were then able to merge our two datasets and populate the main dataset further with Start/End latitude and longitudes, direction (bearing in degrees) of trip and a binary variable for weekday/weekend.

Meteorological data required a bit more cleaning, since it had more than 30 variables and a certain amount of missing values. Some value types had to be changed and class variables (e.g. type of rain, type of fog) were converted to a numerical scale. The three distinct datasets (one for each station) did not all contain the same variables and a third, aggregated meteorological dataset was created. A function was built to compute interpolate weather data based on a distance-based weighted average – however, given the complexity of merging this location specific database with the BIXI data it was not used and the weather data used is simply the mean value of the observations recorded at the three stations, for most variables.

3.3. Computational issues

Due to the sheer amount of data used by our initial plans, we ran into increasingly frustrating memory errors and overall slow calculation times, which hindered progress at the start of the project. Our relative (near-complete) inexperience with Python and the machine learning ecosystem in general led us to use Google Collab as a central tool for all data manipulation and preprocessing, but we eventually had to reconsider our choice. Our team eventually moved to a Git workflow using Jupyter Notebooks, which allowed us to sidestep most issues previously encountered.

Section 4: Predicting BIXI's traffic volume and average trip duration based on weather and calendar data

4.1. Using eXtreme Gradient Boosting (XGboost)

4.1.1. Approach

XGboost, a high-performance gradient boosting algorithm, was chosen for the task because of the prevalence of tree-based methods in bike-sharing machine learning literature, as well as the proven track record of XGboost in Kaggle competitions.

The original goal of this part of the project was to fit models able to predict two variables (number of trips per hour *"Volume"* and average trip duration *"Duration"*) for three different types of time aggregation (hourly, daily, and monthly, using three different types of training data (only weather data, only calendar data, both). This would mean a lot of models – 18, to be exact – which proved to be completely unrealistic. Surprise!

First, a round of simple models was trained using the different types of training data. The model using all variables proved, as was expected, to be the best by a significant margin. The model using nothing but weather data performed poorly. The decision was made to fit only models using both available types of data.

Second, with less than a thousand daily data points, and thirty times fewer monthly data points, it was decided to concentrate our efforts on hourly predictions. This left us with two models to build – a more reasonable task to fathom, let alone execute.

The first round of tuned models (see section *4.1.3. Hyperparameter tuning*) yielded severely overfitted models, despite the usage of cross-validation and early stopping. The model was not performing horribly, but it clearly was not learning a useful generalization of the data. Action had to be taken.

4.1.2. Feature engineering

After XGboost is trained, it can return a chart representing the relative importance of each of the training data features. The models using all available data "as is" showed intriguing patterns (see top of figure 1): for instance, the variables representing dew point, wind direction and humidex, all appeared to be very important in the model, against common sense. The decision was made

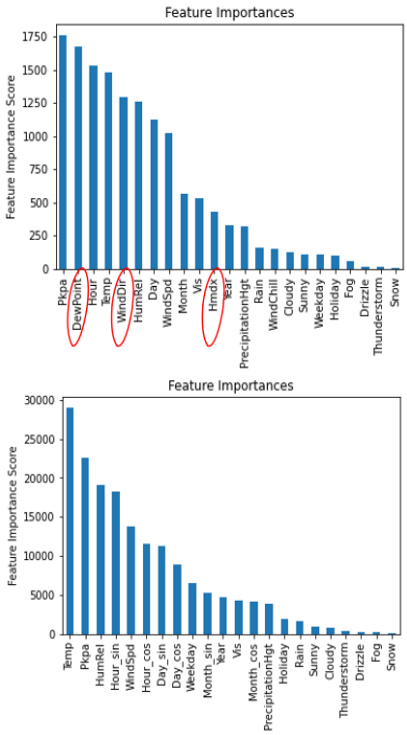


Figure 1: *Feature importance mapping for one initial model (top) and the final fitted model for "Volume" predictions (bottom)*

to remove four features from the dataset for subsequent training, namely dew point, wind speed, humidex and windchill, which is irrelevant as BIXIs are not in use during winter.

The second thing that had to be done was to encode time related information (hour, day and month) as cyclical features. The 23rd hour of the day should be nearly the same as the 1st, not be 22 units apart. A cosine transform was used.

Both these changes helped with the overfitting problem, and the features showing high importance in the final model make sense intuitively (given that pressure and humidity are excellent indicators of weather in a general sense).

4.1.3. Hyperparameter tuning

XGboost can take a lot of hyperparameters and looking for the best combination using a "brute force" grid search can be computationally intensive, to say the least. The tuning methodology suggested by Aarshay Jain in his "Complete Guide to Parameter Tuning in XGboost with codes in Python" was used. Figure 2 outlines the steps taken.

The data was split 80(train)/20(test). The hyperparameter tuning and grid search was performed only on the training set using 5-fold cross-validation and early stopping to prevent excessive overfitting. The hyperparameters yielding the models displaying the lowest average RMSE across the 5 validation folds were selected.

The ranges tested for each hyperparameter were on the conservative side, to reduce model complexity and limit the risks of overfitting. The number of points in each range was limited by computational resources.

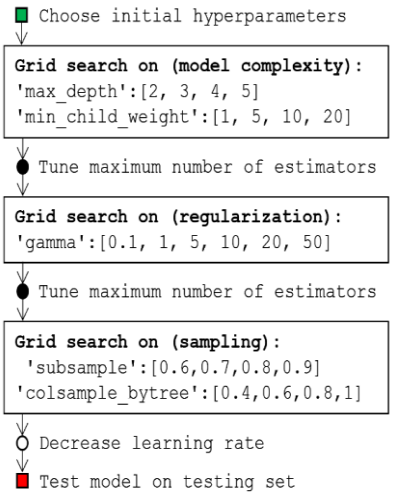


Figure 2: *XGboost hyperparameter tuning process*

4.1.4. Model performance

Table 1 lists the initial and tuned hyperparameters for both models ("Volume" and "Duration").

	Model hyperparameters							Performance	
	Maximum depth of each	Minimum child weight	Gamma (regularization parameter)	Subsample (fraction of sample used per tree)	Column subsample (fraction of variables used per tree)	Learning rate	Best number of estimators	Train RMSE	Test RMSE
Initial	3	10	1	0.8	0.8	0.1	1711	134.08	171.29
							318	55.42	57.43
Volume (Tuned)	5	5	10	0.7	1	0.01	6325	77.46	137.86
Duration (Tuned)	3	1	10	0.8	0.6	0.01	3698	54.12	57.33

Table 1: XGboost hyperparameters summary and model performance before and after tuning.

Despite the efforts made to avoid overfitting, the model predicting "Volume" overfits significantly – as we can infer from the test error which is nearly double the train error. This behaviour is not observed on the second model, predicting "Duration". The same fitting methodology was used for both. In both cases, tuning the hyperparameters reduced the test error.

4.2. Using Sci-Kit Learn's Multi-Layer Perceptron (MLP)

4.2.1. Motivation

Given the overfitting described above, the team decided to explore a second type of model to (1) see if the same behaviour and (2) compare its performance to XGboost. The team chose to use MLPRegressor(), a easy-to-use multi-layer perceptron model available in Sci-kit learn.

4.2.2. Additional data pre-processing

XGboost was able to handle missing data and unscaled values. This is not the case for this MLP.

Two variables contained missing values (precipitation height and visibility). Zeros were imputed for all missing precipitation heights, whilst imputation by the mean was used for visibility. Additionally, all variables were scaled using sci-kit learn's StandardScaler().

4.3.2. Hyperparameter selection

As with XGboost, the data was split 80(train)/20(test) and the hyperparameter selection was based on the mean cross-validation error. Each of 6 hyperparameters (number of layers, number of neurons per layer, alpha – a regularization parameter, the initial learning rate, the type of activation function and the learning rate algorithm were varied individually from a baseline model to see their effect on performance.

For the numeric parameters, the range of values exhibiting the lowest validation error were chosen (see figure 3 for examples) for further investigation in a grid search method. To limit an exponential increase in complexity, a preference was given to simpler models when choosing which range of hyperparameters to test in the grid search phase. For categorical parameters, the best one was simply chosen: consequently, the Relu activation function was deemed best for both target variables, whilst the best learning rate algorithm was "Inverse scaling" for the target variable "Volume", and "Constant" for "Duration".

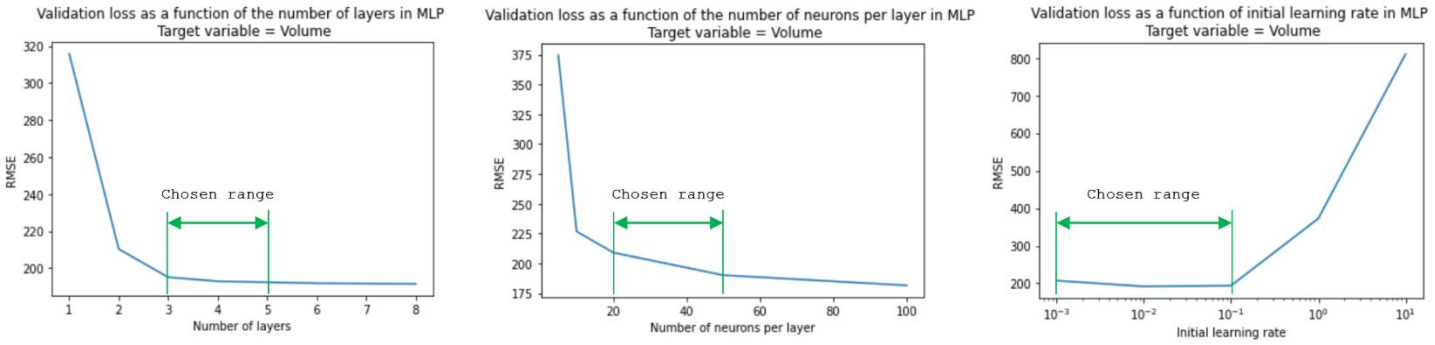


Figure 3: *Examples of the methodology used to select good hyperparameters for MLP*

The grid search allowed to test a reasonable amount of hyperparameter combinations (between 24 and 36). The methodology used does not consider the interaction between parameters and as such, it won't necessarily allow us to find the best hyperparameter combination. However, it was fast and easy and allowed us to get results quickly in the last week before the project was due.

4.3.3. Model performance

Table 2 lists the parameters and performance of the initial and tuned MLPs for both target variables.

	Model hyperparameters						Performance	
	Number of layers	Neurons per layer	Alpha (regularization parameter)	Initial learning rate	Learning rate algorithm	Activation function	Train RMSE	Test RMSE
Initial	2	20	0.0001	0.001	Constant	Relu	-	190.87
							-	61.73
Volume (Tuned)	4	50	0.1	0.01	Invscaling	Relu	146.04	159.77
Duration (Tuned)	3	100	1	0.001	Constant		55.45	60.58

Table 2: Sci-Kit Learn's MLP hyperparameters summary and model performance before and after tuning.

Contrarily to XGboost, the MLP shows no sign of overfitting, as the train and test error are on the same scale. However, the test performance can't match XGboost. It may be possible to gain a few extra points of performance with a more rigorous hyperparameter tuning process, but that a tree-based method performs better is not surprising: it is coherent with what we were expecting following our literature review.

4.3. Results

Despite showing signs of overfitting for the "Volume" target variable, XGboost is the best performing model. The MLP used could not match the performance of XGboost on either of the target variables.

To show the results, the team decided to plot the results comparing both models to real values for two time periods, chosen semi-randomly. The first period covers October 4-6, 2018, namely the end of a regular work week. The second extends from June 23-25, 2019, an example of summer holidays, covering the end of the weekend and a national holiday. The plots are displayed in *Supplementary Materials*, at the end of the report. The patterns are wildly different on both, but the differences don't seem to have an impact on model performance.

As the model predicts discrete values, one for each hour, the predictions lines should not be connected. It was connected solely to be easily legible. A markers-only plot was very difficult to interpret.

4.4. Discussion

The use of time series splitting / sliding window predicting was considered but rejected for two reasons. First, the team assumed that the target variables at any time t were independent of previous observations. The "Volume" variable represents how many *new* trips began during a given hour. It is *not* a tracking of how many bikes are on the network at any given time, and it's a network-wide aggregation, which means that limited availability at some stations should not influence the numbers. Given, the team believed it was a fair assumption to make. Second, the explicit goal of the project was to predict the target variables based on weather and calendar data exclusively. It aims to answer at questions like: "Today is Monday, it will rain in the afternoon. How many bike trips can we expect at noon?" It does not aim to find usage patterns based on the usage data itself.

The team also realized that it did not have a full week of consecutive data points available of testing, or even a month or a year. This would have been ideal to test the accuracy of the model, instead of the sparse testing points available, resulting from random train/test splitting. All the usable data that was left after model training was data from 2020, which was unusable due to the massive shift in BIXI usage patterns in the wake of Covid-19. Using data from 2014-2017 would help with improving the predictive power of the models proposed.

Section 5: Understanding BIXI flow patterns

5.1. Introduction and approach

The second objective of our project was to analyze BIXI users' inbound and outbound traffic flow by neighborhood. To do so, we used two datasets from BIXI's Open Data Portal: a *trip dataset* from 2017-2020 and *BIXI stations' location dataset.*

Our goal was to understand BIXI users' traffic flow by taking into account the start and end station codes as well as the start and end datetimes. Rather than analyzing traffic flow by station, the team deemed it would be more interesting to conduct this analysis on a neighborhood level. In order to do so, we first had to cluster the BIXI stations dataset by neighborhood. We used the following three methods to cluster BIXI stations by neighborhood: k-means, density-based clustering method (DBSCAN) and a supervised classification model.

5.2. Clustering models and results

5.2.1. K-means method (using Sci-Kit Learn's KMeans)

Firstly, we attempted to cluster the BIXI stations using an unsupervised learning technique, more specifically, the k-means method. To do so, we used the elbow method to evaluate an appropriate number of clusters. Based on Figure 4, we can consider that 3 or 4 clusters would be best suited for this analysis, which we plotted in Figure 5.

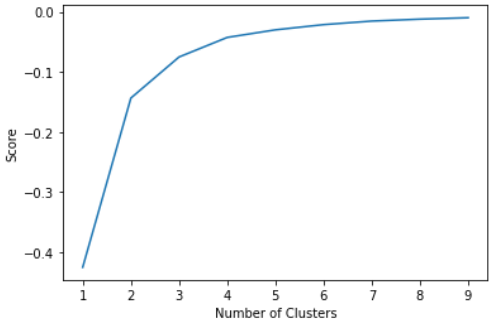


Figure 4: *Elbow Curve*

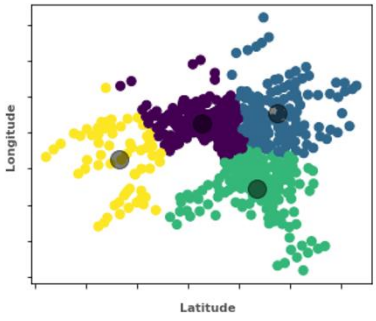


Figure 5: *K-means clustering with n_clusters = 4*

However, it was evident to us that Montreal should have more than 4 neighborhood clusters, so we attempted to increase the number of clusters, up to 20. However, the k-means method did not produce the results we were hoping for due to a major flaw in our preliminary assumption. In fact, this method usually produces interesting results if clusters have a spherical-like shape and are of similar sizes and density. Neither of these 3 things were true in our case. Figure 6, on the following page, further illustrates the shortcomings of using a k-means method within this context. Indeed, since this method is implicitly based on a pairwise Euclidean distance between data points and centroids, it would assign the point to predict to Neighborhood B. However, we can see that this data point really belongs to Neighborhood A.

As such, to counter some of the issues that stemmed from the k-means method, we tried out a different unsupervised learning technique.

5.2.2. Density-based clustering method (using Sci-Kit Learn's DBSCAN)

Secondly, we attempted to cluster the BIXI stations using another unsupervised learning technique, more specifically, the density-based clustering method (DBSCAN). This method is interesting in that it separates high density clusters from low density ones and can identify clusters of different shapes and sizes. Additionally, unlike the previous k-means method, the DBSCAN method does not require one to specify the desired number of clusters beforehand.

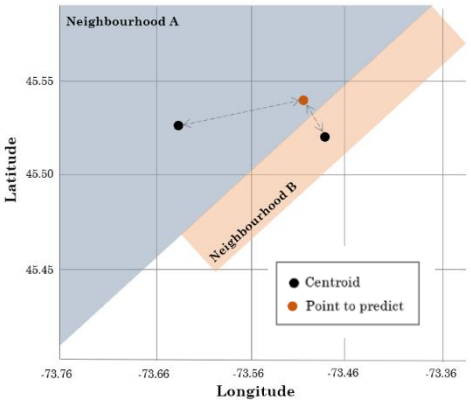


Figure 6: *Illustrative k-means clustering*

Prior to running Sci-Kit Learn's DBSCAN's algorithm, we adjusted some of the key parameters namely the epsilon (eps) and minimum points (min_points). The epsilon parameter specifies how close points should be to each other to be considered a part of a cluster – as such we set epsilon to 1km. Moreover, the min_points parameter specifies the minimum number of data points required to form a region – in our final model, we set this parameter to 3. The result is shown in Figure 7. As we can see from the Figure, nine clusters were formed. However, since the majority of BIXI stations are concentrated near the downtown area, this resulted in one large central cluster and a few sparse "satellite" clusters, which did not help us segment the central neighborhoods. Similar results were generated when reducing the epsilon to a smaller radius, which confirmed that this clustering method was not interesting given the task at hand.

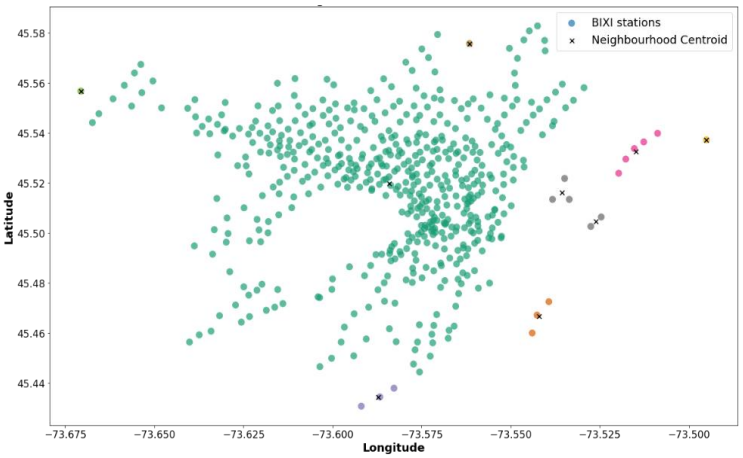


Figure 7: *BIXI neighborhood clusters and centroids*

5.2.3. Classification model (using Sci-Kit Learn's Random Forest Classifier and SVC)

Finally, a third option, and our proposed approach, was to use a supervised learning technique. More specifically, our approach was to train a classification model and subsequently use it to predict the neighborhood associated to each BIXI station. We used a dataset from the City of Montreal (*Montreal neighborhoods' dataset*) containing the latitude and longitude of 3,171 Montreal addresses as well as their associated neighborhood label.

In total, these data points were grouped in 29 distinct neighborhoods, as shown in Figure 8. When pre-processing the data, we also confirmed there were no missing values or extreme values. In addition, we ensured that each of the neighborhoods contained at least 15 observations to improve our model's performance. If that was not the case, we manually inputted additional data points using Google Maps. Only three neighborhoods contained less than 15 observations in the original data set.

Moreover, prior to training this classification model, we plotted both datasets to graphically evaluate the relevance of training such a model. As illustrated in Figure 9, the Montreal neighborhoods' data covers the majority of BIXI stations' data, with the exception of some stations on the North and South shore. In general, the team was satisfied with the overlay results – these confirmed the relevance of training such a model.

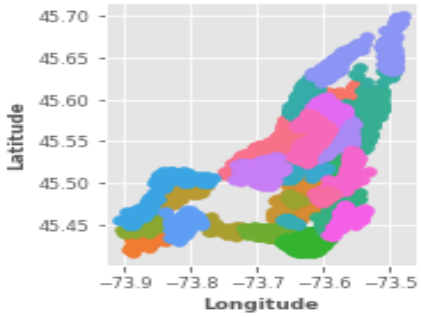


Figure 8: *Montreal neighborhoods*

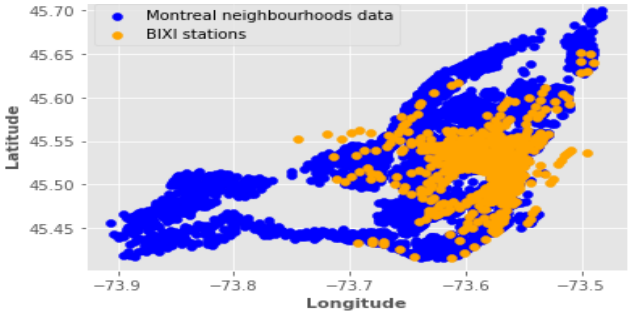


Figure 9: *Overlay of neighborhood data and BIXI stations*

Overall, our proposed approach was to train two classification models using Ski-Kit Learn's RandomForestClassifier and SVC and then select the model with the higher test set accuracy to make predictions. For both models, we encoded the latitude and longitude as the features (input) and the neighborhood code as the label (output).

The team deemed that Random Forests (RFs) could be an attractive option within the context of the task, given that they reduce the risk of overfitting compared to other methods and that they are flexible in that they work well with a variety of data. We increased the number of trees in the forest to 1000 to improve our model's accuracy, even though this required higher computational efforts. In parallel, Support Vector Machines (SVMs) were also deemed an interesting option, given that they usually work well when there is a clear margin of separation between classes and that they are best suited for smaller datasets, such as the *Montreal neighborhoods'* dataset. We adjusted some of the parameters, including setting the kernel as linear and increased the regularization parameter to c=25 in order put more importance on avoiding misclassifications.

Then, we trained both models using the parameters stated above and evaluated their performance using two methods: firstly, an 80% training and 20% testing split, and secondly, a five-fold cross-validation.

	Model hyperparameters (those different than the default values)	Test set accuracy 80% / 20% train/test	Test set accuracy 5-fold cross-validation
RF	n_estimators = 1000	93.8%	95.3%
SVM	kernel = linear, c = 25	71.3%	68.8%

Table 3: RFC and SVM test set

As shown in Table 3, the RF classification model performed much better than the SVM, with both an 80-20 train/test split and 5-fold cross-validation. In fact, the RF classification model produced 93.8% accuracy on the 20% test set while the SVM model produced 71.3% accuracy using this same split. Moreover, the RF classification model generated 95.3% accuracy with 5-fold cross validation compared to only 68.8% accuracy for the SVM model – that is, the RF model performed close to 40% better than the SVM. Given the context of the task, the team was quite satisfied with the RF model's performance. As such, this is the model we selected to predict the neighborhood associated to each BIXI station.

The output of the model is displayed in figure 10. The results will used to analyze BIXI user's flow patterns by neighborhood.

			Input		Output	
	code	name	latitude	longitude	quartier	arrondissement
0	6001	Métro Champ-de-Mars (Viger / Sanguinet)	45.510163	-73.556637	27	Ville-Marie
1	6002	Ste-Catherine / Dezery	45.539385	-73.541000	14	Mercier–Hochelaga-Maisonneuve
2	6003	Clark / Ontario	45.510824	-73.567172	12	Le Plateau-Mont-Royal
3	6004	du Champ-de-Mars / Gosford	45.509655	-73.554009	27	Ville-Marie
4	6005	Metcalfe / du Square-Dorchester	45.500208	-73.571138	27	Ville-Marie
5	6006	18e avenue / Rosemont	45.557896	-73.576529	23	Rosemont–La Petite-Patrie
6	6007	de l'Hôtel-de-Ville / Ste-Catherine	45.511660	-73.562136	27	Ville-Marie

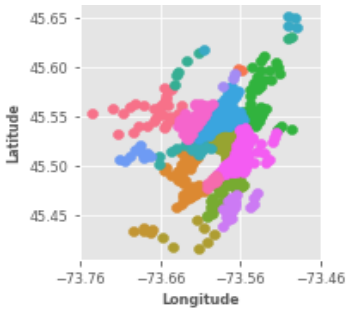


Figure 10: *BIXI stations' neighborhood model sample output (left) and plotted clusters (right)*

5.3. Analyzing biking flow between neighborhoods

Thanks to the accuracy of our classification model, we were now quite confident that almost every BIXI station within our dataset was associated to its correct neighborhood. Using this information, we were able to begin the analysis of movement of BIXI users around the city. The classification label obtained from our model allowed us to populate our dataset with 'Start_Arrondissement' and 'End_Arrondissement' variables. By aggregating the value counts of these two variables by neighborhood, and then calculating the difference between these aggregations, we could now obtain the total inbound and outbound traffic for each neighborhood.

Of course, to obtain any type of significant information, we would have to narrow down the comparisons to a specific time frame. During our literature review, we'd learned that general practice for similar analyses was to split the day into two or three-hour frames and exclude certain time frames where volume and patterns fell below a significant level. As such, we split the day into the six following time frames: 7AM to 9AM, 9AM to 1PM, 1PM to 3PM, 3PM to 6PM, 6PM to 9PM and 9PM to 11PM. Therefore, 11PM to 7AM were considered to be low-traffic hours, which was confirmed by our data.

	Count
Le Plateau-Mont-Royal TO Ville-Marie	7733
Rosemont–La Petite-Patrie TO Le Plateau-Mont-Royal	2990
Le Sud-Ouest TO Ville-Marie	2126
Le Plateau-Mont-Royal TO Rosemont–La Petite-Patrie	1934
Rosemont–La Petite-Patrie TO Ville-Marie	1930

Figure 11: *5 most popular routes for 7am-9am during regular workdays*

After a few exploratory analyses, we decided to narrow down our efforts on the weekday commuting of BIXI users, which was most likely done during the first (7-9AM) and fourth (3-6PM) time frames of our grouping. We were also interested to see whether patterns still existed in 2020 after COVID-19 hit and workers presumably worked mostly from home. At this point, we were also able to create 'route' variables, which concatenated the start and end point of a given bike trip. By aggregating bike trips for these variables, we were able to get an idea of the main axes used by BIXI users in the time frames which interested us. On the right are presented the 5 most-used routes for both time frames, along with the annual number of bike trips for weekdays only (Figures 11 and 12).

	Count
Le Plateau-Mont-Royal TO Ville-Marie	14825
Ville-Marie TO Le Plateau-Mont-Royal	11047
Rosemont–La Petite-Patrie TO Le Plateau-Mont-Royal	9648
Le Plateau-Mont-Royal TO Rosemont–La Petite-Patrie	8744
Ville-Marie TO Le Sud-Ouest	7441

Figure 12: *5 most popular routes for 3pm-6pm during regular workdays*

Following this inkling, we looked for a way to visualize these patterns. We considered a few visualization options, such as node graphs illustrating the strength of the links between each neighborhood, or others indicating the percentage of outbound trips headed to each neighborhood (including trips that started and ended within the same neighborhood). We also experimented, using the direction bearing we'd calculated for each trip, with arrow designs representing outward directions taken by the users. However, these efforts weren't conclusive.

Based on other visualisations centered around the clustering element of studies on the subject, we set about creating an interactive map using the Folium library. Two screenshots from these maps are presented below (Figures 13 and 14), illustrating the inbound and outbound traffic in central neighborhoods for our timeframes of interest. Each neighborhood is represented by a circle whose area is relative to the absolute value of net departures (meaning that certain neighborhoods might have high traffic but modest difference in arrivals/departures, resulting in small circles), while the color is determined by the principal direction of traffic (green if there are more arrivals, red for more departures).

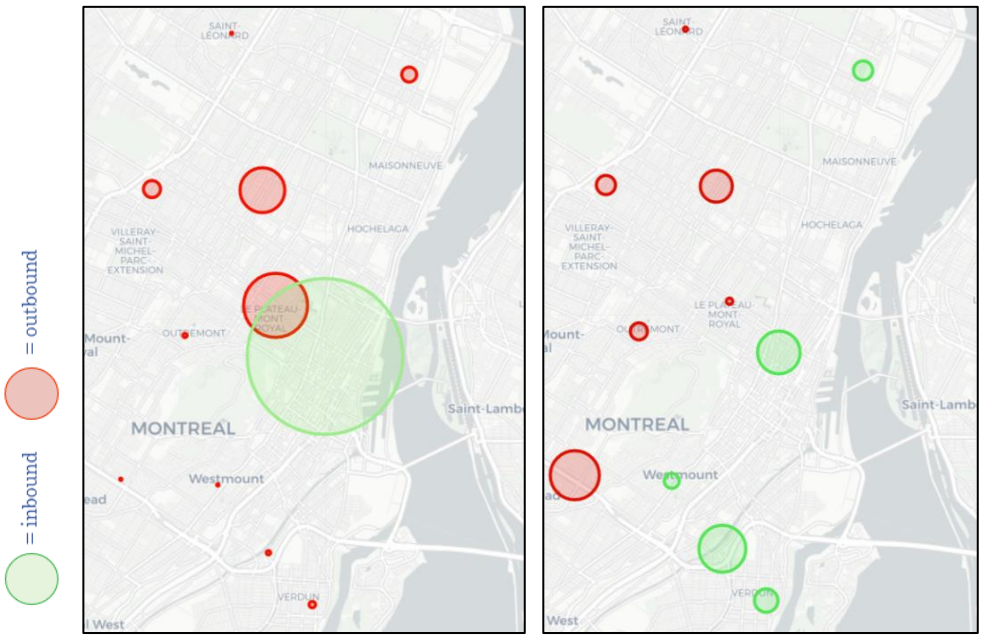


Figure 13: *Montreal biking flow for 7-9*

Figure 14: *Montreal biking flow for 6-9*

Specifically for our time frames of interest, we assessed that BIXI users tended to congregate towards the downtown area of Montreal, since it was the only one with heavily positive net departures. For the night commute, patterns weren't as obvious, but most peripheral neighborhoods such as Verdun, Hochelaga-Maisonneuve or Le Sud-Ouest all had much more arrivals than departures, leading us to suspect that BIXI users probably used the biking system to head back home. We also noticed that the most popular routes were usually reversed from the morning commute's (e.g. Ville-Marie to Plateau Mont-Royal, Plateau Mont-Royal to Rosemont-Petite-Patrie, etc.) Finally, we also highlighted a strong outbound flow from Côte-des-Neiges at the end of the day, headed towards Plateau and Downtown.

Section 6: Conclusions

6.1. Key insights

Acquired knowledge and tears of rage, distress, woe, and other violently tragic emotions were both plentiful during the realization of this project. Despite the hurdles, the team has reached the Holy Grail: interesting results to report – notwithstanding the real treasure any adventure brings on its way towards a long-dreamed destination: friends. Now, back to the results:

- With nRMSE:s south of 15% and 8% respectively, the team has demonstrated that it's possible to predict how many BIXI trips will begin during any given hour of the biking season, as well as

the average duration of these trips, based only on easily accessible weather and calendar data, with reasonable accuracy. XGboost, a tree-based gradient boosting algorithm, offered the most accurate predictions for the task, outshining Sci-Kit Learn's MLP, as was expected.

- A random Forest Classifier was the most accurate model to cluster BIXI stations to their respective neighborhoods, surpassing K-means, DBSCAN and a SVM. The model was accurate to 95% on the validation set used.
- The results from the neighborhood classification were used to illustrate graphically the bike flows between Montreal neighborhoods under different conditions (morning commute, evening commute). As was expected, flow patterns varied wildly based on the time windows analyzed.

6.2. Future directions

The team ran into time and computational constraints and suggest the following avenues for future development. For the volume and duration training model:

- As discussed in the associated section, it would be interesting to use more the available data from 2014-2017 to train the model to see if it improves performance and reduces overfitting. It will also be interesting to see if biking patterns will come back to what they were like in 2019 after the end of the pandemic, or whether some of the changes are irreversible. Additional data would also allow to test how a similar is able to perform on daily predictions rather than hourly.
- It would be interesting to compare the results obtained with bike patterns observed in cities other than Montreal, to see if weather and calendar data has the same influence everywhere, or if it is region/city specific.
- Comparing the performance of the model in different situations (holidays vs workdays, rain vs no rain) could help to understand the workings of the model and discover hidden bias.
- The weather data used is averaged over three stations forming a triangle around Montreal. A function to interpolate weather more precisely was developed, but left unused, and it would be interesting to see if using this module could improve the model's predictive power.
- Demand ("Volume") was predicted on a network-wide basis. In theory, it could be modified to make predictions on a station-to-station basis, though this would imply a whole new set of challenges, as station by station volume is dependant on station specific characteristics not taken into account in the current model.

And for the traffic flow analysis:

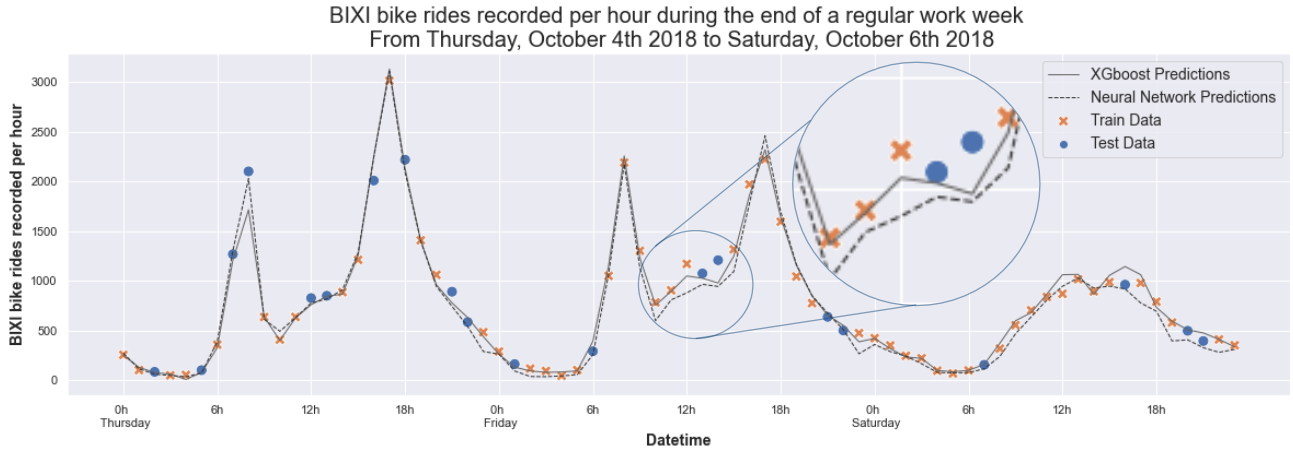
- Explore whether other clustering methods not necessarily tied to neighborhoods could provide more interesting results.
- Possibility to deepen analysis by analyzing specific routes, allowing the researchers to identify most-used cycling axes. This could provide useful information for city planners and officials alike. Similarly, visualising these routes in a meaningful way could help determine whether current biking lanes could use updates or outright relocation.
- Apply prediction methods on a station-to station basis in order to distribute optimally bikes across stations, by prioritizing those with pronounced outbound patterns.

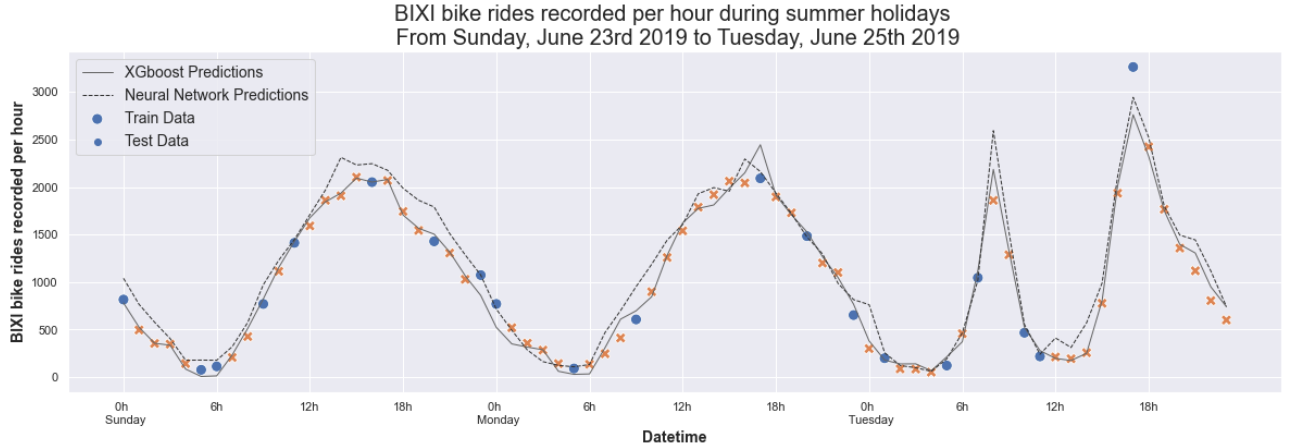
Supplementary materials

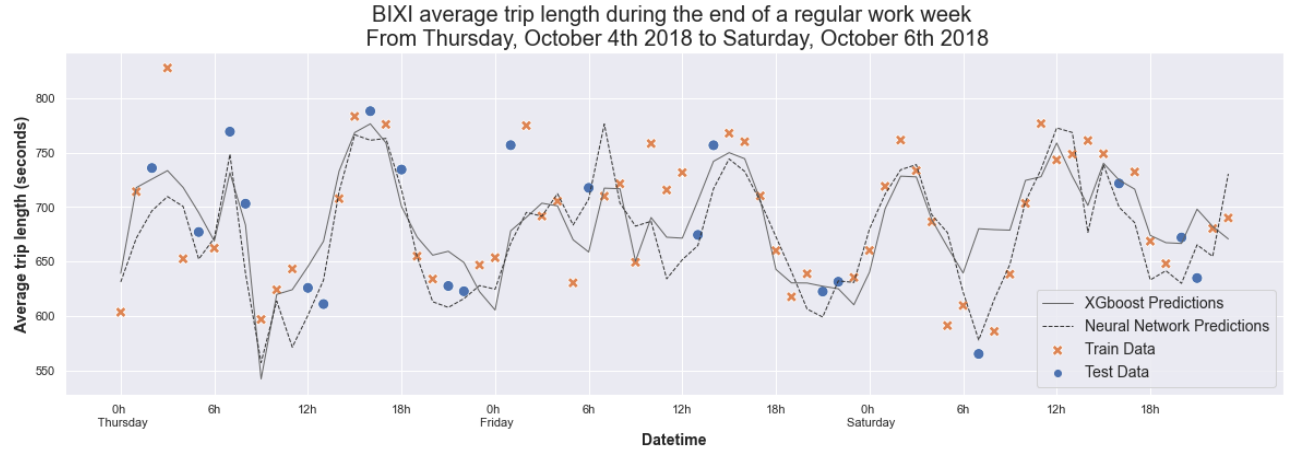
Plots for Section 4

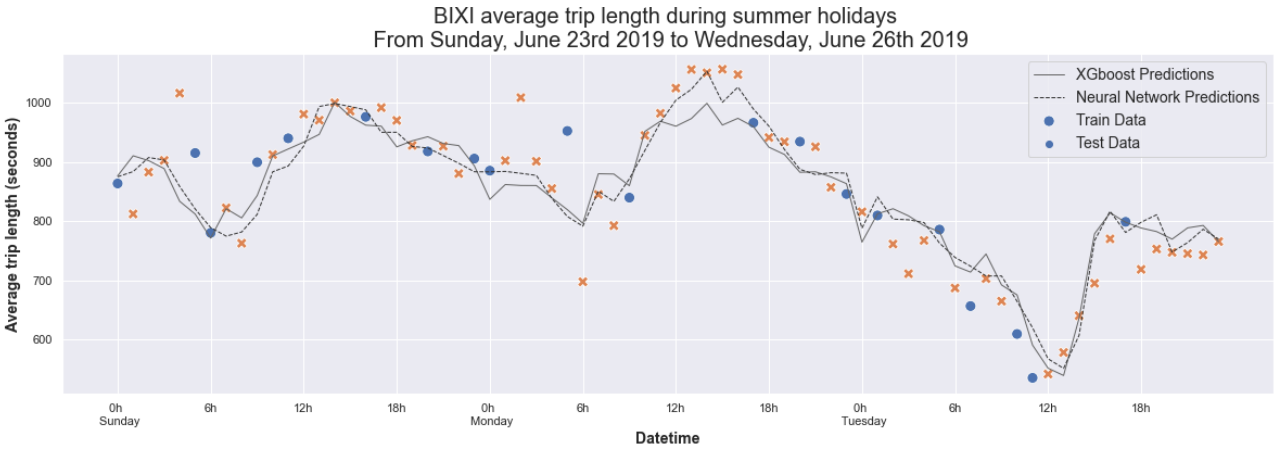
The first two plots show the results for the first target variable "Trips per hour", whilst the last two display results for the second target variable, "Average Trip Duration".

In the first plot, an example of overfitting is shown. The predicted values for XGboost (solid line) pass straight through the orange train points but miss the blue test points. The MLP does not exhibit this behaviour.









Work distribution

As requested, here is a brief overview of the tasks completed by each team member:

Gabriella:

- Initial Colab set up
- Write-up – Literature Review
- *Sections 5.1. and 5.2.* – Clustering models (K-means, DBSCAN and classification models)
- A lot of Googling

Simon:

- BIXI datasets pre-processing
- *Section 5.3.* – Analysis of biking flows
- Write-up - *Sections 1 & 3*
- Asking a lot of questions to a lot of people

William:

- Weather data pre-processing
- *Section 4* – fitting XGboost and Sci-Kit Learn's MLP in order to predict trips per hour and average trip duration.
- GIT setup
- Report layout
- A record breaking 34 tabs of Stack Overflow tabs opened at the same time

References

Albuquerque, V., Sales Dias, M., & Bacao, F. (2021). Machine Learning Approaches to Bike-Sharing Systems: A Systematic Literature Review. *ISPRS International Journal of Geo-Information*, *10*(2), 62.

Boufidis, N., Nikiforiadis, A., Chrysostomou, K., & Aifadopoulou, G. (2020). Development of a station-level demand prediction and visualization tool to support bike-sharing systems' operators. *Transportation Research Procedia*, *47*, 51-58.

C-M, Gregoire (2019) Understanding BIXI Commuters: An Analysis of Montreal's Bike Share System in Python. *Towards Data Science*. Available online : <https://towardsdatascience.com/understanding-BIXI-commuters-an-analysis-of-montreals-bike-share-system-in-python-cb34de0e2304> (accessed on April 6th, 2021).

Goodman, A., & Cheshire, J. (2014). Inequalities in the London bicycle sharing system revisited: impacts of extending the scheme to poorer areas but then doubling prices. Journal of Transport Geography, 41, 272-279.

Hsu, C-L. (2018). Clustering on New York City Bike Data Set. *Github Pages*. Available online : <https://chih-ling-hsu.github.io/2018/01/02/clustering-python/> (accessed on April 1st, 2021).

Jain, A. (2016). Complete Guide to Parameter Tuning in XGboost with codes in Python. *Analytics Vidhya*. Available online : <https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-XGboost-with-codes-python/> (accessed on April 2nd, 2021).

Lin, L., He, Z., & Peeta, S. (2018). Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach. *Transportation Research Part C: Emerging Technologies*, *97*, 258-276.

Meddin, R.; DeMaio, P.J. The Meddin Bike-Sharing World Map. Google Maps. 2020. Available online: <https://bikesharingworldmap.com/#/all/2.3/-1.57/33.92/> (accessed on March 30th, 2021).

VE, S., & Cho, Y. (2020). A rule-based model for Seoul Bike sharing demand prediction using weather data. *European Journal of Remote Sensing*, *53*(sup1), 166-183.

Wright, R. (2014). Public Bicycle-Sharing Company wobbles into bankruptcy. *Financial Times.*

Zhou, X. (2015). Understanding spatiotemporal patterns of biking behavior by analyzing massive bike sharing data in Chicago. *PloS one*, *10*(10), e0137922.