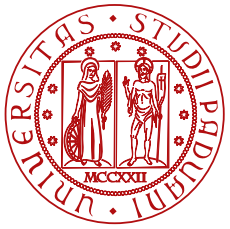

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Computer Engineering

Boosting predictions of free-floating bike sharing systems with spatial clustering techniques

Master Candidate

Gabriele Zanatta

Student ID 2026728

Supervisor

Prof. Matteo Comin

University of Padova

Academic Year
2022/2023

Abstract

There exist various types of public transports in the cities, starting from trains, buses, trams to the newest rental bikes and electric scooters. The use of the latter is even more helpful to the ecosystem, given that no carbon dioxide is produced during use and they have a lot of potential to improve the city's environment, making it eco-sustainable and smart. The stationless bike sharing system, called floating, is replacing the one that uses fixed stations, called docking, for a matter of ease of use and freedom left to the end user. A bike can be left in any place you choose, even a few meters from the destination, without having to look for a mandatory delivery area. One of the remaining problems is the availability of means of transportation in case of need. These are distributed through the use of vans that collect and place them in defined areas, so that they can be found available and sorted as needed.

The aim of this thesis is to identify areas of use of this type of bikes through various spatial clustering techniques, using the city of Padua (Italy) as center of the study. In addition, a rectangle division of the analysis area was used to go deeper and further highlight differences. Subsequently, two prediction methodologies were implemented to identify the number of vehicles that should be made available in a specific region and in a specific time slot. ARIMA and XGBoost were used as prediction technologies, which allow in a reliable and precise way to have a real value of means of transportation that will be used in that time period. The various predictions were then compared using statistical methods to analyze the best clustering method and the number of ideal regions to create. The application of this method of analysis, study and prediction can be used to make the distribution of bicycles more optimized in order to make this type of transport more sustainable for the environment.

Contents

List of Figures

List of Tables

List of Code Snippets

1 Introduction

Starting from the ancient river boats, passing through stagecoaches, steam trains, trams, until the newer buses, high-speed trains, planes, subways, the public transports have gradually evolved giving access to fast and safe travel systems to millions of people. Now, the goal is to make these vehicles eco-sustainable, so as to affect the environment as little as possible. There are already various cities that implemented the most recent ones, such as: electric buses, bike sharing and electric scooters. This thesis will concentrate on the bike sharing system, because is the main travel method used in the city where the University is placed, that is Padua (Italy).

Bike sharing is a system in which bicycles are made available to individuals for short-term rentals. The aim of bike sharing is to provide an easy and affordable transportation option to the public one, promoting sustainability and reducing the number of cars on the roads. It has a relatively long history, dating back to the 1960s in Europe, when free bikes were made available for public use in Amsterdam. However, it was not until the 1990s and early 2000s that the modern era of bike sharing began, with the introduction of technology-driven, commercial systems. One of the earliest bike sharing systems was launched in Copenhagen, Denmark in 1995. The system, called Bycyklen, used smart cards and computerized docking stations to manage the distribution of bikes. Since then, bike sharing has continued to grow in popularity around the world, with new systems being launched in cities of all sizes. Two common solutions are docking and floating, each with their own unique features and benefits. Docking bike sharing systems involve bicycle being housed in a network of docking

stations throughout a city. Users can rent a bike from any docking station using a membership card, credit card or smartphone app. Once the user has paid the rental fee, he can unlock the bike and then ride it. The return is done by placing the bicycle on any docking station near to the destination. The cycle is then locked until another user rents it. Floating bike sharing systems, on the other hand, do not require docking stations. Bikes are distributed throughout the city and can be found using a smartphone app or website, which shows the user the location of available bicycles. Users rent it using the app, and then ride to their destination. Once they have arrived, the user can leave the bike locked in a safe and legal location, ready for the next user to find and rent. This last system is the one used in Padua and it is highly flexible, as bikes can be rented and returned anywhere within the designated service area. So, the users can easily travel to locations that are not well-served by docking stations. Floating systems can also be more cost-effective than docking systems, as they do not require the installation or maintenance of stations. However, the distribution of bikes can be more challenging, as bicycle can become clustered in certain areas or be difficult to find.

There is a big problem with the floating bike sharing system, that is to say: the dynamics of human mobility often lead to inevitable bike supply and demand imbalance. The aim of this thesis is to identify areas of use of this type of bike through various spatial clustering techniques, using Padua (Italy) and Montreal (Canada) as the study cities. Padua, as said before, has a floating bike sharing system instead of Montreal that has a docking system. We used the Canadian city to compare the clustering methods and to verify the reliability of the predictions. Identifying different regions in a city permits us to: process data independently in each zone and provide a method for predicting the number of bikes that have to be available in a specific time range in a region. There are lots of researches in the literature about this problem. As an example, in a paper, the authors concentrate their research on the trucks route optimization[8], using a graph approach, including the truck drivers' home in the dataset. The aim of their work is to maximize the sum of the following three parts:

- Optimizing the repositioning system.
- Lowering the workload of all truck drivers.
- Lowering the travel cost of all trucks.

For doing this, they divided their workflow into three steps:

1. Using travel clustering to identify virtual stations.
2. Estimating repositioning targets of all stations.
3. Planning routes and calculating workload of all truck drivers

Another research analyzed the flow characteristics of dockless shared bikes that are expounded through the time series location data and after they used a model called DestiFlow[6] to describe the spatio-temporal flow of urban dockless shared bikes. This flow can be divided into 3 parts. In the first one, they used a POI-based clustering, with POI that stands for point of interest and can be a house, shopping mall, station, ecc... To find each region containing a POI, they processed the dataset with a K-Means based clustering. After, they gave to every cluster a probability that indicates how much a bike starting from a point can arrive in the selected region. Obviously, the more active an aggregation area is, the more popular the aggregation area is, and the greater the probability of departure and arrival from there is. For the last and third step, they constructed a formula to estimate the number of flows in a time window $[t, t + \Delta_t)$:

$$|flow_s^{\Delta_t}| = C * \Phi(t, t + \Delta_t) \quad with \quad \Phi(t, t + \Delta_t) = \int_t^{t+\Delta_t} \phi(x)dx$$

Where $\phi(x)$ is the one-dimensional Gaussian mixture model and C is the total number of flow. Moreover, there are scientific papers that lean on convolutional neural networks. One of this paper [13], divides a city into a grid, with every cell representing a region. With the use of origin-destination trips between areas, they define a Spatial-Temporal Memory Newtork divided into 3 independent Convolutional LSTM that are used to extract the temporal dependency based on spatial relationships from a historical sequence. In their network they capture the temporal dependency from three points of view: closeness, period, and trend. Closeness captures the temporal dependency of bicycle usage in the past one day, period captures the bicycle usage in the past a couple of days and trend aims to capture the long-term dependency on a weekly scale. After that, the 3 convolutional LSTM have to be fused into the final prediction and they used 3 different typologies, such that: weighted element-wise addition, concatenation and weighted concatenation.

In this thesis the Padua and Montreal datasets will be divided with different types of clustering that give various results as output, a few with a more homogeneous distribution of bikes and others that concentrate in the creation of more

areas in the outskirts zones. This approach will be done because the scope of this thesis is to predict the number of bikes that have to be available in a certain area of Padua in a specific range of time. Then, to be able to forecast this value in every area of the city, the entire dataset has to be divided into clusters and a prediction algorithm has to be applied to each of them. For the forecasting process, two types of forecasting methods have been implemented, ARIMA and XGBoost. The first one is one of the most used techniques and it provides reliable results in the most common cases. The second one gained significant favor in the last few years as a result of boosting the processing times and results in forecasting challenges. The implementation of it was done to test its efficiency, accuracy and feasibility comparing the results with the ARIMA ones. They use different approaches to the forecasting problem and we will observe in a more detailed way further on.

2

Clustering and prediction techniques

Observing different datasets from different cities that have a floating bike sharing system, it is simple to notice a region based partitioning of the drop and retrieval locations. There are zones with a lot of bikes, one near to each other, and others with a very few of them and only in certain hours. The best way to analyze these datasets is to divide them into regions, using specific clustering techniques. We have chosen this approach because the aim of this thesis is to predict the number of bikes that have to be available in a certain zone of Padua in a specific range of time. Then, to be able to forecast this value in an area of the city, we have to divide the entire dataset into clusters and apply a forecasting algorithm to each of them.

2.1 Clustering

Clustering is a type of unsupervised machine learning algorithm[12] that is used to group together similar data points based on their intrinsic characteristics. In clustering, the goal is to identify patterns and structures in the data that may not be immediately apparent and to group data points that are similar into clusters. Clustering is useful in a wide range of applications, including data mining, image processing, social network analysis and market segmentation. For example, in bio-informatics, clustering can be used to identify groups of genes or proteins that are co-regulated or functionally related, which can help to understand the underlying mechanisms of diseases.

There are various methodologies for clustering, each with its own strengths

and weaknesses. Here are some of the most common clustering methodologies:

- Hierarchical clustering[17]: the data points are grouped into a tree-like hierarchy of clusters, either by recursively dividing the data into smaller clusters (Divisive clustering) or by merging clusters until a desired number of clusters is reached (Agglomerative clustering).
- Density-based clustering: clusters are formed based on the density of the data points in a particular region, rather than the distance between the data points. One popular algorithm for density-based clustering is DBSCAN[5].
- Model-based clustering: clusters are formed by fitting a statistical model to the data, such as a Gaussian mixture model (GMM[14]) or a Hidden Markov model[18]. Model-based clustering can be useful in identifying complex patterns in the data.
- Graph-based clustering[10]: the data points are represented as nodes in a graph, and the clusters are identified as connected sub-graphs. Graph-based clustering can be useful in identifying clusters in high-dimensional data, where distance-based methods may not be effective.

Clustering is a powerful tool for discovering patterns and relationships in data and can be used in a wide range of applications. The choice of clustering method depends on the characteristics of the data and the specific problem.

2.1.1 Squares

The first technique is not really a clustering method. It simply divides the area of interest into squares with the same size. It can be applied in every dataset. In our case, the only mandatory note for the use is that a perfect square as clusters' number is needed, in a way that there is a big square divided into smaller ones of the same dimension. The algorithm calculates first the ends of the main square, taking the 4 points that are on the edges of the area of interest, so there is one object for each cardinal point. Successively, it processes the length and the width of the smaller squares, simply dividing the distance from the northernmost and the southernmost points and the distance from the easternmost and the westernmost points by the root of the number of desired clusters. This is not a very good clustering technique as it will be seen later, because it can create a large number of nearly empty clusters and ones with a very high number of objects.

During the development of the thesis has been realized that the statistical evaluations on the prediction of the values using this type of clustering returned

very high values, precisely because there were many semi-empty clusters and therefore it was easy to predict the number of bikes in that region.

2.1.2 K-Means

K-Means[11] is the most famous and used clustering method and it is applied in search engines, astronomy, market segmentation and statistics. Its goal is to split whole data into k, fixed a priori, different clusters. The algorithm begins by randomly selecting k initial centroids, which are the center points of the clusters. Each data point is then assigned to the closest centroid, based on its Euclidean distance to each center. Once all data points have been assigned to their nearest centroid, the algorithm computes the mean of the data points assigned to each cluster and moves the centroid to this new mean location, called barycenter. This process is repeated until the centroids no longer move or a specified number of iterations have been completed.

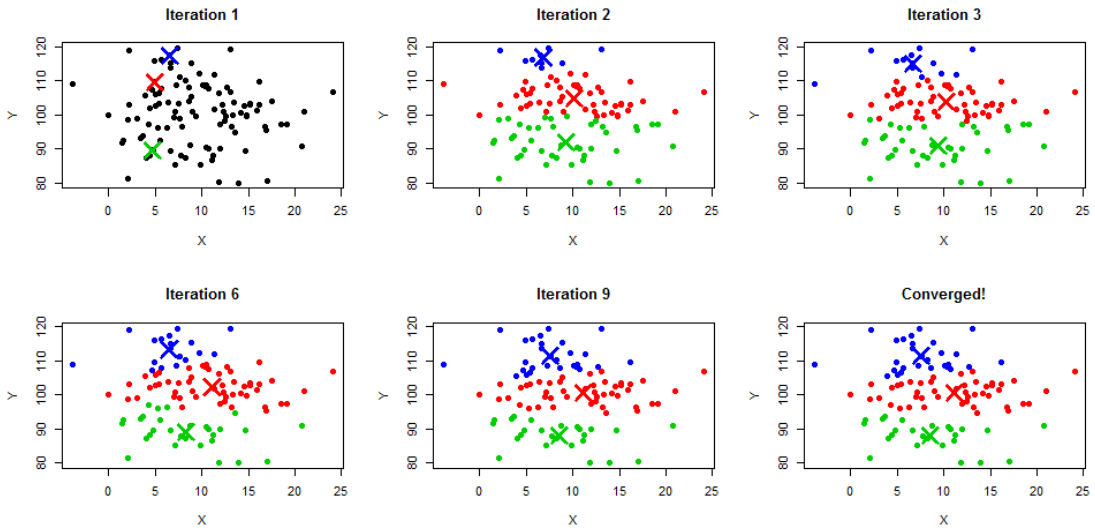

Figure 2.1: Example of K-Means iterations

The real aim of this algorithm is to minimize a squared error function, also known as the Within-Cluster Sum of Squares (WCSS). The WCSS is used as the objective function of the algorithm, and is a measure of how well the clusters represent the data. The smaller the WCSS, the better the clustering result. So,

the objective function is:

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2$$

Where μ is the centroid of points in S_i.

K-means clustering has several advantages, including its simplicity, speed, and ability to handle large datasets. However, it also has some limitations. For example, it requires the number of clusters to be specified beforehand, which can be difficult or impossible in some datasets. It is also sensitive to the initial choice of centroids and can converge to a sub-optimal solution if the initial centroids are poorly chosen. Additionally, K-means is only able to identify spherical clusters and may not perform well on datasets with non-spherical clusters or clusters of varying size and density.

2.1.3 GMM

GMM (Gaussian Mixture Model)[14] clustering is a statistical method for clustering data points based on their probability distribution. GMM assumes that the data points are generated from a mixture of Gaussian distributions, where each Gaussian distribution represents a different cluster. The goal of GMM clustering is to estimate the parameters of these Gaussian distributions, such as their means and standard deviations, and to assign each data point to the most likely cluster.

The GMM clustering algorithm begins by randomly initializing the means and standard deviations of the Gaussian distributions. It then iteratively updates these parameters using the Expectation-Maximization (EM) algorithm, until the algorithm converges to a stable solution. The EM algorithm consists of two steps: the E-step and the M-step. In the E-step, the algorithm estimates the probability of each data point belonging to each cluster, given the current parameters of the Gaussian distributions. This is done using Bayes' theorem, which calculates the posterior probability of each cluster given the observed data. In other words, the E-step calculates the degree of membership of each data point to each cluster. In the M-step, the algorithm updates the parameters of the Gaussian distributions based on the probabilities calculated in the E-step. Specifically, the means and standard deviations of the Gaussian distributions are updated based on the

weighted average of the data points assigned to each cluster, where the weights are the probabilities calculated in the E-step. The algorithm iterates between the E-step and the M-step until the parameters converge to a stable solution. At this point, the algorithm assigns each data point to the most likely cluster based on the posterior probabilities calculated in the E-step.

In this thesis, the GMM clustering with K-means initializer has been used. It is a variant of GMM clustering that uses the K-means algorithm to initialize the means of the Gaussian distributions. The GMM clustering algorithm with K-means initializer follows a similar process to the standard GMM clustering algorithm. However, instead of randomly initializing the means of the Gaussian distributions, the means are initialized using the means of the K-means clusters. This can help to improve the convergence of the algorithm and avoid the problem of getting stuck in local optima. The algorithm for GMM follows this steps:

1. Initialize the number of clusters K and the convergence criteria.
2. Use the K-means algorithm to partition the data into K clusters and calculate the means of each cluster.
3. Use the means of the K-means clusters as the initial means of the Gaussian distributions.
4. Calculate the covariance matrices and mixing coefficients for each Gaussian distribution.
5. Iterate the Expectation-Maximization (EM) algorithm until convergence is achieved.
6. Assign each data point to the most likely cluster based on the posterior probabilities.

One advantage of GMM clustering with K-means initializer is that it can be faster and more robust than standard GMM clustering, especially for large datasets. This is because the K-means algorithm can quickly partition the data into K clusters, which can provide a good starting point for the GMM algorithm. However, GMM clustering with K-means initializer also has some limitations. For example, the quality of the clustering result can be sensitive to the choice of K, and the algorithm may not work well if the data distribution is not well-suited to the Gaussian mixture model. Additionally, GMM clustering with K-means initializer can be sensitive to outliers, which can affect the initial clustering result and the convergence of the algorithm.

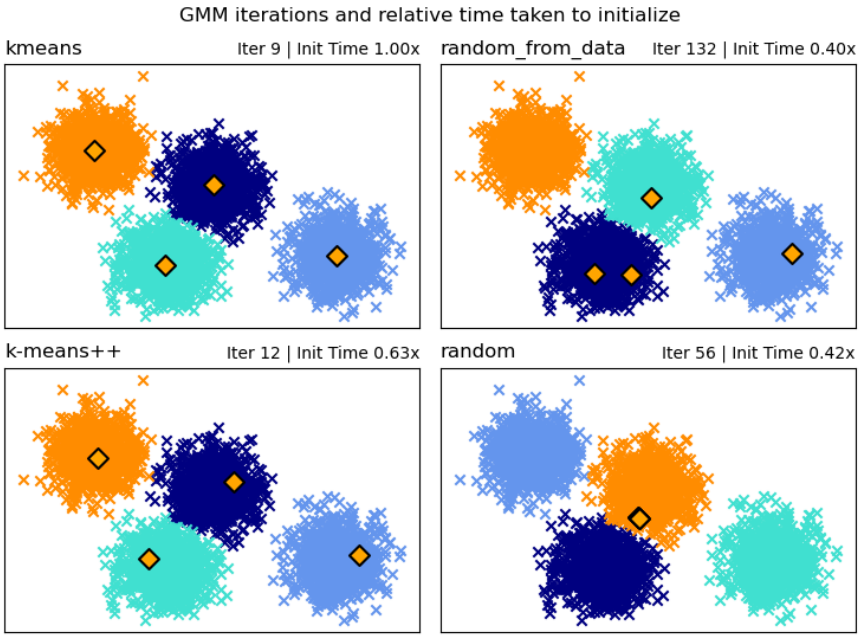

Figure 2.2: Example of GMM with different initializer

2.1.4 Mean Shift

Mean Shift[2] clustering is a non-parametric clustering algorithm that identifies the modes of a density function and uses them to cluster the data. The algorithm works by iteratively shifting each object towards areas of high density in the data, until the points converge to a final set of cluster centers. The Mean Shift clustering algorithm can be summarized as follows:

1. Choose a bandwidth h, which controls the size of the window used to estimate the density of the data.
2. For each data point, define a window around the point with radius h.
3. Estimate the density of the data within the window using a kernel function, such as the Gaussian kernel.
4. Calculate the mean shift vector for the data point by taking the weighted average of the data points within the window, where the weights are determined by the kernel function.
5. Shift the data point towards the mean shift vector.
6. Repeat steps 2-5 until the data points converge to a final set of cluster centers, which define the clusters in the data.

One advantage of Mean Shift clustering is that it can handle datasets with irregular shapes or sizes of clusters, as the clusters are defined based on the local density of the data. Additionally, the algorithm is able to automatically determine the number of clusters in the data, as the algorithm converges to a set of cluster centers. However, the algorithm can be sensitive to the choice of kernel bandwidth, as a bandwidth that is too small can result in overfitting, while a bandwidth that is too large can result in over-smoothing. Additionally, the algorithm can be computationally expensive for large datasets, as the density estimates and cluster center updates must be calculated for each data point. Overall, Mean Shift clustering is a powerful method for discovering the structure of data, and can be useful in a wide range of applications such as image segmentation, object tracking, and anomaly detection.

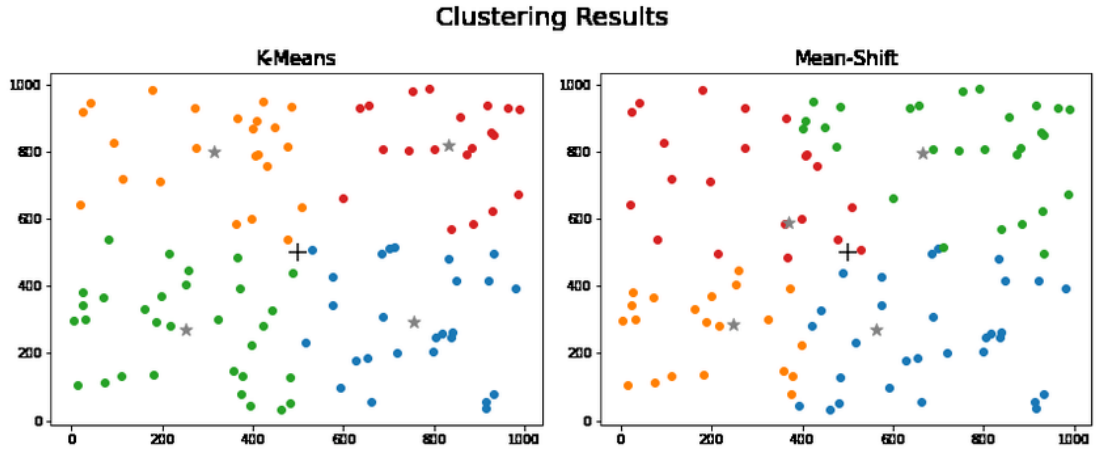

Figure 2.3: Comparison between Mean Shift and K-Means

2.1.5 BIRCH

BIRCH[19] (Balanced Iterative Reducing and Clustering using Hierarchies) is a hierarchical clustering algorithm that is designed to be scalable to large datasets. It is a memory-efficient method that uses a tree-based data structure to perform clustering. The algorithm consists of three main steps. In the first step, it constructs a tree-based data structure to represent the data, called the Clustering Feature Tree (CFT). The CFT is a hierarchical structure that recursively partitions the data into smaller clusters, with the root node representing the entire dataset. In the second step, BIRCH performs a clustering pass through the CFT to identify the final clusters. The algorithm uses a clustering criterion based on the radius

and diameter of the clusters to merge the leaves of the CFT into larger clusters. The resulting clusters are then represented by their centroids, which are updated to reflect the new members in each cluster. In the final step, BIRCH optionally applies a traditional clustering algorithm, such as K-means, to the centroids identified in the second step. This step is intended to improve the clustering accuracy by refining the cluster assignments.

One of the key advantages of BIRCH is its ability to handle large datasets efficiently. The algorithm has a low memory footprint, as it only requires the storage of the CFT and the centroids of the final clusters. Additionally, it is able to identify clusters with arbitrary shapes and sizes, as it uses a clustering criterion that is based on the diameter and radius of the clusters. However, one limitation of BIRCH is that it assumes that the clusters are roughly spherical and have similar densities. This may not be appropriate for datasets with clusters that have non-spherical shapes or significantly different densities. Additionally, BIRCH may not perform as well as other clustering algorithms, such as DBSCAN, in datasets with noise or outliers.

2.2 Predictions

For comparing the results of the clustering techniques and for getting a value of bikes in a cluster in a specific time range that can be used in the real world, two time series prediction methods were used. The time series prediction is the task of forecasting future values of a sequence of observations, where the order and timing of the observations are important. Time series prediction is a fundamental problem in many domains, such as finance, economics, weather forecasting, and industrial process control, among others. Accurate time series prediction can be valuable in decision-making, planning, and risk management. One of the most common approaches to time series prediction is to use statistical methods such as ARIMA (Auto-Regressive Integrated Moving Average) and its variants. ARIMA models attempt to capture the auto-correlation of the time series data and use it to make predictions. However, ARIMA models can be limited in their ability to capture more complex patterns, such as seasonality or non-linearity. Another approach to time series prediction is to use machine learning methods, such as neural networks and decision trees. These methods can be used to model the nonlinear relationships between the inputs and the outputs of the time series, as well as capture the effects of external factors. Neural networks,

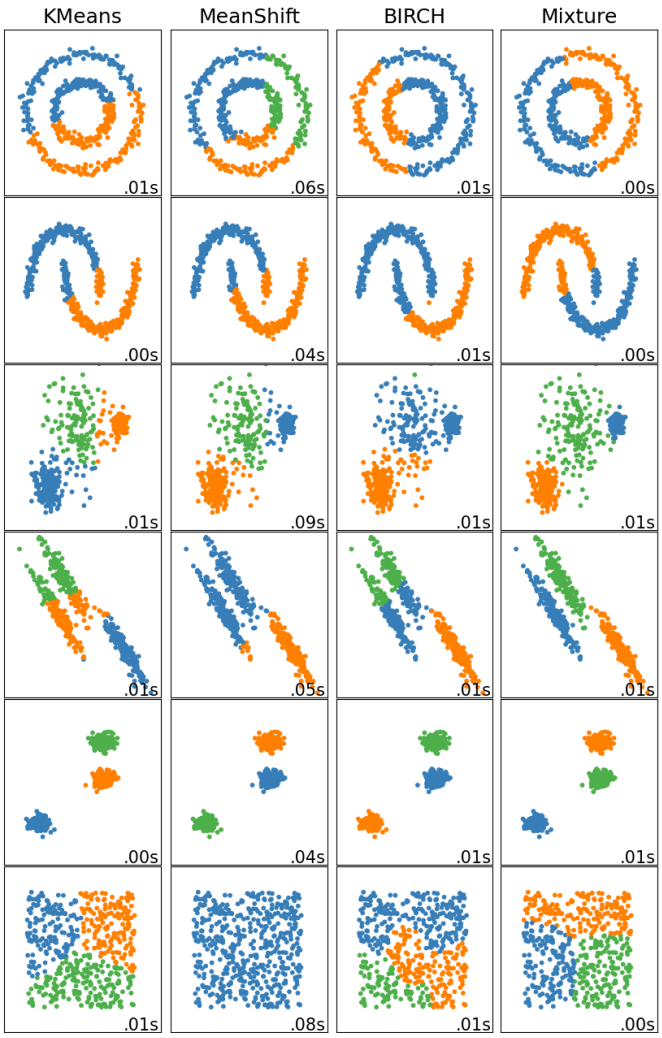

Figure 2.4: Comparison between different clustering techniques

in particular, have been shown to be effective in handling complex time series data, such as image or audio signals. For this thesis, ARIMA has been chosen as entry point in the forecasting world because is one of the most used techniques for forecasting and it provides reliable results in the most common cases. Furthermore, there exists a large community of users that has implemented it in various way and this helped us to setup and find the best values for the model. Afterwards, XGBoost was introduced, that gained significant favor in the last few years as a result of helping individuals and teams win virtually every Kaggle structured data competition. It was implemented to test its efficiency, accuracy and feasibility comparing its results with the ARIMA ones. They use different approaches to the forecasting problem and it will get to more of that in the next

chapters.

2.2.1 ARIMA

ARIMA (AutoRegressive Integrated Moving Average)[7] model is a popular and widely-used statistical techniques for time series analysis and forecasting. ARIMA models are commonly used in many fields, including finance, economics, engineering, and environmental science, among others. ARIMA models attempt to capture the auto-correlation in the time series data, which means the correlation of the current observation with past observations, and use it to make predictions. ARIMA model has three components: the Auto-regressive (AR) component, the Integrated (I) component, and the Moving Average (MA) component. The AR component is responsible for modeling the auto-correlation in the data, while the MA component is responsible for modeling the moving average of the error terms. The I component is used to make the time series data stationary, which means that the statistical properties of the data remain constant over time. The AR component of the ARIMA model is defined as a linear regression of the current observation on the previous observations, or the lagged values of the time series. The order of the AR component, represented by p, specifies the number of lagged values used in the regression. The MA component of the ARIMA model is defined as a linear combination of the previous error terms, which are the differences between the observed values and the predicted values. The order of the MA component, represented by q, specifies the number of previous error terms used in the combination. The I component of the ARIMA model is used to make the time series data stationary. Stationary data has a constant mean, variance, and auto-correlation over time. The order of the I component, represented by d, specifies the number of times the data needs to be differenced to achieve stationarity. Differencing means taking the difference between consecutive observations.

The parameters of the ARIMA model, including p, d, and q, can be estimated using various methods, such as maximum likelihood estimation or least squares estimation. Once the parameters are estimated, the ARIMA model can be used to make predictions of future values of the time series. For each of these, the higher the order, the more complex the model becomes. ARIMA models are a powerful and flexible technique for time series analysis and forecasting. However, they have some limitations, such as their assumption of linearity and

stationary data, and their inability to capture nonlinear patterns or long-term trends. Nonetheless, ARIMA models remain a widely-used method for time series analysis and forecasting due to their simplicity and interpretability, as well as their ability to handle a wide range of time series data.

2.2.2 XGBoost

XGBoost (Extreme Gradient Boosting)[1] is a powerful and widely-used machine learning algorithm for classification and regression problems. It is an implementation of gradient boosting that uses decision trees as the base learner. XGBoost has gained popularity in recent years due to its exceptional performance in many machine learning competitions and its ability to handle large and complex datasets. It works by iteratively adding decision trees to the model, each one correcting the errors of the previous tree. The algorithm is designed to minimize a loss function, such as mean squared error for regression problems or log loss for classification problems. In each iteration, the algorithm calculates the gradient and the second derivative of the loss function with respect to the predicted values and then fits a decision tree to the negative gradient. The predictions of the decision tree are then added to the predictions of the previous trees, and the algorithm moves to the next iteration. The learning rate is a hyperparameter that determines the contribution of each tree to the final predictions, and can be tuned to balance between overfitting and underfitting. XGBoost has several features that make it powerful and efficient. First, it uses a technique called "regularization" to prevent overfitting. Regularization includes L1 and L2 regularization, which penalize large coefficients and reduce the complexity of the model. XGBoost also has a built-in method for handling missing data, and can automatically handle categorical features by encoding them as numerical variables. In addition, it supports parallel processing on multiple CPU cores and can be distributed across multiple machines for training on very large datasets. XGBoost has become a popular choice in many machine learning tasks due to its ability to handle a wide variety of data types and perform well on many different types of datasets. It is commonly used in industry for a variety of tasks, such as fraud detection, recommendation systems and image/speech recognition. XGBoost has also been used successfully in various scientific domains, including genomics, physics, and neuroscience to analyze and model complex data.

3

Implementation

For the implementation, the chosen development language is Python. It is a high-level, interpreted programming language that is widely used for a variety of purposes such as web development, scientific computing, artificial intelligence, machine learning, data analysis and more. An advantage of Python is the large community and libraries, because it has a huge and active community of developers who contribute to a vast library of pre-built modules and packages. This makes it easier to find solutions to problems and speeds up the development process. There are a lot of available packages regarding machine learning, clustering and prediction techniques that do for us. However, Python is not perfect. It has a slow execution, given that is an interpreted language, which means that it runs code line by line rather than compiling it first.

We relied on different libraries, such as:

- NumPy[16]: is a popular open-source Python library that provides support for large, multi-dimensional arrays and matrices, along with a large collection of mathematical functions for operating on these arrays.
- Pandas[15]: used for data manipulation, analysis and preparation. It provides powerful data structures and functions for working with structured data, including time series data. Pandas library is built on top of NumPy. More specifically, we used Pandas dataframes and Input/Output. A dataframe is a data structure, which is a two-dimensional table-like structure with rows and columns. It can easily handle heterogeneous data types and missing values. Pandas can read and write data from various file formats such as CSVs, that we used to store results.
- Matplotlib[9]: for creating static, animated, and interactive visualizations in Python. It provides a flexible framework for creating various types of

visualizations, including line plots, scatter plots, bar plots, histograms, heatmaps, and more. In this work, it was used for visualizing the various clustering results.

- Sklearn[3]: also known as Scikit-learn, it is a very popular library for machine learning. It provides a wide range of algorithms and tools such as classification, regression, clustering and dimensionality reduction. It is very useful in this work because it is integrated with NumPy and Pandas packages, it provides metrics for quantifying the quality of predictions and multiple clustering methods.

3.1 Datasets

In this section the datasets used in this thesis are descibed, one of Padua and the other of Montreal. These are briefly reported on Table 3.1 that shows the: types of bike sharing system of the city of origin of the dataset, number of stations if the system is a docking one, number of total bike trips, number of filtered bike trips, size of the file, starting and ending date of the entire collection. The number of filtered trips refers to the travels in a specific range of time that will be detailed on Section 3.2.

Dataset	Padua	Montreal
Bike sharing system	Floating	Docking
Stations number	-	3742
Trips number	979756	29460723
Filtered trips number	119747	3952782
Size	152MB	1.73GB
Starting date	02/05/2019	01/06/2014
Ending date	13/05/2022	16/11/2020

Table 3.1: Comparison between Padua and Montreal datasets

3.1.1 Padua

This thesis was designed in order to try to improve the floating bike sharing system currently in place in the city of Padua. The hometown of the university provided bikes data from May 2019 to May 2022 in the form of a CSV file, that contains a total of 979756 rows. A row contains a lot of columns as can be observed in Figure 3.1 such as: UserId, RideDistance, VehicleType, SOCConsumed, ecc...

But the ones more interesting and useful are the GPS coordinates, the timestamp of the starting and ending moments of the travels. The GPS data were used for the clustering, but a filter has to be applied to them , taking only the starting points. This was done because the ending point is always a starting point for another travel, so there were useless duplicates in the calculations. Given that, only the starting times have been taken, because the aim of this thesis is to know the bikes starting from a specific cluster on a determined time period for predict the number of bicycles that have to be available. This analysis simplified the initial work and the clustering processing time, given a minor number of values in input as showed in Figure 3.2.

```
StartLongitude,StartLatitude,EndLongitude,EndLatitude,UserId,BikeNo,RideDistance(meters),PassUser,
    StartTime,EndTime,ComputedDuration,ComputedDistance,VehicleType,SocStart,SocEnd,SOCConsumed,StartMatrixCoord,EndMatrixCoord

11.87643825,45.41695031,11.88204,45.411818,uid7.964300362248399e+27,A056003615,1188.0,1.0,
    2019-05-02 09:49:43,2019-05-02 09:55:15,5.533333333333333,719.1405947953041,bike,,,,182,201
11.88202872,45.41177937,11.876542,45.407382,uid7.964300362248399e+27,A056003615,2357.0,1.0,
    2019-05-02 10:02:38,2019-05-02 10:19:06,16.466666666666665,650.2101943143313,bike,,,,201,179
11.87573248,45.40558001,11.87603,45.416572,uid7.964300362248399e+27,A056003615,566.0,1.0,
    2019-05-02 10:59:01,2019-05-02 11:17:49,18.8,1222.8195378621745,bike,,,,179,181
11.87765206,45.40810254,11.877587,45.40832,uid7.964300362248399e+27,A056002735,221.0,1.0,
    2019-05-06 10:42:53,2019-05-06 10:49:50,6.95,24.715058269283897,bike,,,,201,201
11.87778234,45.40803452,11.8764222,45.41016349,uid5.535366450069402e+27,A056016862,590.0,0.0,
    2019-05-06 12:16:59,2019-05-06 12:20:39,3.6666666666666665,259.524571361046,bike,,,,201,180
11.87976889,45.41708111,11.877773,45.407934,uid7.964300362248399e+27,A056003636,1387.0,1.0,
    2019-05-06 12:26:09,2019-05-06 12:31:48,5.65,1029.2658247373229,bike,,,,203,201
11.87777664,45.40814527,11.89399,45.409432,uid5.512383884461091e+27,A056003636,2051.0,1.0,
    2019-05-06 12:36:06,2019-05-06 12:44:42,8.6,1274.0963521620922,bike,,,,201,243
11.87780165,45.40807285,11.892915,45.4103,uid7.964300362248399e+27,A056002735,1747.0,1.0,
    2019-05-06 12:40:49,2019-05-06 12:51:22,10.55,1205.847854946591,bike,,,,201,243
11.879774,45.417008,11.907166,45.411337,uid4.839409742773154e+27,A056003979,2830.0,1.0,
    2019-05-06 13:01:11,2019-05-06 13:20:08,18.95,2229.796704822364,bike,,,,203,285
11.87710921,45.39965216,11.875199,45.40908,uid7.964300362248399e+27,A056004509,0.0,1.0,
    2019-05-06 13:27:18,2019-05-06 13:34:50,7.533333333333333,1059.1812334292458,bike,,,,199,180
```

Figure 3.1: Padua dataset

3.1.2 Montreal

Afterwards, some reasoning about a method to evaluate the clustering application methods and the prediction techniques have been done. So, other datasets have been searched and, unfortunately, there were no more floating bike sharing system datasets available. Then, we chose the docking sharing system of Montreal[4] given that its data structure contained a large number of values, exactly 29460723 bike movements. This dataset is a little bit different, given that it has a stations file that lists all the 3742 available docking stations in Montreal as showed in Figure 3.3. This file is the one used for clustering, because the bike travels have fixed starting and ending points. For the predictions, the CSV does not contain directly the GPS coordinates as you can be seen in Figure

```
Latitude,Longitude,Time
45.414245,11.86894,2019-09-01 00:01:01
45.40661471,11.87212596,2019-09-01 00:03:44
45.398165,11.884864,2019-09-01 00:04:43
45.408074,11.87175,2019-09-01 00:06:44
45.40806192,11.87368615,2019-09-01 00:07:15
45.412629,11.878804,2019-09-01 00:13:08
45.41709565,11.8812204,2019-09-01 00:23:48
45.41354039,11.86517864,2019-09-01 00:24:59
45.40767,11.873025,2019-09-01 00:25:04
45.413162,11.89016,2019-09-01 00:28:02
45.41323226,11.8903891,2019-09-01 00:28:06
45.40783902,11.87403571,2019-09-01 00:28:56
45.4064173,11.85944793,2019-09-01 00:30:48
45.40811122,11.91047595,2019-09-01 00:35:21
45.431898,11.865873,2019-09-01 00:42:19
45.413065,11.869516,2019-09-01 00:47:12
45.406941,11.872886,2019-09-01 00:48:47
45.413817,11.874852,2019-09-01 00:49:19
45.40696822,11.89522449,2019-09-01 00:49:55
45.41274,11.891424,2019-09-01 00:59:15
```

Figure 3.2: Padua dataset with only the useful information

3.4. Then, the activity was to merge the bike travels dataset with the stations one, substituting the station's ID in a travel row, with its GPS coordinates. This process gave as outpu the number of bikes in a specific location, that is a station, in a determined time window.

```
code,name,latitude,longitude,yearid
6209,Milton / Clark,45.51252,-73.57061999999999,2014
6436,Côte St-Antoine / Clarke,45.48645209646392,-73.59523415565491,2014
6214,Square St-Louis,45.51735,-73.56906,2014
6248,St-Dominique / Rachel,45.518593,-73.58156600000001,2014
6164,Chambord / Laurier,45.5329550401632,-73.58419418334961,2014
6216,Parc Jeanne Mance (monument George-Étienne Cartier),45.514959999999995,-73.58503,2014
6082,6e avenue / Rosemont,45.550613,-73.582883,2014
6149,Chapleau / du Mont-Royal,45.53867,-73.56936,2014
6265,Parthenais / Laurier,45.539259,-73.577338,2014
6211,Roy / St-Laurent,45.515609999999995,-73.57569000000001,2014
6252,Mozart / St-Laurent,45.53318,-73.61544,2014
6233,Bernard / Jeanne-Mance,45.524295872822634,-73.60484719276428,2014
6064,de Maisonneuve / Stanley,45.50038,-73.57507,2014
6408,Métro George-Vanier (St-Antoine / Canning),45.4891,-73.57656,2014
6068,Mansfield / Sherbrooke,45.50297,-73.57505,2014
6093,Atwater / Sherbrooke,45.49132174886071,-73.58780980110168,2014
6253,Lajeunesse / Jean-Talon,45.53973680304488,-73.61425638198854,2014
6336,Faillon / St-Hubert,45.542794353049906,-73.61810535192491,2014
6223,du Mont-Royal / du Parc,45.516999999999996,-73.589,2014
6062,Drummond / Ste-Catherine,45.49863930330429,-73.57422709465027,2014
6434,Victoria / de Maisonneuve,45.47742856483265,-73.60016942024231,2014
6099,Bishop / de Maisonneuve,45.49741095430152,-73.57827052474022,2014
6280,Fairmount / St-Dominique,45.524504989237535,-73.59414249658585,2014
6912,de Châteaubriand / Beaubien,45.535340600000005,-73.60365897,2014
6231,St-Viateur/Parc,45.52223310369562,-73.60208183526994,2014
6206,Prince-Arthur / du Parc,45.51059,-73.57547,2014
6411,Clark / Prince-Arthur,45.513338,-73.57295,2014
6227,de l'Esplanade / Laurier,45.52111622098953,-73.59478354454039,2014
6100,Mackay /de Maisonneuve (Sud),45.496590000000005,-73.57851,2014
6364,Chambly / Rachel,45.549910693591,-73.55826258659361,2014
6380,Parc J.-Arthur-Champagne (Chambly / du Mont-Royal),45.551584000000005,-73.561916,2014
```

Figure 3.3: Montreal stations dataset

```
start_date,start_station_code,end_date,end_station_code,duration_sec,is_member,yearid
2014-06-01 00:00:00,6223,2014-06-01 00:27:00,6004,1620.0,0,2014
2014-06-01 00:00:00,6223,2014-06-01 00:28:00,6004,1680.0,0,2014
2014-06-01 00:00:00,6255,2014-06-01 00:05:00,6907,300.0,0,2014
2014-06-01 00:00:00,6059,2014-06-01 00:21:00,6008,1260.0,0,2014
2014-06-01 00:00:00,6059,2014-06-01 00:21:00,6008,1260.0,0,2014
2014-06-01 00:00:00,6203,2014-06-01 00:03:00,6047,180.0,1,2014
2014-06-01 00:00:00,6202,2014-06-01 00:08:00,6175,480.0,1,2014
2014-06-01 00:00:00,6170,2014-06-01 00:12:00,6112,720.0,1,2014
2014-06-01 00:00:00,6038,2014-06-01 00:08:00,6007,480.0,1,2014
2014-06-01 00:00:00,6350,2014-06-01 00:11:00,6003,660.0,1,2014
2014-06-01 00:00:00,6328,2014-06-01 00:31:00,6186,1860.0,1,2014
2014-06-01 00:03:00,6241,2014-06-01 00:37:00,6092,2040.0,0,2014
2014-06-01 00:00:00,6323,2014-06-01 00:19:00,6221,1140.0,1,2014
2014-06-01 00:00:00,6180,2014-06-01 00:15:00,6122,900.0,1,2014
2014-06-01 00:01:00,6070,2014-06-01 00:11:00,6107,600.0,1,2014
2014-06-01 00:01:00,6184,2014-06-01 00:34:00,6012,1980.0,0,2014
2014-06-01 00:01:00,6063,2014-06-01 00:15:00,6211,840.0,1,2014
2014-06-01 00:01:00,6214,2014-06-01 00:07:00,6902,360.0,1,2014
2014-06-01 00:02:00,6241,2014-06-01 00:38:00,6092,2160.0,0,2014
2014-06-01 00:01:00,6912,2014-06-01 00:12:00,6200,660.0,0,2014
2014-06-01 00:01:00,6323,2014-06-01 00:22:00,6221,1260.0,1,2014
2014-06-01 00:01:00,6211,2014-06-01 00:18:00,6264,1020.0,1,2014
2014-06-01 00:01:00,6183,2014-06-01 00:09:00,6199,480.0,1,2014
2014-06-01 00:01:00,6260,2014-06-01 00:23:00,6396,1320.0,1,2014
2014-06-01 00:01:00,6231,2014-06-01 00:09:00,6912,480.0,1,2014
2014-06-01 00:01:00,6280,2014-06-01 00:20:00,6009,1140.0,1,2014
2014-06-01 00:01:00,6142,2014-06-01 00:19:00,6273,1080.0,1,2014
2014-06-01 00:02:00,6114,2014-06-01 00:15:00,6037,780.0,1,2014
2014-06-01 00:02:00,6211,2014-06-01 00:07:00,6020,300.0,1,2014
2014-06-01 00:03:00,6915,2014-06-01 00:25:00,6202,1320.0,1,2014
2014-06-01 00:04:00,6196,2014-06-01 00:15:00,6083,660.0,0,2014
2014-06-01 00:03:00,6050,2014-06-01 00:15:00,6921,720.0,0,2014
2014-06-01 00:03:00,6149,2014-06-01 00:07:00,6134,240.0,1,2014
2014-06-01 00:03:00,6700,2014-06-01 00:08:00,6397,300.0,1,2014
2014-06-01 00:04:00,6050,2014-06-01 00:15:00,6921,660.0,0,2014
2014-06-01 00:03:00,6070,2014-06-01 00:17:00,6729,840.0,1,2014
```

Figure 3.4: Montreal trips dataset

3.2 Data processing

At the start of 2022, as everybody knows, there was the start of the Covid-19 spread. The Padua university and the whole world have to start a massive use of remote working and lecturing. Being the floating bike sharing system of Padua very influenced by the students presence in the city, it suffered a drop in usage. So, we had to think about which time period of the dataset to analyze. We decided to take a three month long time window, starting from the 1st of September 2019 to the 31th of November of the same year. This is because it is the period in which the university starts the lectures and there are a lot of student movements in the city.

For Montreal, we applied the same logic, adding the consideration about the weather in Canada. Since it is a more colder region, we decided to take a different period 4 months long, starting on the 1st of June 2019 ending on the 30th of September 2019.

For both of them, we saved in a Pandas dataframe with the GPS coordinates of the points, the timestamp of the travel's start for the prediction process. We decided to predict the number of bikes that have to be available in a certain cluster for every hour in a day. For doing that, we had to calculate the number of travels that have started from a region every hour for training our prediction model.

```
startDate = datetime.datetime(2019, 9, 1, 0, 0, 0)
endDate = datetime.datetime(2019, 11, 30, 23, 59, 59)
data= pd.read_csv("./Padova.csv")

for row in data.itertuples():
    date = datetime.datetime.strptime(row.StartTime, '%Y-%m-%d %H:%M
    :%S')

    if date >= startDate and date <= endDate:
        longData.append(row.StartLongitude)
        latData.append(row.StartLatitude)
        time.append(date)
```

Code 3.1: Padua data filtering and processing example

```
startDate = datetime.datetime(2019, 6, 1, 0, 0, 0)
endDate = datetime.datetime(2019, 9, 30, 23, 59, 59)
tripsData = pd.read_csv("./Montreal/trips.csv")
stationsData = pd.read_csv("./Montreal/stations.csv")

for row in stationsData.itertuples():
        stationsDict[row.code] = GpsData(row.longitude, row.latitude)
        insertedStations[row.code] = 0

for row in tripsData.itertuples():
    date = datetime.datetime.strptime(row.start_date, '%Y-%m-%d %H:%M
    :%S')

    if date >= startDate and date <= endDate:
        stationCode = int(row.start_station_code)

        if insertedStations[stationCode] == 0:
            insertedStations[stationCode] = 1
            longData.append(stationsDict[stationCode].longitude)
            latData.append(stationsDict[stationCode].latitude)
```

```
20                time.append(date)
```

Code 3.2: Montreal data filtering and processing example

In the Montreal process, Code 3.2, we loop on all the trips data to get only the station used in the fixed time period. Furthermore, for distinct them we used an Hash map that has as a key the ID of a station and for value a flag equal to 0 if the station is not inserted and 1 otherwise. So, we bypass the station not used or already selected.

3.3 Clustering techniques

In this thesis, as described in the previous chapter, we used different clustering techniques and the next sections will describe their implementations. For determining the perfect number of clusters for the Padua dataset, we used the K-Means algorithm with a range of values from 1 to 300. For each result, we calculated its objective function and at the end we compared them. As will be seen below, we have to take a perfect square as clusters number for the Squares algorithm. Given that and the results of the K-Means process, we chose to cluster the dataset into 16, 36 and 64 regions.

3.3.1 Squares

The function that processes the division into squares accepts as input two parameters.

- A dataframe, that contains the GPS coordinates of every point.
- An integer, that is the number of desired clusters. It has to be a perfect square, because the algorithm divides the whole area into rows and columns with the same number of squares.

Afterwards, the algorithm processes the maximum and minimum latitude and longitude for storing the border of the main area. With them, the calculation of the square height and width can be easily obtained by dividing the subtraction of the maximum and minimum latitude and longitude respectively with the square root of the clusters' number. At the end of these foreplay steps, the function iterates each coordinate point in the dataset, calculating its cluster label. If the point will not be included into any region, it means it is a right border point and has to be included in the rightest square of the current row that is being processed.

```
def DivideIntoSquares(data, numberOfSquares):
    squaresPerRow = int(sqrt(numberOfSquares))

    maxLat = maxLong = -181
    minLat = minLong = 181

    for latitude in data["Latitude"]:
        if latitude > maxLat:
            maxLat = latitude
        else:
            if latitude < minLat:
                minLat = latitude

    for longitude in data["Longitude"]:
        if longitude > maxLong:
            maxLong = longitude
        else:
            if longitude < minLong:
                minLong = longitude

    squareHeight = (maxLat - minLat) / squaresPerRow
    squareWidth = (maxLong - minLong) / squaresPerRow

    labels = []

    for row in data.itertuples():
        flag = False

        for i in range(1, squaresPerRow + 1):
            if row.Latitude >= minLat + (squareHeight * (i - 1)) and
   row.Latitude < minLat + (squareHeight * (i)):
                for k in range(1, squaresPerRow + 1):
                    if row.Longitude >= minLong + (squareWidth * (k -
    1)) and row.Longitude < minLong + (squareWidth * (k)):
                        labels.append(squaresPerRow * (k - 1) + i -
   2)
                        k = i = squaresPerRow + 10
                        flag = True

        if flag == False:
            labels.append(squaresPerRow * (k - 1) + i - 1)

```

```
40     return labels
```

Code 3.3: Squares clustering function

3.3.2 K-Means

For the K-Means algorithm, it was used the Sklearn clustering package passing the number of desired clusters and selecting as initializer the "k-means++" option.

```
1 kmeans = KMeans(n_clusters = numberOfClusters, init ='k-means++')
2 kmeans.fit(data)
3 labels = kmeans.predict(data)
```

Code 3.4: K-Means clustering implementation

3.3.3 GMM

For GMM, it was used the same clustering package as K-Means, passing only the number of desired clusters. The default initializer of GMM is "k-means".

```
1 gmm = GaussianMixture(n_components = clusters).fit(data)
2 labels = gmm.predict(data)
```

Code 3.5: GMM clustering implementation

3.3.4 Mean Shift

The implementation of Mean Shift uses the Sklearn clustering library, that accepts as input a parameter called Bandwidth. If not given, it calculates it by itself. The Bandwidth is the distance scale of the kernel function and is calculated with a quantile value, that is used in KNN inside the "estimate_bandwidth" method to determine the bandwidth. Different quantile values produce various Bandwidth and the Mean Shift clustering gives as output a different number of clusters. Many testes have been done to identify the perfect quantile values for generating fixed numbers of clusters. The results with our datasets are:

- Quantile: 0.13 produces 16 clusters on Padua and 0.08 produces 9 clusters on Montreal
- Quantile: 0.06725 produces 36 clusters on Padua and 0.052 produces 16 clusters on Montreal

- Quantile: 0.037 produces 64 clusters on Padua and 0.028 produces 36 clusters on Montreal

```
bandwidth = estimate_bandwidth(data, quantile=quantile)
labels = MeanShift(bandwidth=bandwidth).fit(data).labels_
```

Code 3.6: Mean Shift clustering implementation

3.3.5 BIRCH

For BIRCH clustering, it was used one more time the Sklearn clustering library. The algorithm accepts two input parameters, such as: the number of clusters and a threshold value. The threshold has to be higher than the radius of the subcluster obtained by merging a new sample and the closest subcluster, otherwise a new subcluster is started. But, because it has been used a fixed number of clusters, we set this value to a low number, exactly 0.001. That permits a higher division of regions in the dataset, so the algorithm can perform a merge of these zones to obtain the desired clusters' number.

```
labels = Birch(threshold=distanceThrehsold, n_clusters=clusters).fit(
    data).labels_
```

Code 3.7: BIRCH clustering implementation

3.4 Data filtering

After the clustering, the script transforms the output result into a matrix in which every column is a cluster and every row is the number of bikes in that cluster in a given hour. In other words, the first row indicates the number of bicycles in every region from the 0:00 to the 0:59 of the first day chosen (see section 3.2), the second row from 1:00 to 1:59, and so on.

```
def ProcessElementsPerCluster(data, clusters):
    elementsPerCluster = []
    for i in range(0, clusters):
        elementsPerCluster.append([0])

    currentStartDate = datetime.datetime.strptime(data["Time"][0], '%
    Y-%m-%d %H:%M:%S').replace(hour=0, minute=0, second=0)
    currentEndDate = currentStartDate + timedelta(minutes=59, seconds
    =59)
```

```
8
9     for row in data.itertuples():
10        rowTime = datetime.datetime.strptime(row.Time, '%Y-%m-%d %H:%
    M:%S')
11
12        if rowTime >= currentStartDate and rowTime <= currentEndDate:
13            elementsPerCluster[row.ClusterLabel][-1] += 1
14        else:
15            currentStartDate = currentEndDate + timedelta(seconds =
    1)
16            currentEndDate = currentStartDate + timedelta(minutes=59,
     seconds=59)
17
18            for i in range(0, clusters):
19                elementsPerCluster[i].append(0)
20
21            elementsPerCluster[row.ClusterLabel][-1] += 1
22    return elementsPerCluster
```

Code 3.8: Count elements per cluster script

Every clustering algorithm gives as output a different result. Sometimes it is similar, like GMM and K-Means, other times it is very different, such as Mean Shift and Squares. As an example, the Squares approach creates a lot of almost empty squares, that are near the border of the main square. This situation produces easier predictions in those regions, because the real value is almost always equal to 0. Given that, the evaluation results are higher than expected and we need to find a way to balance the output of different clustering methods. At this point, it is important to introduce some kind of filtering methods.

The first one is the filter by values method. The function accepts as inputs the matrix of elements per cluster, the number of clusters processed by the clustering algorithm and a threshold, that is the minimum number of values in total that a region must have to be considered in the evaluation. We decided to filter the clusters, taking into account only the ones with at least 90 values inside. Namely, areas with a minimum of one travel, that started inside the area, per day. The function gives as outputs the new filtered matrix and the updated clusters number.

```
1 def FilterClusters(elementsPerCluster, clusters, threshold):
2     if threshold <= 0:
3         return elementsPerCluster, clusters
4
```

```
    currentColumn = 0

    for i in range(0, clusters):
        i = currentColumn
        cont = 0

        for k in range (0, len(elementsPerCluster[i])):
            cont += elementsPerCluster[i][k]

        valuesList.append(cont)

        if cont < threshold:
            elementsPerCluster.pop(i)
            clusters -= 1
        else:
            currentColumn += 1

    return elementsPerCluster, clusters
```

Code 3.9: Filtering by a threshold

The second method of filtering is taking only the first k populous clusters. This can bring different results. The K-Means algorithm creates more or less balanced clusters, looking at the number of values inside them. Instead, Mean Shift does not divide the area with the most values into many clusters, but creates more clusters in the outermost regions. Therefore, this filtering system serves more to understand and evaluate the goodness of the predictions, given that having much higher and fluctuating values, it is much more difficult to correctly predict the exact value of bikes. The function takes as inputs the matrix of elements per cluster, the number of clusters processed by the clustering algorithm and the number of the clusters to give as output.

```
def GetKTopCluster(elementsPerCluster, clusters, k):
    if k <= 0:
        return elementsPerCluster, clusters

    valuesList = list()
    topClusters = []
    topClustersValues = []
    topClustersIndex = []

    for i in range(0, clusters):
        cont = 0
```

```
12
13          for z in range (0, len(elementsPerCluster[i])):
14              cont += elementsPerCluster[i][z]
15          valuesList.append(cont)
16
17      for i in range(0, k):
18          maxIndex = 0
19
20          for x in range(1, clusters):
21              if valuesList[x] > valuesList[maxIndex]:
22                  maxIndex = x
23
24          topClustersIndex.append(maxIndex)
25          topClusters.append(elementsPerCluster[maxIndex])
26          topClustersValues.append(valuesList[maxIndex])
27          valuesList[maxIndex] = -1
28
29      return topClusters, k
```

Code 3.10: Filtering by the top k clusters

3.5 Predictions

For generating the predictions, as described in Section 2.2, we implemented two techniques: ARIMA and XGBoost. Both take as inputs the matrix of elements per cluster and the number of clusters processed by the clustering algorithm. Moreover, we divided each column of the matrix into train and test sets. The first one is two-third of the original size of the column and the second is the remaining part.

3.5.1 ARIMA

The predictions function that uses ARIMA, loops for each column in the matrix and divides every column in the train and test sets. After, it loops for each value in the test set, training every time the model with the train set with the desired order of ARIMA. More specifically, we pass the (p, d, q) order of the model for the autoregressive, differences, and moving average components. d is always an integer, while p and q may either be integers or lists. We used in order the values $(1, 1, 1)$ that are the most common parameters for the non-seasonal

time series. The function gives as output the real values of the test set and the predicted ones.

```
def ProcessArimaPredictions(elementsPerCluster, clusters):
    predictions = list()
    values = list()

    for i in range(0, clusters):
        tmp = elementsPerCluster[i]
        size = int(len(tmp) * 0.66)
        train, test = tmp[0:size], tmp[size:len(tmp)]
        values.extend(test)
        history = [x for x in train]

        for t in range(len(test)):
            model = ARIMA(history, order=(1,1,1))
            model_fit = model.fit()
            output = model_fit.forecast()
            predictions.append(output[0])
            history.append(test[t])

    return values, predictions
```

Code 3.11: ARIMA predictions function

3.5.2 XGBoost

XGBoost works slightly differently from ARIMA. It needs again the train and test sets, but split into two more sets. Then, we have X and Y train and test sets. Y contains the same elements of X, but shifted left by one value, because it has to contain the value of the next hour with respect to the one in the same position in the X set. In other words, a cell in Y represents the number of bikes that has to be available in the next hour, given the bikes available now, that are contained in X. The function loops for each existing cluster, constructing a regressor. The regressor takes as inputs the train sets and an evaluation set, composed by all train and test sets for the current cluster. After this operation, for each regressor we take the predicted values.

```
def ProcessXGBoostPredictions(elementsPerCluster, clusters):
    predictions = list()
    values = list()

```

```
    X_train_list = []
    X_test_list = []
    y_train_list = []
    y_test_list = []
    totSize = len(elementsPerCluster[0])
    testSize = int(len(elementsPerCluster[0]) * 0.66)

    for i in range(0, clusters):
        X_train = pd.DataFrame()
        X_test = pd.DataFrame()
        y_train = pd.DataFrame()
        y_test = pd.DataFrame()

        trainList, testList = elementsPerCluster[i][0:testSize],
    elementsPerCluster[i][testSize:totSize]

        X_train["Values"] = trainList
        y_train["Values"] = trainList
        X_train = X_train.tail(-1)
        y_train = y_train.head(y_train.shape[0] -1)

        X_test["Values"] = testList
        y_test["Values"] = testList
        X_test = X_test.tail(-1)
        y_test = y_test.head(y_test.shape[0] -1)
        values.extend(X_test["Values"])

        X_train_list.append(X_train)
        X_test_list.append(X_test)
        y_train_list.append(y_train)
        y_test_list.append(y_test)

    reg_list = []
    for i in range(0, clusters):
        reg = XGBRegressor(base_score=0.5, booster='gbtree',
                           n_estimators=900,
                           early_stopping_rounds=50,
                           objective='reg:squarederror',
                           max_depth=10,
                           learning_rate=0.005)
        reg.fit(X_train_list[i], y_train_list[i],
                eval_set=[(X_train_list[i], y_train_list[i]), (
    X_test_list[i], y_test_list[i])],
```

```
46                  verbose=100)
47          reg_list.append(reg)
48
49      for i in range(0, clusters):
50          predictions.extend(reg_list[i].predict(X_test_list[i]))
51
52      return values, predictions
```

Code 3.12: XGBoost predictions function

4 Results

This section will describe the results of the different clustering techniques and the prediction algorithms applied to the Padua and Montreal datasets.

4.1 Clustering

We applied the different clustering techniques presented on Section 2.1, that gives us very different results in terms of regions created. The distinction is already very clear in the partition in 16 clusters of Padua or 9 regions of Montreal. Omitting the Squares algorithm, that divides the dataset into regions of the same size, leaving out the density of data, K-Means and GMM have a similar approach, given that GMM uses K-Means as initializer. However, K-Means still makes regions that are more homogeneous and with a similar number of bikes inside each one. Mean Shift uses a complete opposite approach. It creates macro regions in the center of Padua and Montreal, that is to say where there are the highest number of items, and tends to divide the external parts into several sections, creating clusters with very low bicycles inside. BIRCH is in between them. Create areas, that are larger than K-Means and GMM, but still try to divide the whole area into balanced clusters. All of these observations can be noticed in both divisions of Padua in 36 or 64 clusters and of Montreal in 16 or 36 areas.

For the clustering implementation and process, we used the same machine, in a way that we can compare the execution time of each clustering technique. We reported the data in Figure 4.7 (table format on Table A.1) and the values

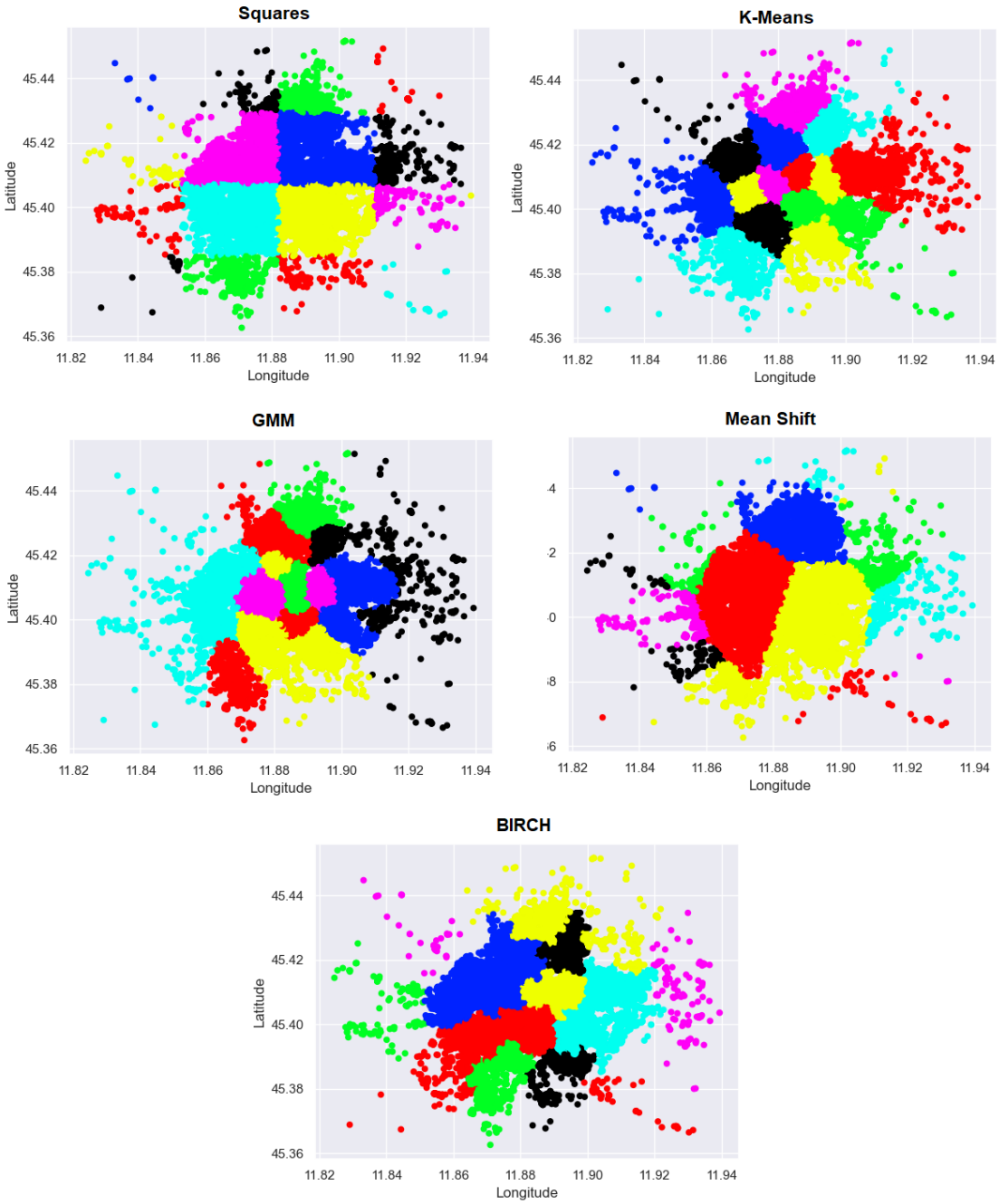

Figure 4.1: Results of clustering Padua in 16 regions

are very similar. The unique exception is the Mean Shift, with very high times with the highest linked with the lowest number of clusters desired. One more thing to notice, is the time that takes GMM to perform 64 clusters. With a lower number of regions the execution time is very fast, the last one is a little bit over

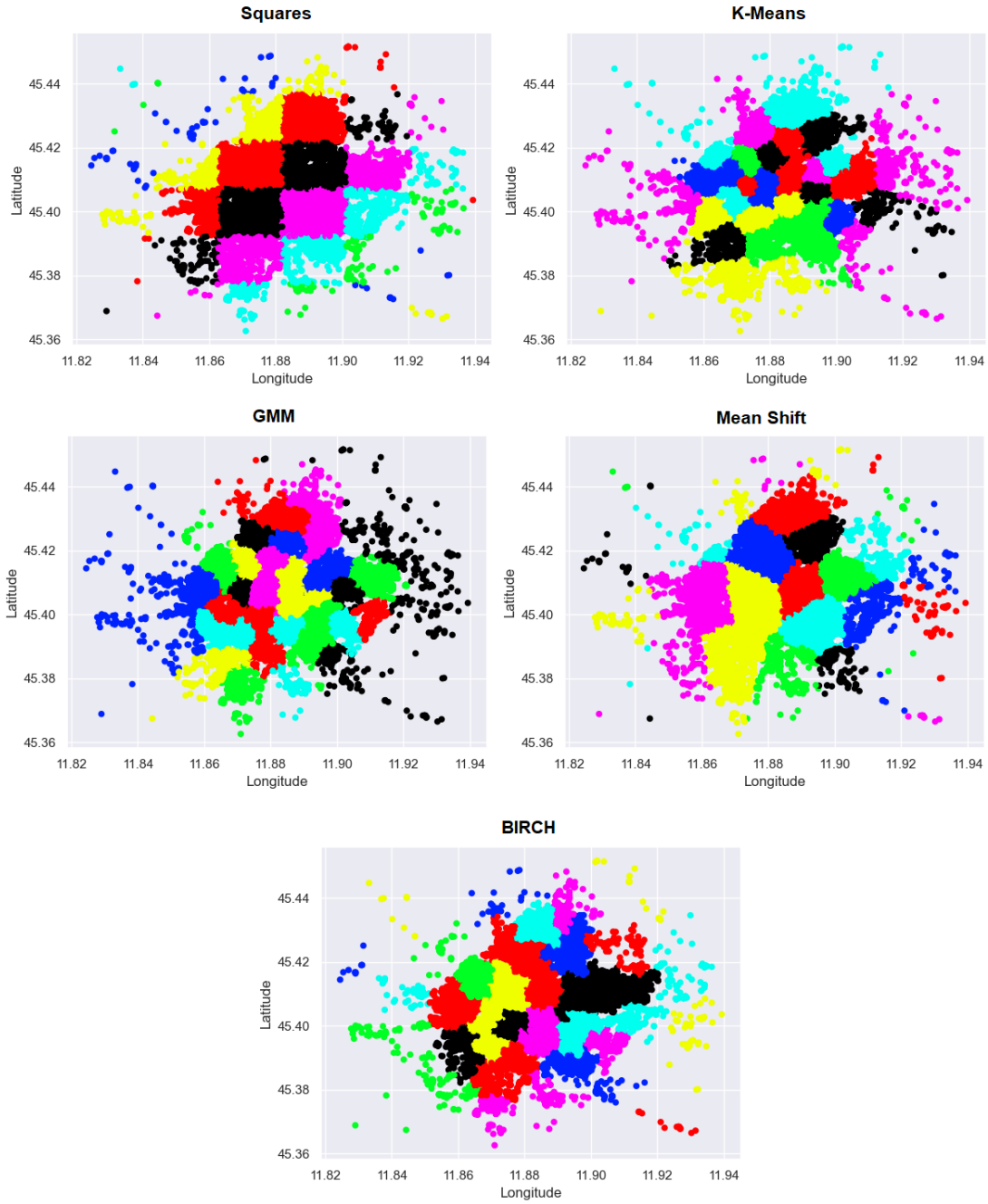

Figure 4.2: Results of clustering Padua in 36 regions

the mean, probably caused by the iterations that GMM has to do for creating a number of clusters higher than the correct one for the dataset. For the Montreal dataset, we have not reported the execution times, given they are below the second, apart for Mean Shift that is around 3 seconds.

To analyze better the differences between the clustering techniques, we con-

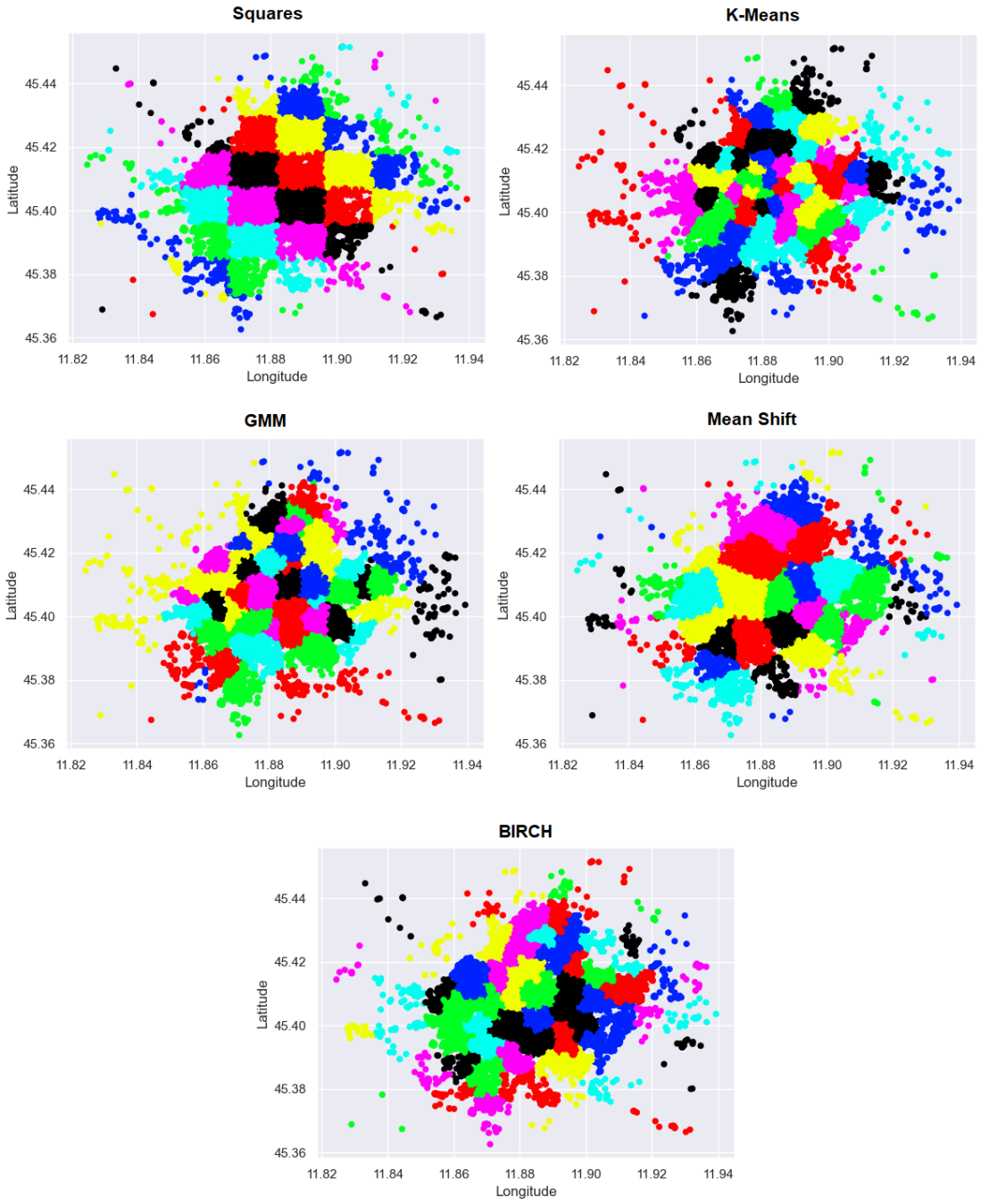

Figure 4.3: Results of clustering Padua in 64 regions

centrated into the number of bikes per cluster. We want to know which is the better clustering method in term of homogeneity, because if a clustering algorithm creates one or more region with a very low number of bikes or only one with a very elevated, it is not good for the scope of this thesis. With reference

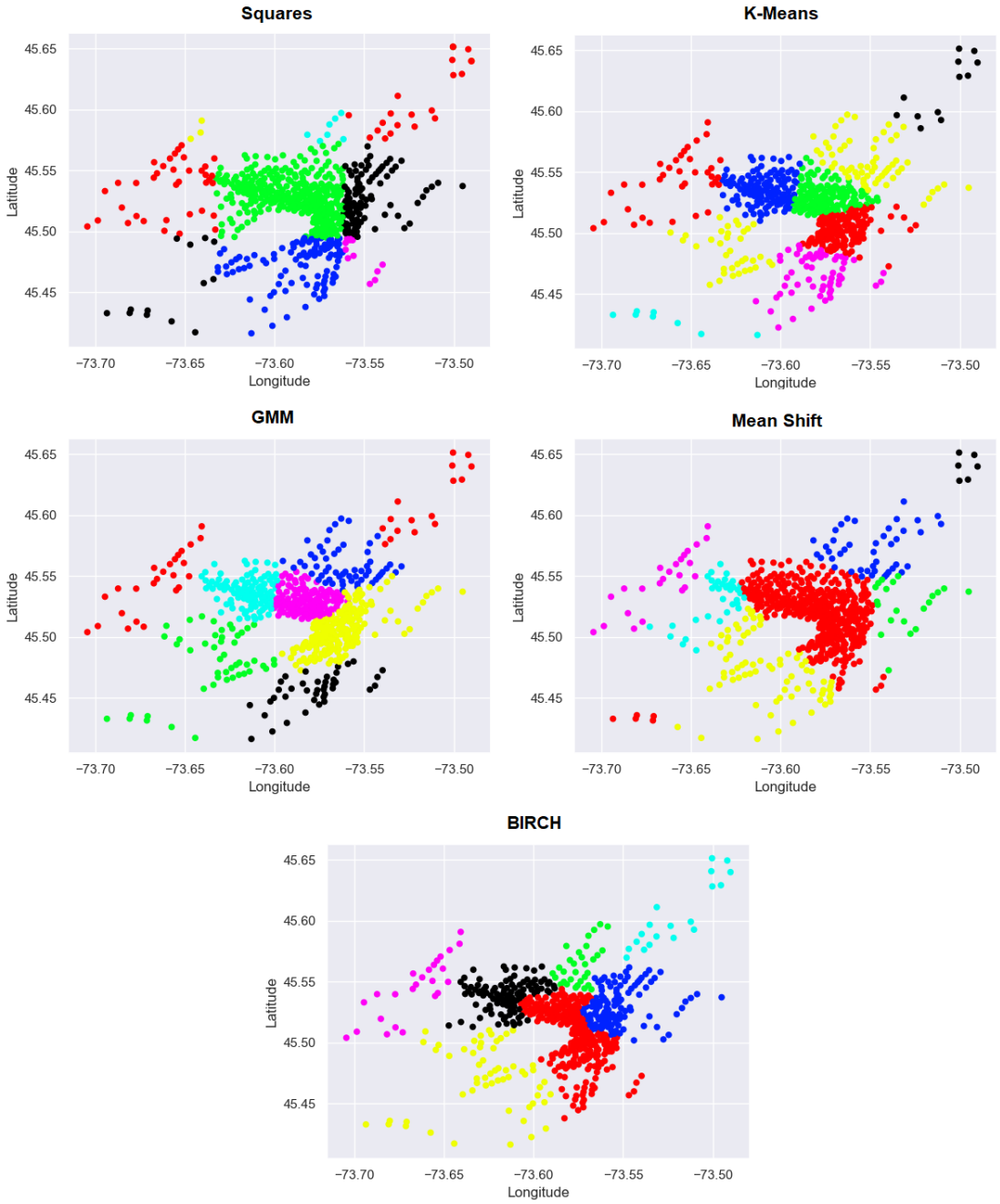

Figure 4.4: Results of clustering Montreal in 9 regions

to Figures 4.8, 4.9 and 4.10, we can see again that K-Means is the more balanced clustering algorithm, having a not so high maximum number of items in a cluster, that is similar to BIRCH and GMM in every division of the Padua dataset. Furthermore, it has always the highest minimum, that means it creates regions that are almost always not empty. GMM works similarly in the first two partition

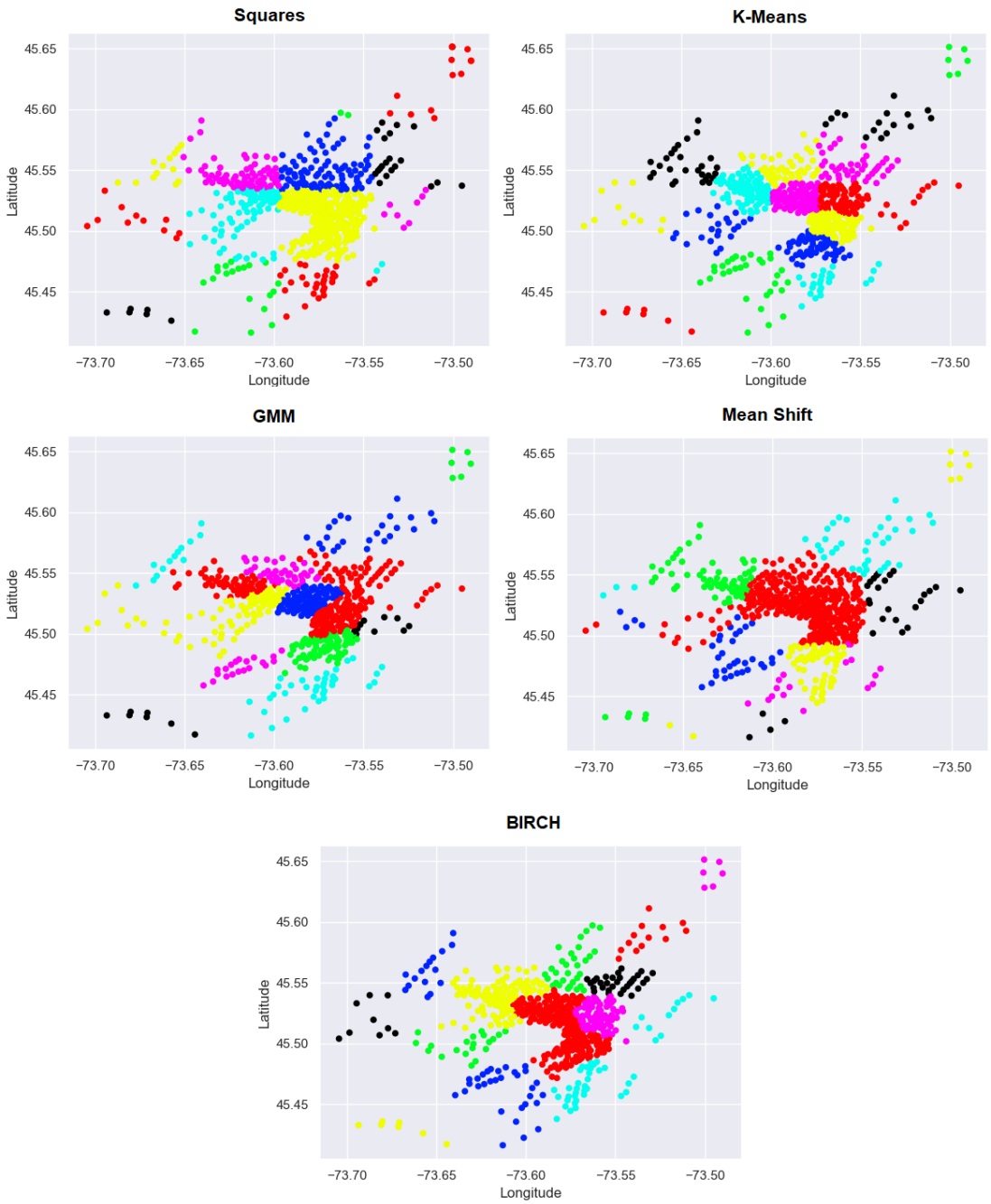

Figure 4.5: Results of clustering Montreal in 16 regions

cases: 16 and 36, while in the last it creates at least an empty cluster. Considering this evaluation, Mean Shift is the worse one, creating few clusters with a lot of bikes inside them and many with a low number of items. BIRCH, as before, is somewhere in between, but creating enough more or less empty clusters.

Observing the Montreal dataset, in Figures 4.11, 4.12 and 4.13, we can see that

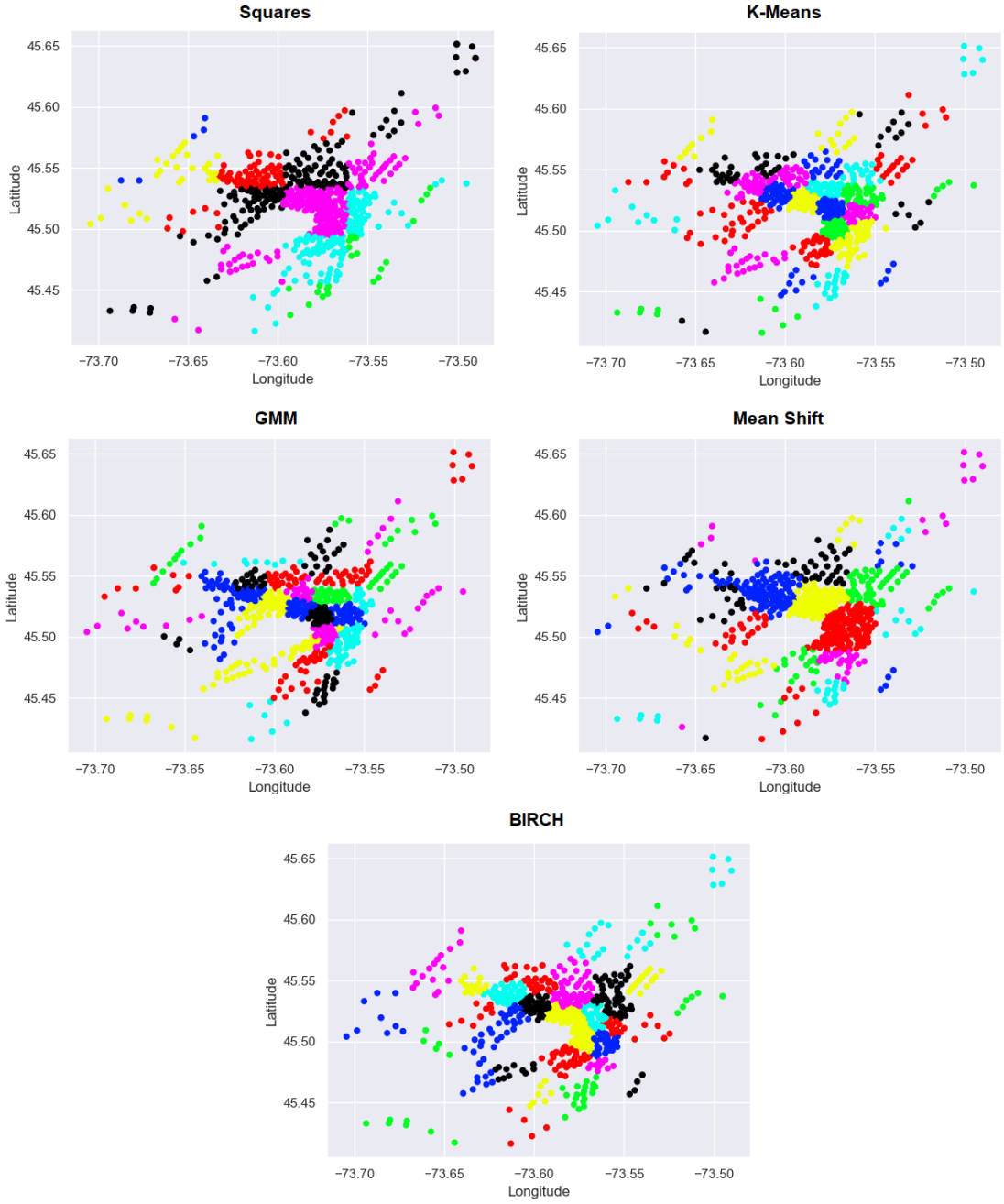

Figure 4.6: Results of clustering Montreal in 36 regions

the results are similar to the Padua ones, with GMM and K-Means winning the battle and BIRCH near them. In this cases, GMM tends to be better compared with K-Means, because this time we have chosen a not very high number of clusters. A curious thing that happened is that the lower number of items in a cluster is equal for all but Squares, in the 16 and 36 cases. This is caused by the

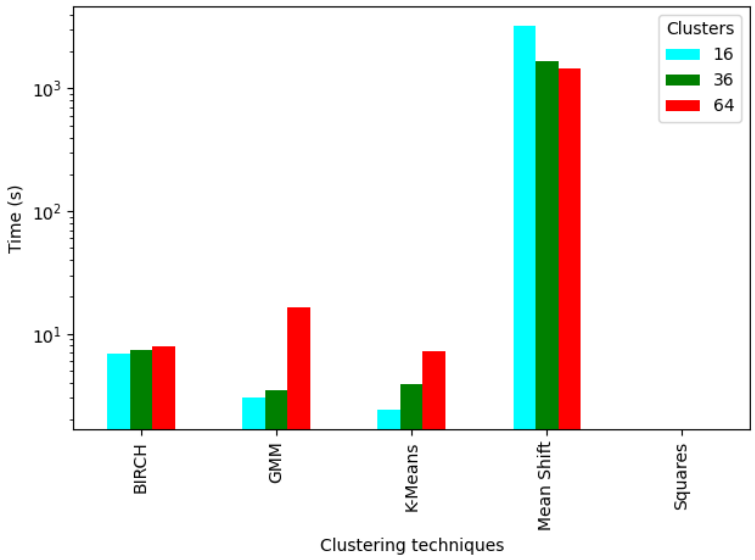

Figure 4.7: Execution times of the clustering algorithms on Padua dataset

fact that the stations are already spread in the city and the clustering algorithms tend to create similar clusters in the areas with a low number of items.

At the end, we can say that GMM and K-Means are the best clustering techniques for the problem presented in this thesis, with GMM having a drop when the desired number of clusters exceeds the perfect number for that dataset.

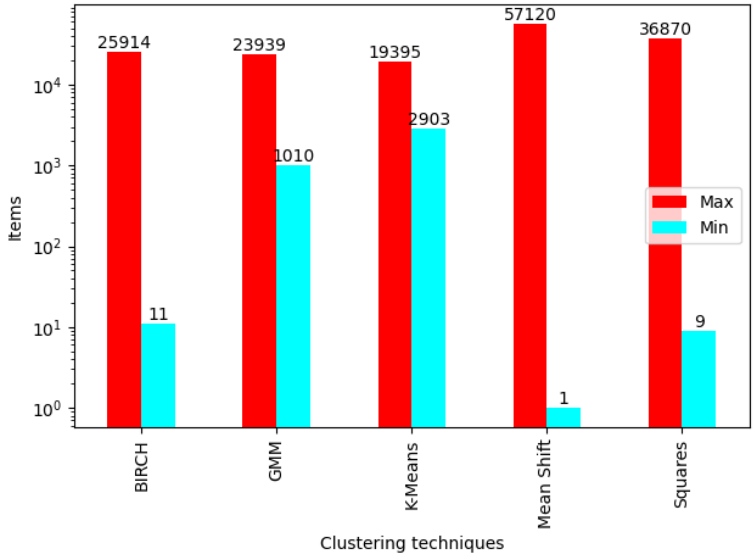

Figure 4.8: Clusters with the higher and lower number of items on 16 clusters for Padua

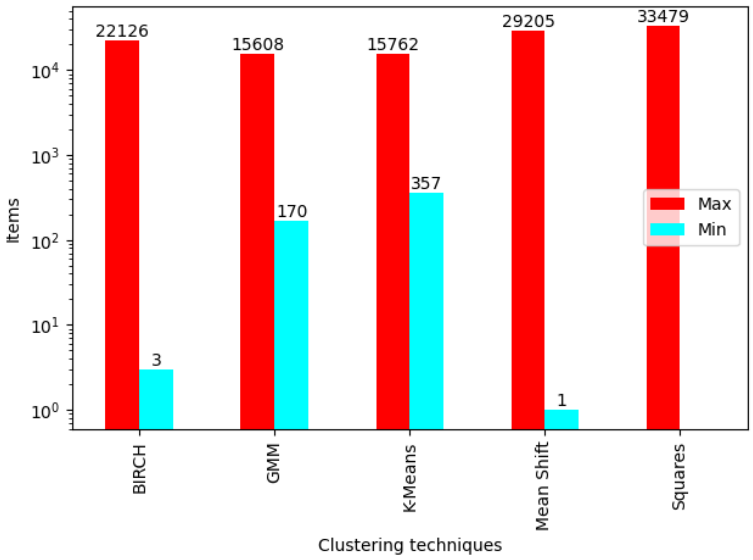

Figure 4.9: Clusters with the higher and lower number of items on 36 clusters for Padua

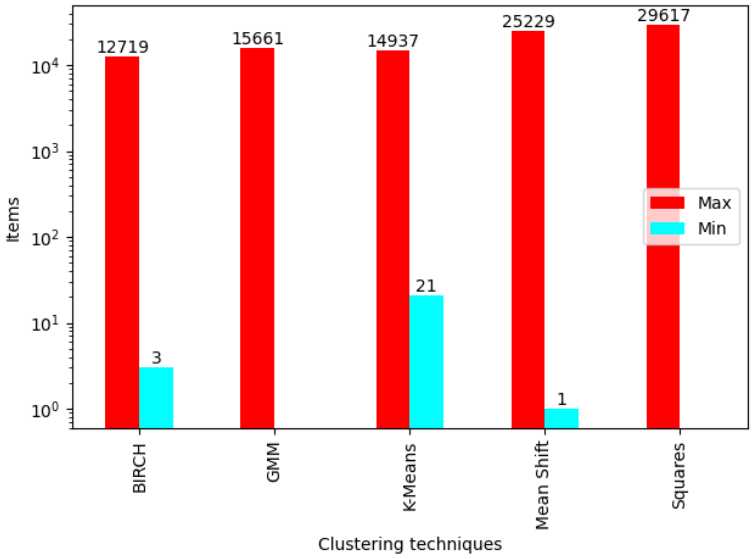

Figure 4.10: Clusters with the higher and lower number of items on 64 clusters for Padua

4.2 Evaluation metrics

After the clustering, we proceeded with the prediction steps. We use two techniques, ARIMA and XGBoost, that provide us different results. We had to find a way to compare their outcomes to understand better which is nicer and

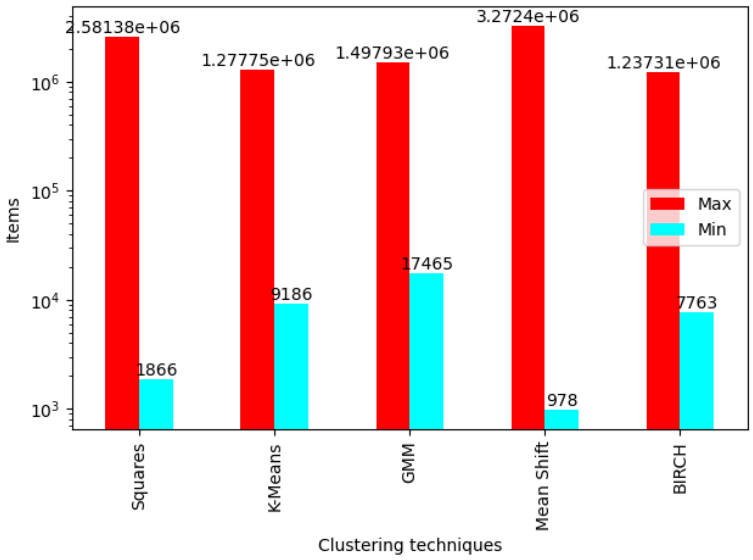

Figure 4.11: Clusters with the higher and lower number of items on 9 clusters for Montreal

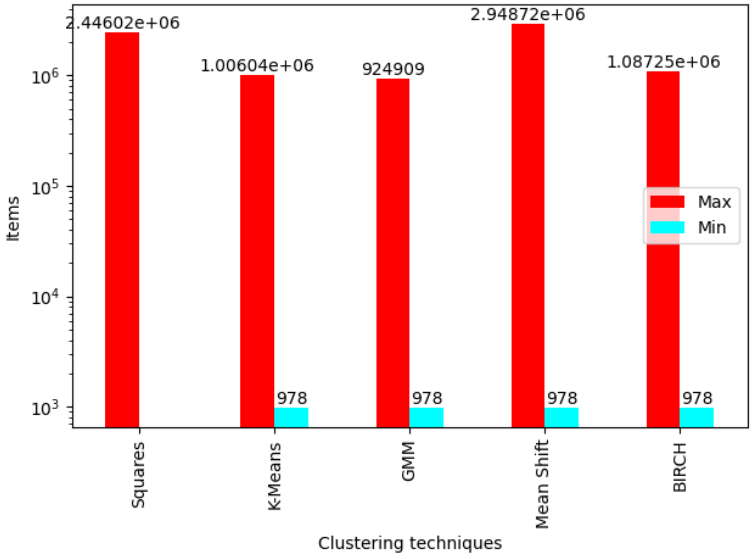

Figure 4.12: Clusters with the higher and lower number of items on 16 clusters for Montreal

why. We introduced different statistic metrics:

- Real values mean: a simple mean that considers all the values returned by the function presented on Snippet 3.8, summing all the values from the dataset, present at each hour in every cluster divided by the number of time intervals.
- Predicted values mean: as above, but using the predicted values for each

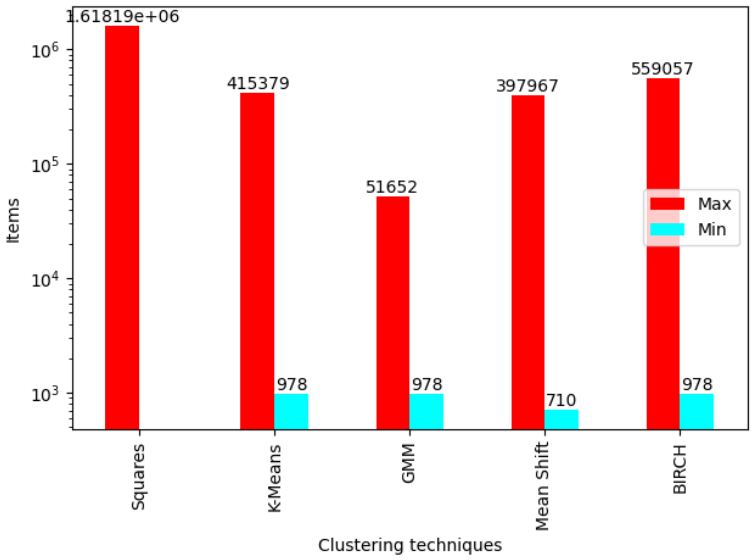

Figure 4.13: Clusters with the higher and lower number of items on 36 clusters for Montreal

hour in the time period.

- RMSE: stands for Root Mean Square Error and it measures the average difference between the actual values and the predicted values of a set of data.

$$RMSE = \sqrt{\frac{1}{n} * \sum (real\_value - predicted\_value)^2}$$

RMSE is particularly useful when you want to measure the accuracy of a predictive model, such as a regression model or a machine learning algorithm. It is often used as a benchmark to compare the performance of different models, with lower values of RMSE indicating better performance. For example, if the RMSE value is 10 for a set of data, it means that on average, the predicted values of the model are off by 10 units from the actual values.

- MAE: Mean Absolute Error, it measures the average absolute difference between the actual values and the predicted values of a set of data.

$$MAE = \frac{1}{n} * \sum |real\_value - predicted\_value|$$

As before, the lower the better. The interpretation of the result is quite similar to the RMSE.

- R2 score: known as the coefficient of determination, is a statistical measure that is used to evaluate the goodness of fit of a regression model. It is a number between 0 and 1 that represents the proportion of the variance in the dependent variable that is explained by the independent variables in

the model.

$$R2 = 1 - \frac{SSres}{SStot}$$

where $SSres$ is the sum of the squared residuals (the difference between the actual and predicted values) and $SStot$ is the total sum of squares (the difference between the actual values and the mean of the dependent variable). The R2 score ranges from 0 to 1, where a value of 0 indicates that the model does not explain any of the variance in the dependent variable, while a value of 1 indicates that the model perfectly explains all the variance in the dependent variable. However, it can be influenced by the number of independent variables in the model and other factors.

4.3 Comparison

In this section we will compare the results of the prediction with ARIMA and XGBoost techniques, both on Padua and Montreal datasets.

4.3.1 Padua

For the Padua dataset, the clustering is important as the predictions. This is because there is a floating bike sharing system and there can be a bike anywhere in the city. Dividing the dataset in areas that do not benefit the size of the same or the number of bikes contained is crucial. This can lead to less bicycle variance between time periods and then faster and more precise predictions.

We started with the use of the ARIMA method, that was implemented as can be seen on Snippet 3.11. ARIMA is not the best solution for our scope, but it is a good method to try and learn. A bad thing about it is the time for the processing. It takes hours to process the clusters generated with the Padua dataset. With ARIMA is it possibile to see immediately the need of using different evaluation metrics and not only one. This is because it tends to follow the mean of the dataset passed in input, but this can be good or bad, depending on the field in which it is applied to. In our case this is not the behavior we are trying to investigate, because follow only the mean can give us false results. If you observe the result at macro level, it is near to the perfection, meanwhile if you zoom in you can see a lot of variance in the prediction of a single time period. ARIMA tends to lean too much on the value of the previous hour and if the

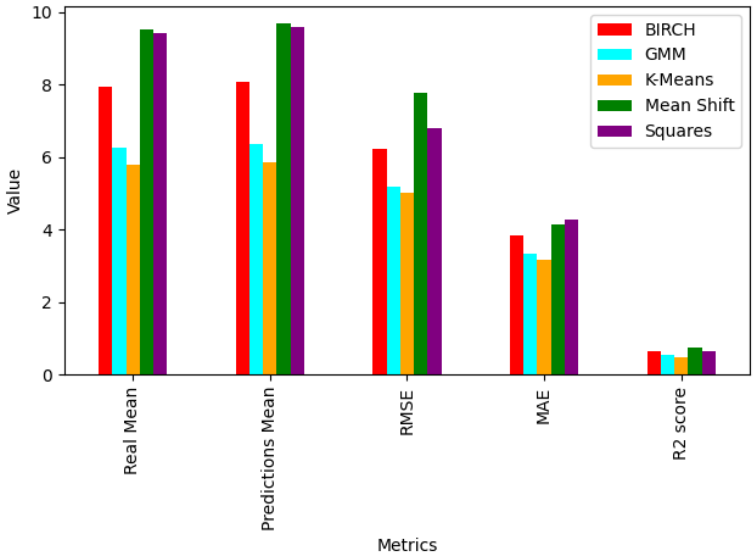

Figure 4.14: Results of top 5 of 16 clusters with ARIMA on Padua

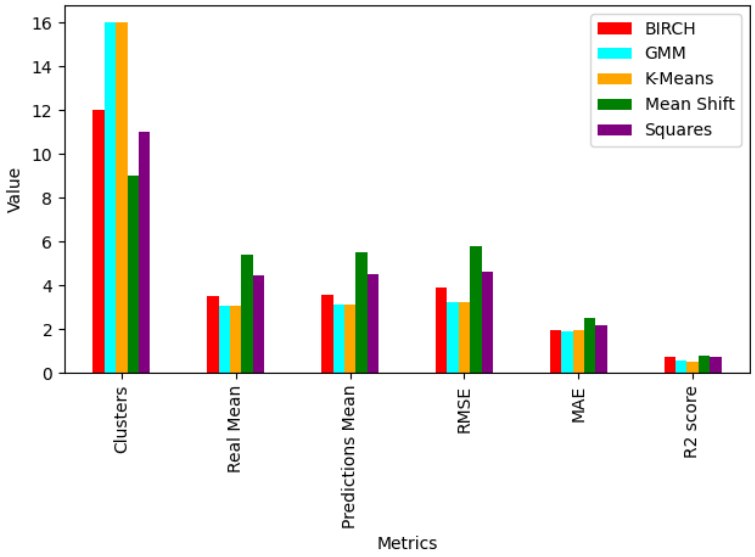

Figure 4.15: Results of regions with at least 90 items over 16 clusters with ARIMA on Padua

number of bikes has an increasing or decreasing spike from a period to another, ARIMA will predict it wrong surely by a fairly high value.

It is more clear with the analysis of the results showed in Figures 4.14, 4.15 and 4.16 (more precise numeric results can be seen in Tables A.5, A.6 and A.7). The predicted means are very close to the real one and the R2 score is not bad, mostly with Mean Shift and Squares approach. But, as described before, the

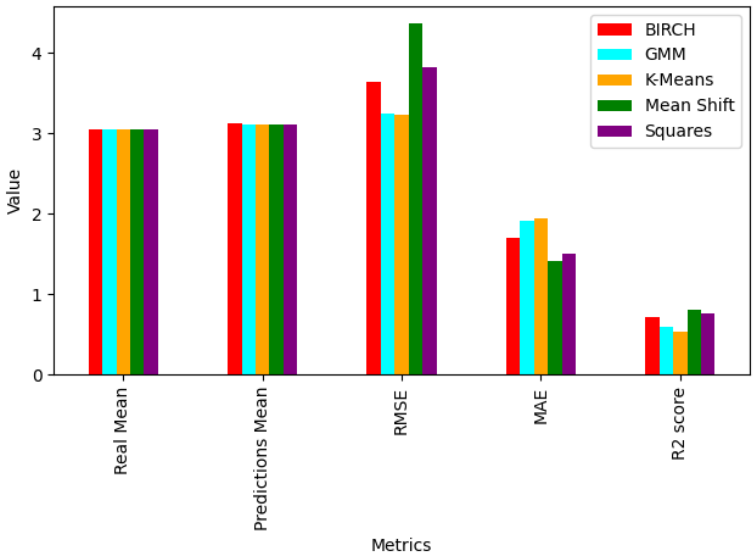

Figure 4.16: Results of 16 clusters with ARIMA on Padua

R2 score can lead to a false evaluation if it is taken alone. Observing the RMSE and MAE, it can be noticed that the higher values are from the Mean Shift and Squares again. These are indices of a not so good prediction. As in Section 4.1, the best clustering for the ARIMA method are again the K-Means and GMM, but this time with a not so clear winner, given that if we analyse the RMSE and MAE the K-Means is better, but the R2 score tells us otherwise, crowning the GMM. A small note is that only for 16 clusters, in both of the three cases we took in examination (top 5 clusters, clusters with at least 90 bikes and all clusters), the RMSE is similar to the mean. This will not happen in the next cases. For a better understanding, we have to watch the results with 36 and 64 clusters.

With 36 clusters, referring to Figures 4.17, 4.18 and 4.19 generated from Tables A.8, A.9 and A.10, in the top 5 clusters case, the results are similar to the 16 clusters approach. Taking into account the regions with at least 90 bikes, there is immediately a note to consider. GMM and K-Means have all the clusters with a good amount of data, meanwhile the other clustering algorithms lose from 10 to 20 clusters. This is even more pronounced in the 64 clusters case. Considering 36 regions, the RMSE takes a boost, increasing to nearly double of the mean. This means that the predictions are not very good, because they deviate by a value nearly equal to two times the mean from the real value.

In the last case for ARIMA, with 64 clusters, for the top 5 clusters the results are similar to the others. Referring to Figures 4.20, 4.21 and 4.22 with the

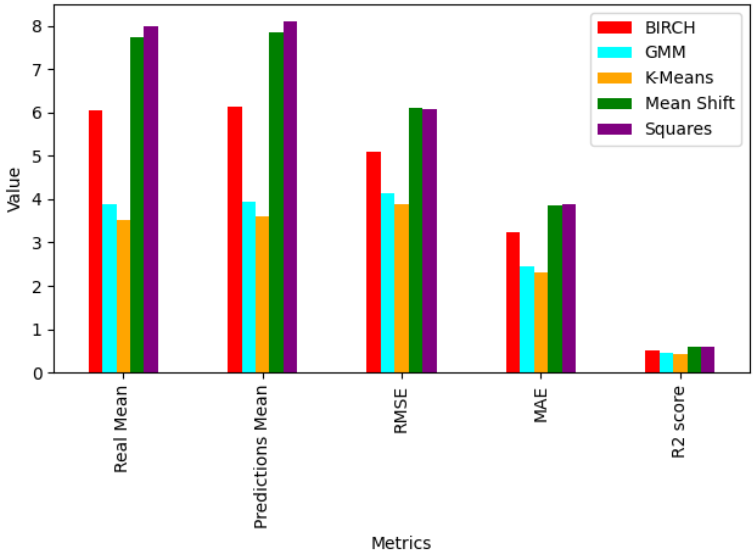

Figure 4.17: Results of top 5 of 36 clusters with ARIMA on Padua

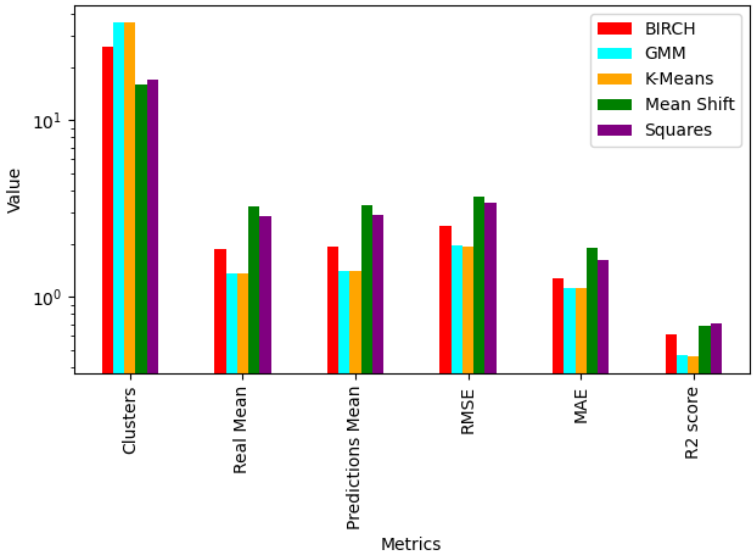

Figure 4.18: Results of regions with at least 90 items over 36 clusters with ARIMA on Padua

data from Tables A.11, A.12 and A.13, the clusters deleted by the threshold filter applied in Snippet 3.9, are more the 50% for the Squares and Mean Shift approaches, and a little less for BIRCH. The other 2 cases are similar. In both the RMSE reaches a higher value with respect to the mean. Overall, the GMM gives to us a better result than the K-Means, losing in the RMSE, but winning in the MAE and R2 score.

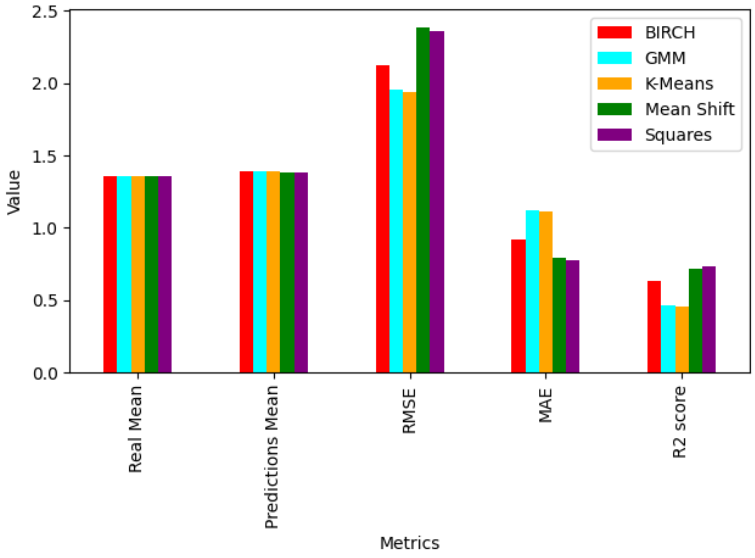

Figure 4.19: Results of 36 clusters with ARIMA on Padua

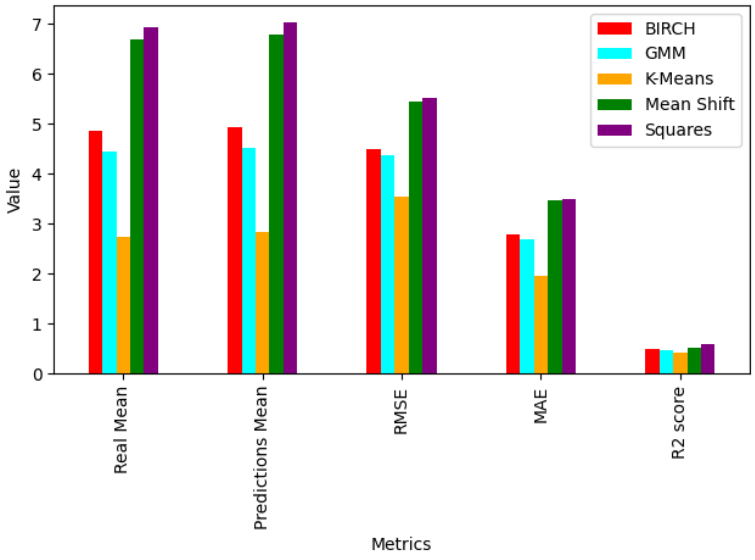

Figure 4.20: Results of top 5 of 64 clusters with ARIMA on Padua

For giving a overall look at the results, we can observer Figures 4.23 and 4.24 that represent the 2 most important evaluation metrics in our study. As stated before, the RMSE follows the mean, lowering as the number of clusters decreases and with the K-Means constantly below the other algorithms. The R2 score has the same trend, with the Mean Shift and the Squares battling for the higher position in the podium. As said before, this can cause a blander, because the same algorithms have the MAE and RMSE above the others in every case.

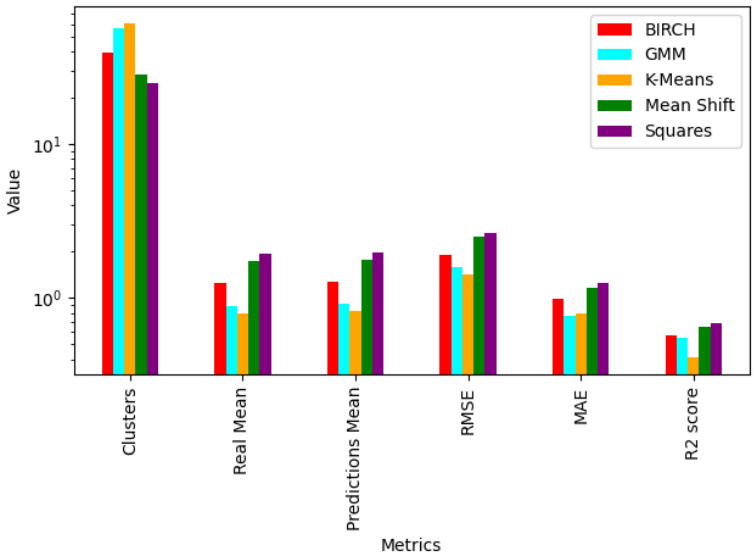

Figure 4.21: Results of regions with at least 90 items over 64 clusters with ARIMA on Padua

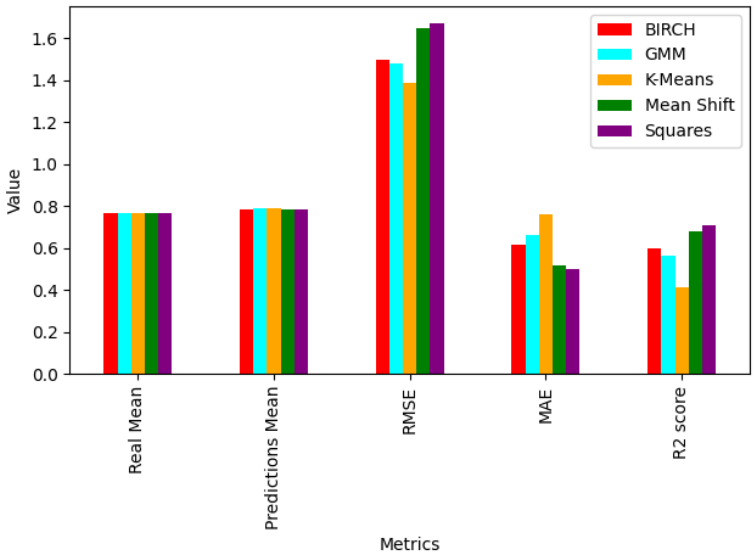

Figure 4.22: Results of 64 clusters with ARIMA on Padua

So, the R2 score metric alone can not considered as sufficient to give value to the results. In conclusion, ARIMA is not the best prediction method for the scope of this thesis. However, considering this approach, with a normal high number of clusters for the dataset, the best clustering algorithm is the GMM, returning not the higher R2 score, but a good overall result considering the RMSE, MAE and the predictions mean.

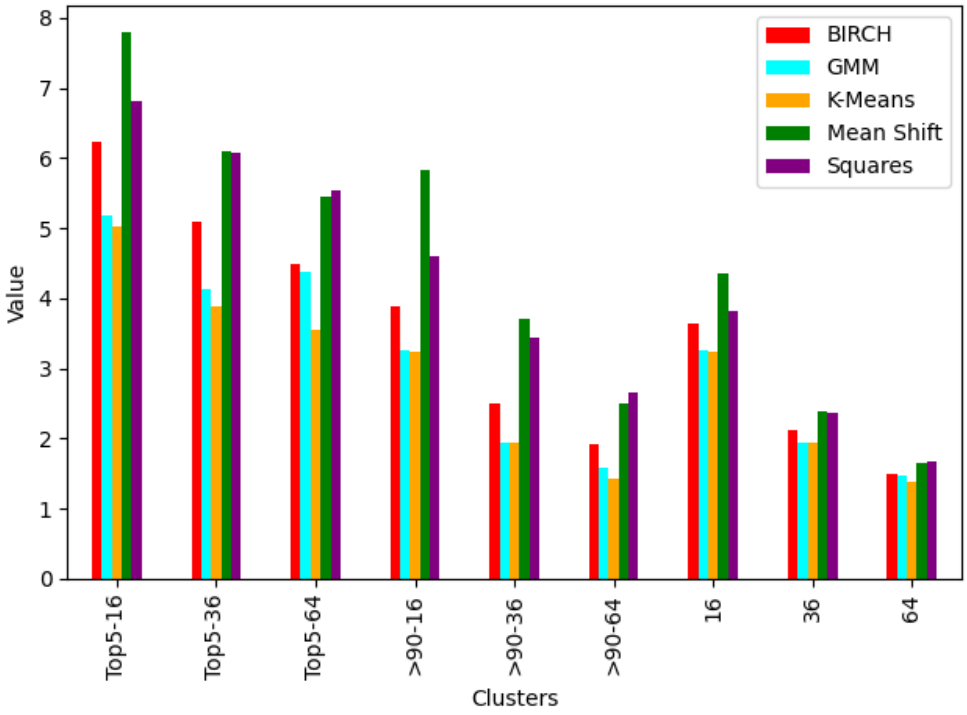

Figure 4.23: Comparison of RMSE results with ARIMA on Padua

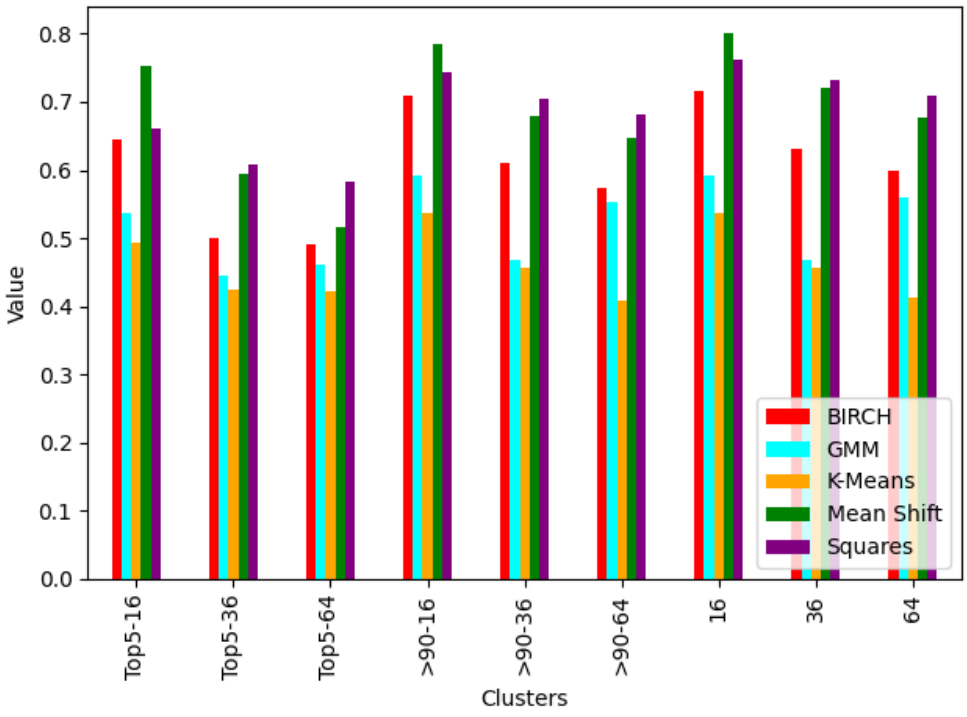

Figure 4.24: Comparison R2 score comparison with ARIMA on Padua

Now, it is important to consider the XGBoost approach, implemented as shown in Snippet 3.12. The most impressive thing about this method is the execution time. XGBoost takes very few minutes, if not seconds, for processing the Padua dataset, with optimal results as we will see next.

Considering the 16 clusters case, showed with histograms in Figures 4.25, 4.26 and 4.27, generated from Tables A.14, A.15 and A.16 respectively, we can immediately see a big difference from ARIMA. The RMSE is way below the mean and the R2 score is above the 0.800 with all the clustering techniques. This indicates that the predictions are good for all the clustering methods. In this case, noticing all the 3 types of filtering, the GMM wins right away, having a lower RMSE and MAE, but a higher R2 score compared to the K-Means.

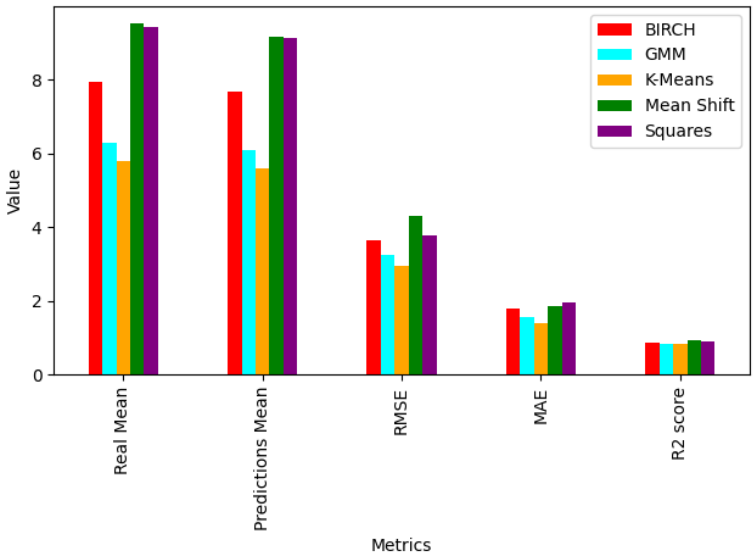

Figure 4.25: Results of top 5 of 16 clusters with XGBoost on Padua

In the 36 clusters case, represented with Figures 4.28, 4.29 and 4.30, obtained from Tables A.17, A.18 and A.19, the RMSE is close to the mean, but the R2 score is still very high. In this case too, GMM wins, considering all the aspects, but by a very little to the K-Means.

In the 64 clusters case, showed by Figures 4.31, 4.32 and 4.33, that represent Tables A.20, A.21 and A.22, the RMSE is still close to the mean and the R2 score is high, but not as before. In this case too, GMM wins, considering all the aspects, but by a very little to the K-Means. An important note to take into account, is the fact that all the clustering techniques return to us very good results with XGBoost highlighting efficiency and precision of this forecasting method. For

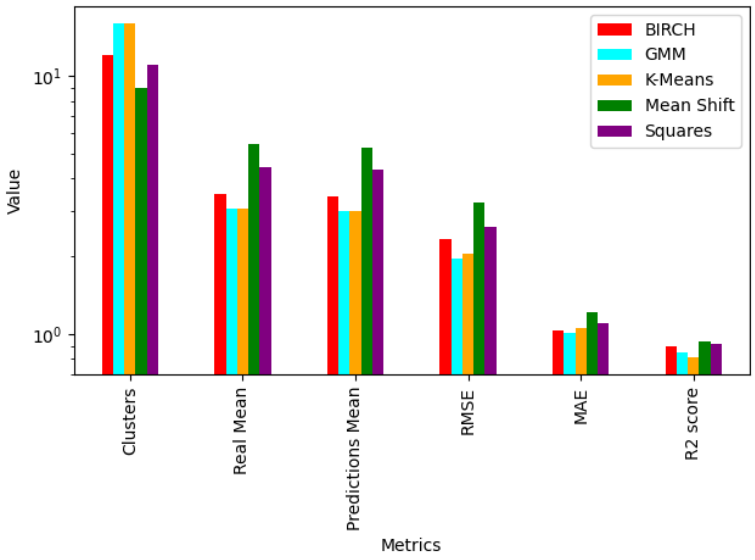

Figure 4.26: Results of regions with at least 90 items over 16 clusters with XGBoost on Padua

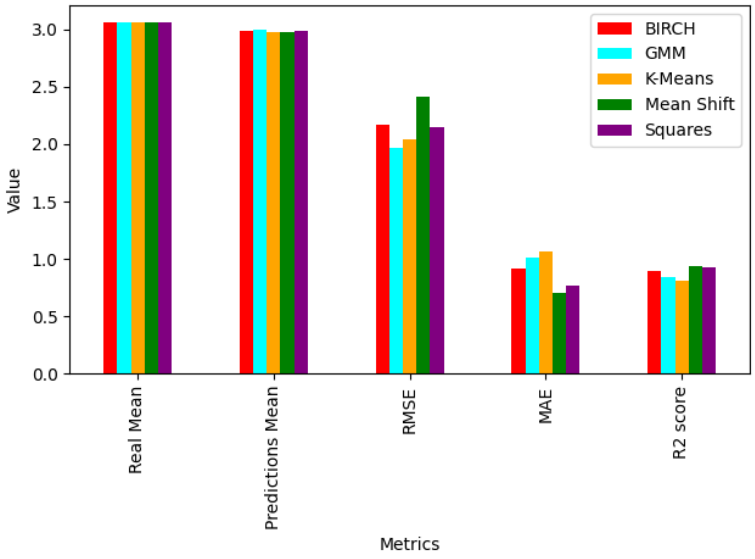

Figure 4.27: Results of 16 clusters with XGBoost on Padua

declaring the best clustering algorithms, we concentrated again in the ones with the best regions partitioning.

To conclude the Padua section, we can say that XGBoost is way better compared to ARIMA, considering the evaluation metrics and the processing times that can be seen in Figures 4.34 and 4.35. XGBoost gives us similar results with all the clustering methods and numbers, crowning GMM as the best clustering

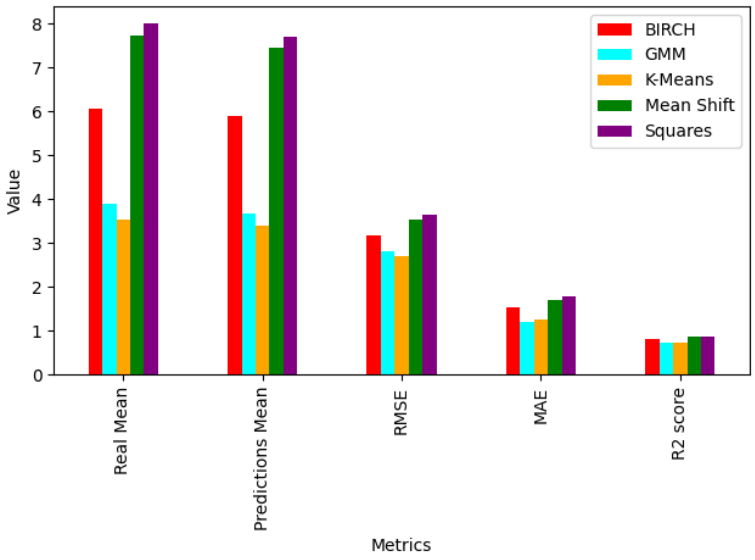

Figure 4.28: Results of top 5 of 36 clusters with XGBoost on Padua

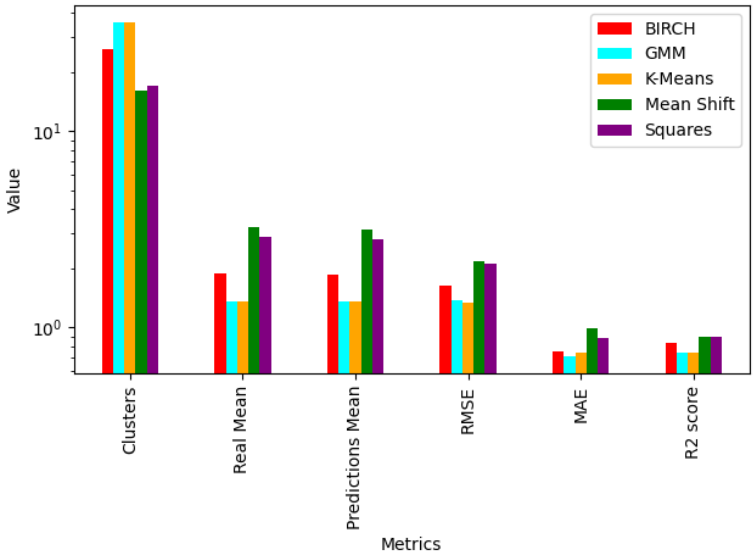

Figure 4.29: Results of regions with at least 90 items over 36 clusters with XGBoost on Padua

algorithm to accomplish the aim of this thesis. K-Means is also a very good alternative, but usually produces a lower R2 score. BIRCH produces a good RMSE, MAE and R2 score, similar to the K-Means and GMM ones, but it loses by a lot in terms of creating regions, with too many almost empty zones. The last two, Squares and Mean Shift are very similar, because they give a lot of importance to the outer zones. This gives as result a higher R2 score, given that predicting

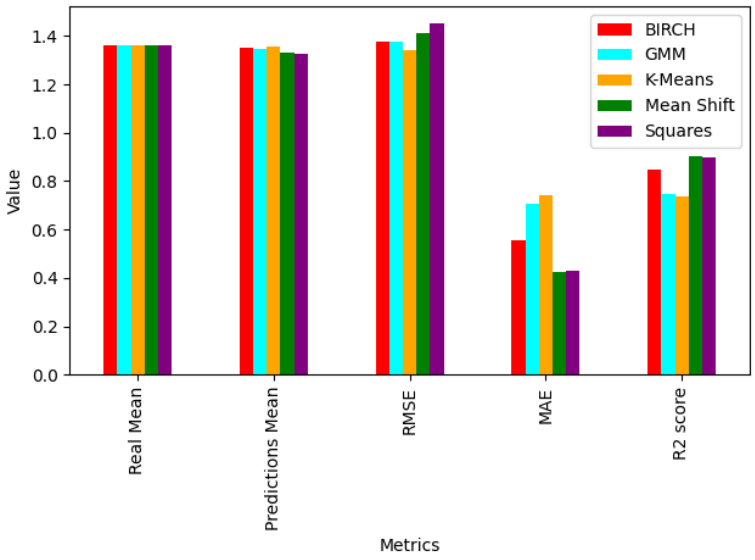

Figure 4.30: Results of 36 clusters with XGBoost on Padua

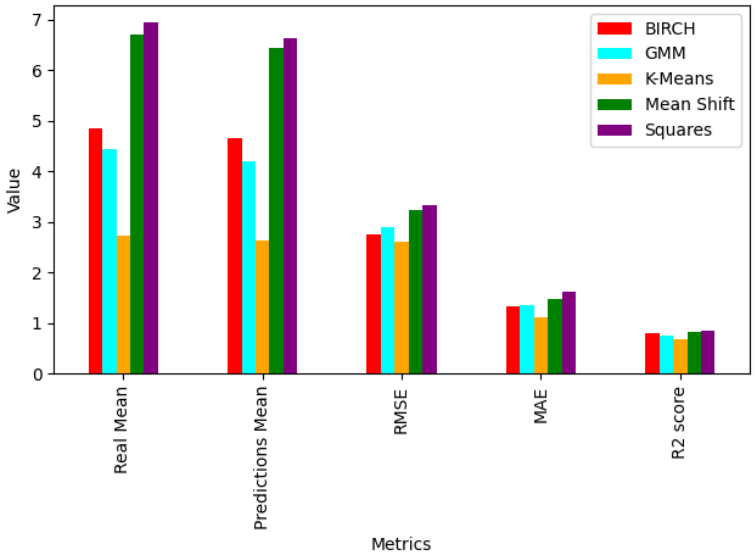

Figure 4.31: Results of top 5 of 64 clusters with XGBoost on Padua

regions with nearly 0 items is easy, because the prediction will be almost always 0. This will falsify the results, but it will be balanced with the center clusters that have a very high number of bikes and are so difficult to predict, a fact that raises the RMSE and MAE.

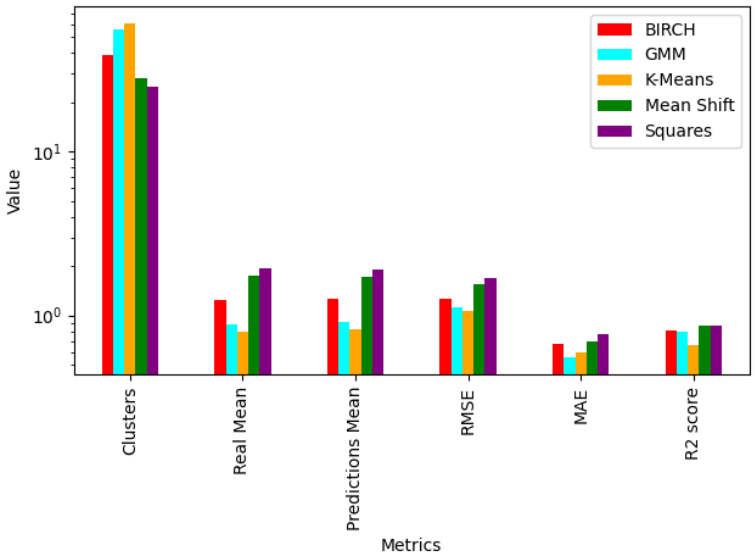

Figure 4.32: Results of regions with at least 90 items over 64 clusters with XGBoost on Padua

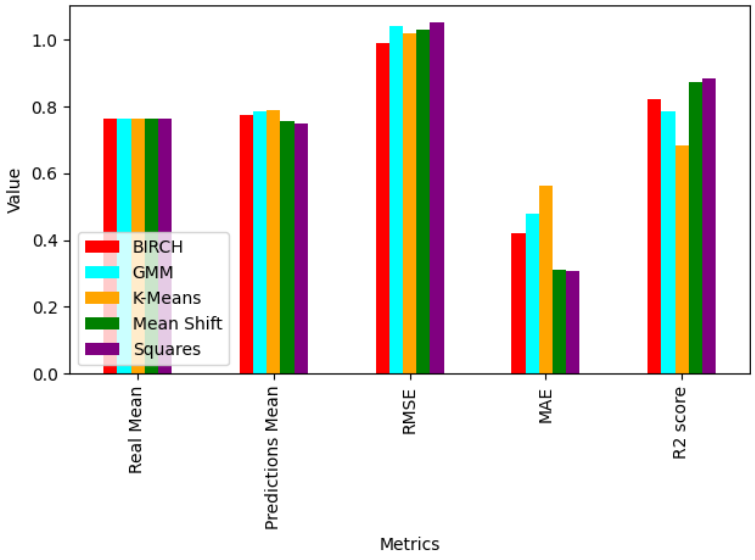

Figure 4.33: Results of 64 clusters with XGBoost on Padua

4.3.2 Montreal

In the Montreal case, there are some differences with respect to Padua. The clustering process is done using the station data, so the amount of points is very low and the items are distributed all over the city.

We started again with the ARIMA method. The results are showed in Figures

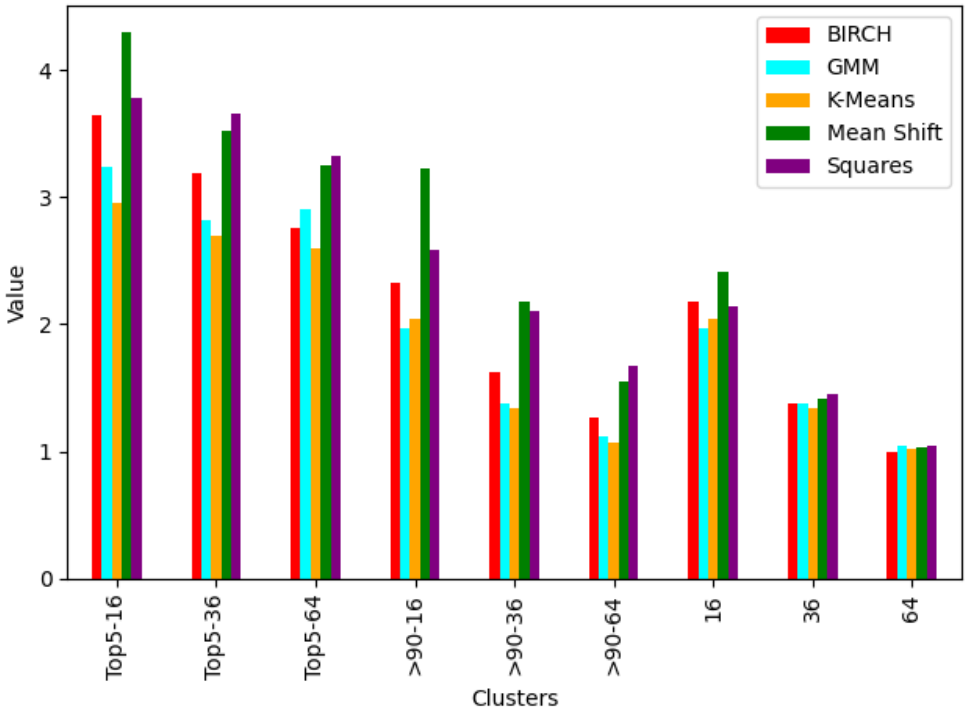

Figure 4.34: Comparison of RMSE results with XGBoost on Padua

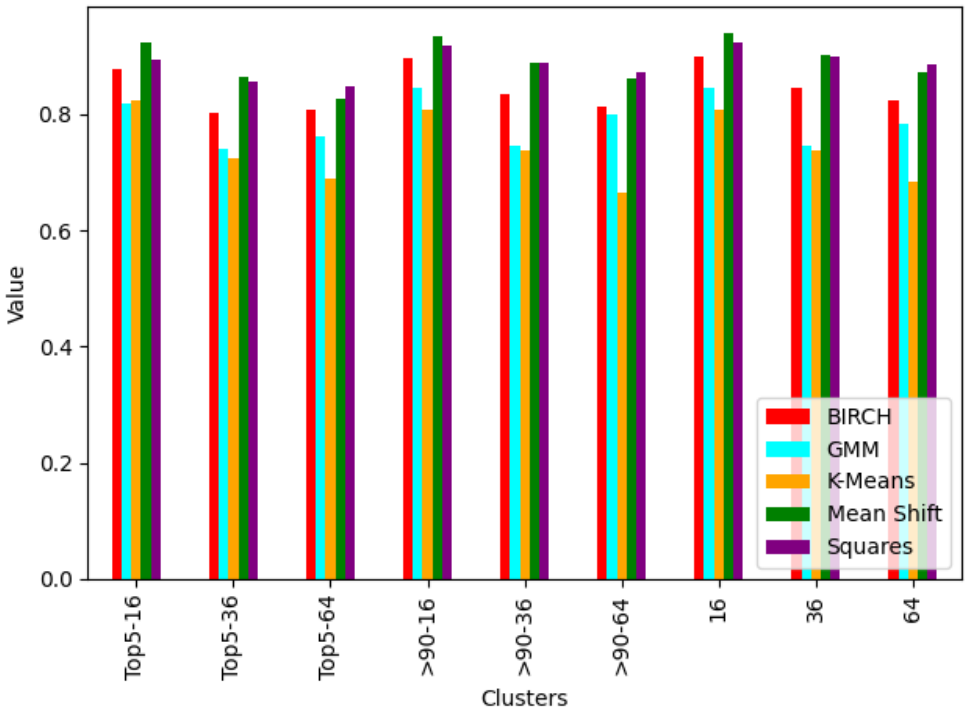

Figure 4.35: Comparison R2 score comparison with XGBoost on Padua

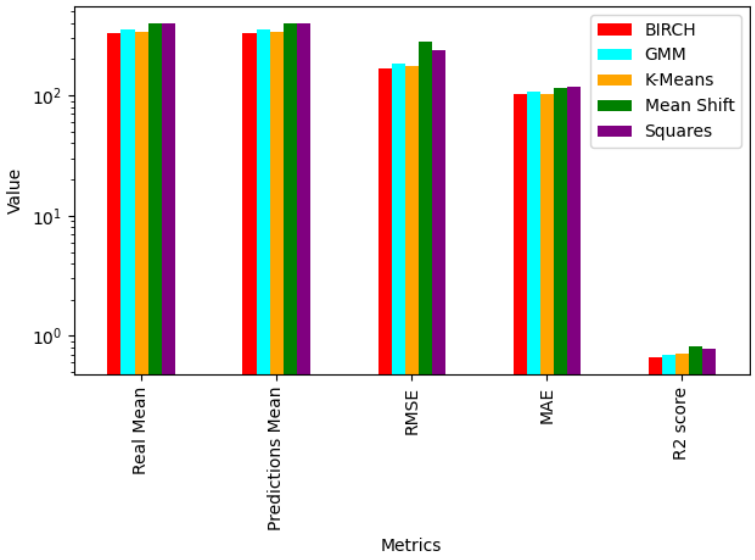

Figure 4.36: Results of top 3 of 9 clusters with ARIMA on Montreal

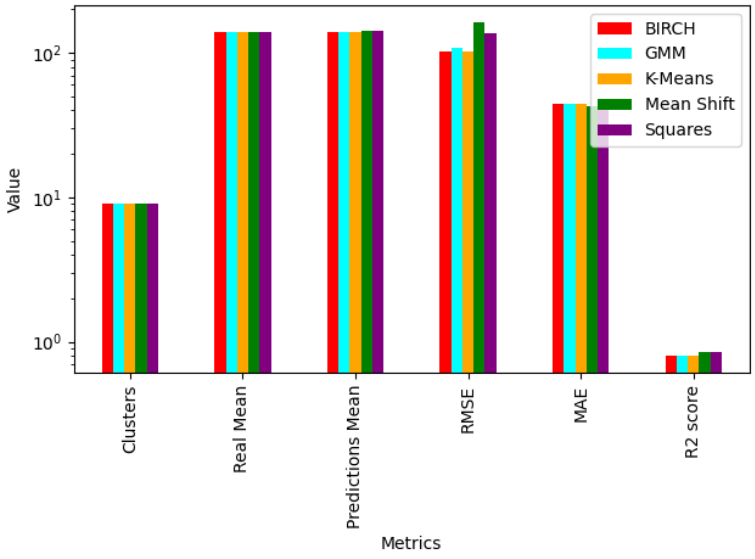

Figure 4.37: Results of regions with at least 120 items over 9 clusters with ARIMA on Montreal

4.36, 4.37 and 4.38 linked to Tables A.26, A.27 and A.28. They are pretty similar to the Padua ones, with GMM and K-Means with the more balanced RMSE, MAE and R2 score ratio, but BIRCH is about on the same level. As before, Squares and Mean Shift tend to have a higher R2 score, but a higher RMSE and MAE too.

With 16 clusters, referring to Figures 4.39, 4.40 and 4.41 generated from Tables

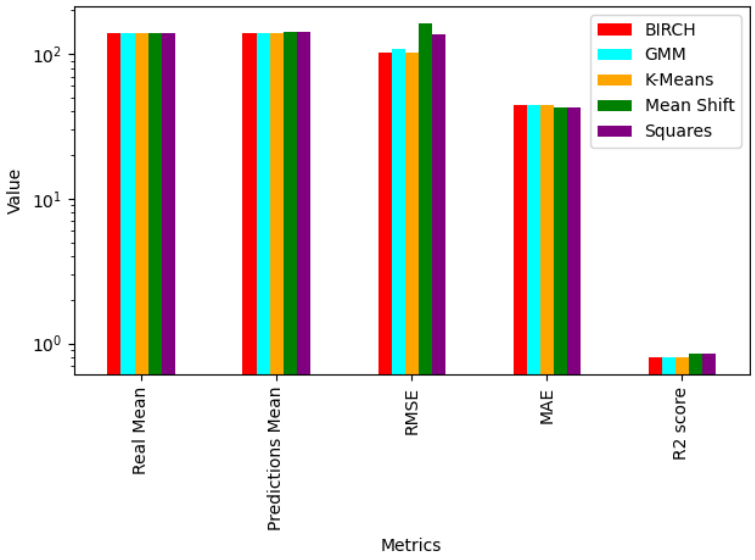

Figure 4.38: Results of 9 clusters with ARIMA on Montreal

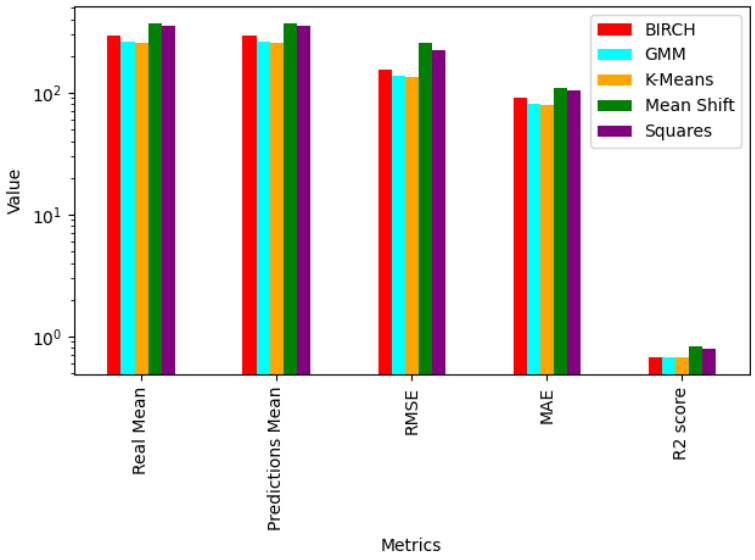

Figure 4.39: Results of top 3 of 16 clusters with ARIMA on Montreal

A.29, A.30 and A.31, the trend continues, with the RMSE being lower or equal to the mean in all of the cases. A difference from Padua, is that the only clustering techniques dropping regions with the application of the threshold is Squares. All the others still the whole dataset partitioned. The best clustering position is still contended by BIRCH, K-Means and GMM, with all 3 having good RMSE and R2 score.

Referring to Figures 4.42, 4.43 and 4.44 with the data from Tables A.32, A.33

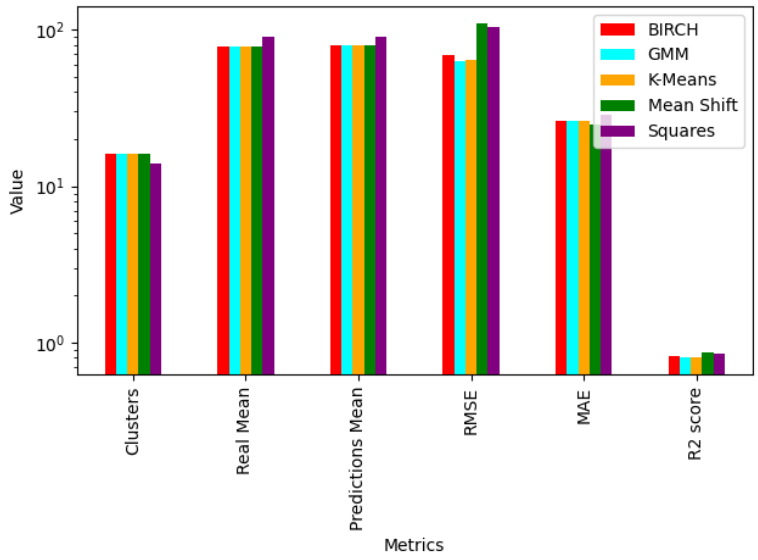

Figure 4.40: Results of regions with at least 120 items over 16 clusters with ARIMA on Montreal

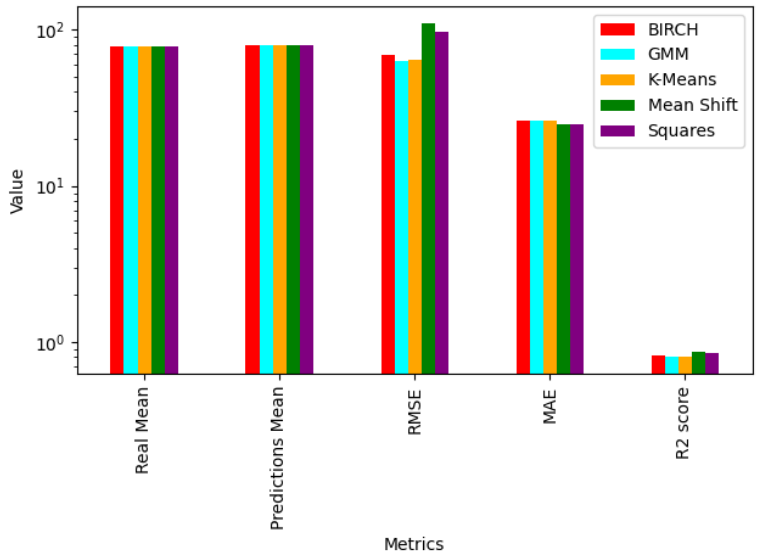

Figure 4.41: Results of 16 clusters with ARIMA on Montreal

and A.34, the threshold filter applied in Snippet 3.9 still does not apply any effect in the real clustering techniques. Also in this cases, we have the same results as before. Considering ARIMA on Montreal, there is not a specific best algorithm, but K-Means, GMM and BIRCH are all very similar, with GMM gaining a little bit of advantage observing the items per cluster, but this is not a fundamental evaluation metric.

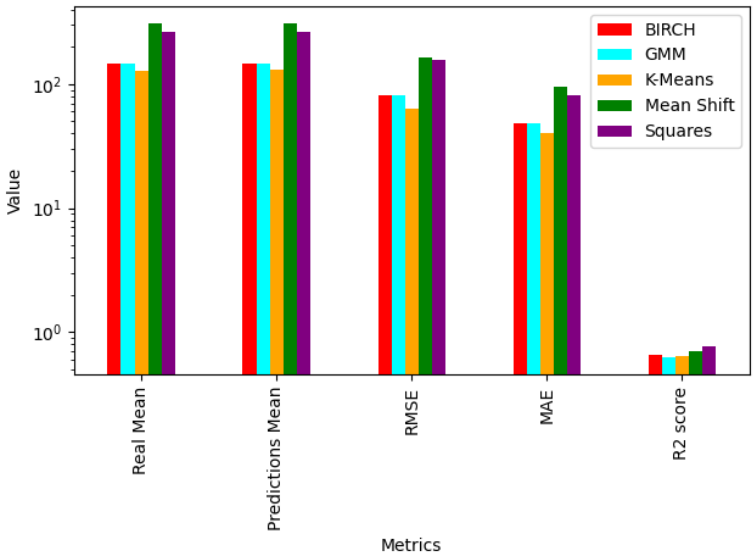

Figure 4.42: Results of top 3 of 36 clusters with ARIMA on Montreal

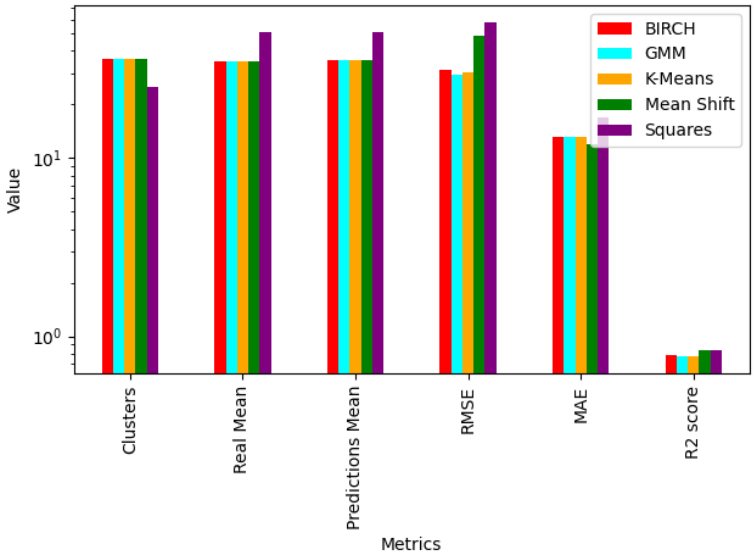

Figure 4.43: Results of regions with at least 120 items over 36 clusters with ARIMA on Montreal

Observing Figures 4.45 and 4.46, we can see that the RMSE follows a decreasing trend grouped on 3 sets of steps each. As with Padua dataset, the K-Means is the best one, but this time GMM is nearly equal or lower in the cases with higher number of regions. The R2 score is very high in all cases, proving that the clustering on few and already spaced out station, gives better results compared to the same done on sparse points.

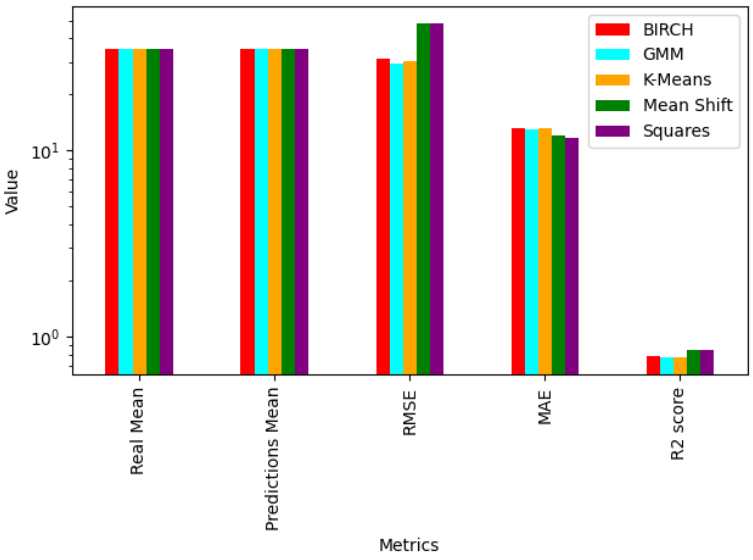

Figure 4.44: Results of 36 clusters with ARIMA on Montreal

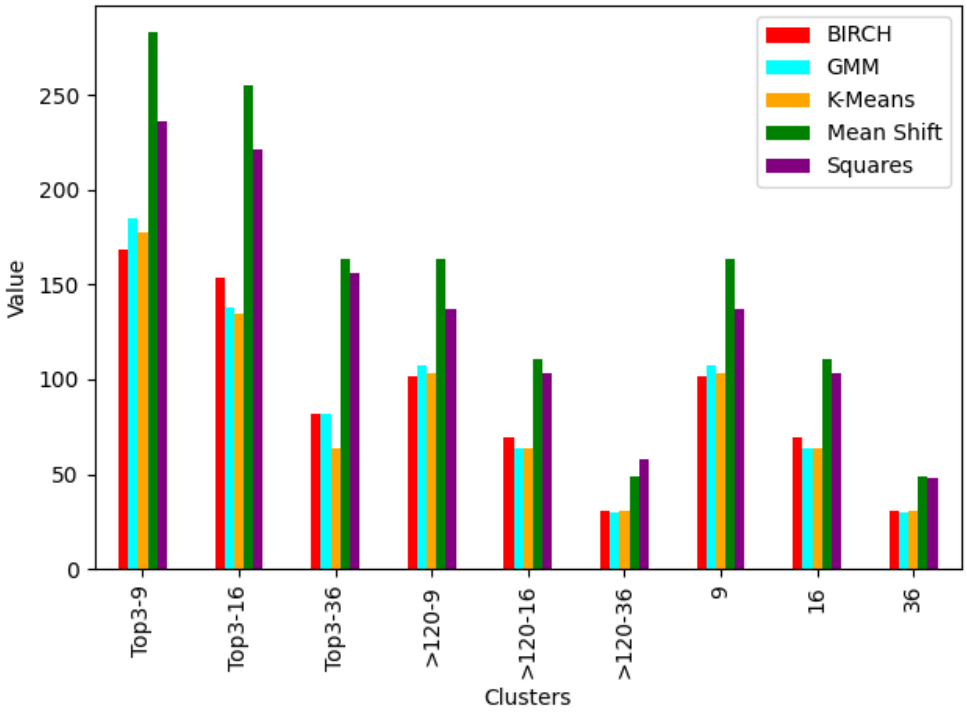

Figure 4.45: Comparison of RMSE results with ARIMA on Montreal

We have to use XGBoost to see more marked differences. Considering the 9 clusters case, that can be observed in Figures 4.47, 4.48 and 4.49, linked to Tables A.35, A.36 and A.37 respectively, we can see a small change from ARIMA. The

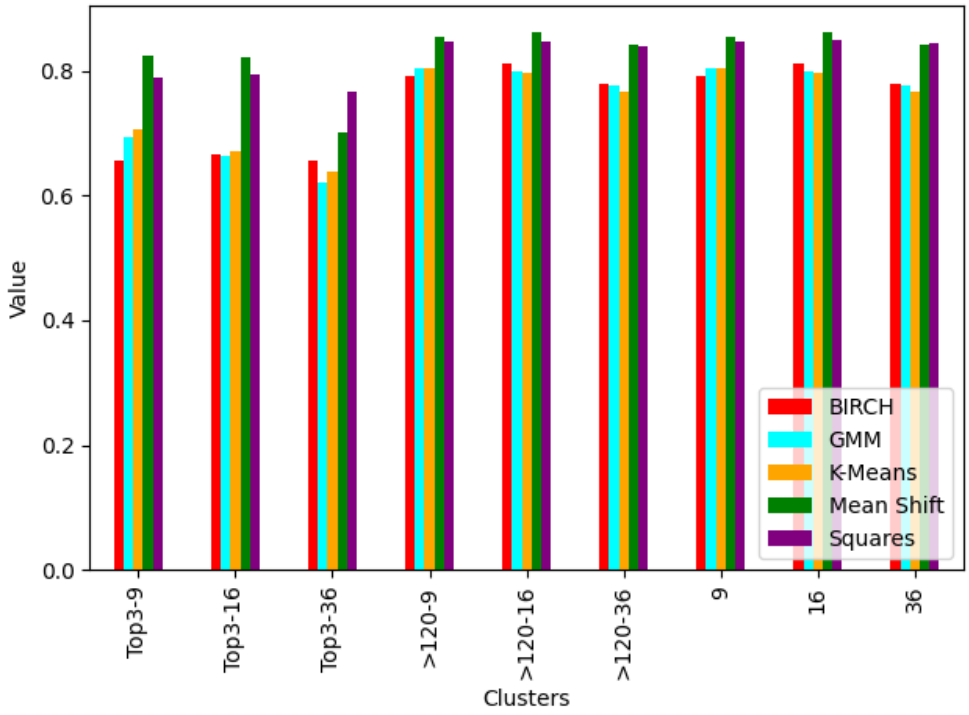

Figure 4.46: Comparison R2 score comparison with ARIMA on Montreal

overall result are very similar to the previous, with higher R2 score for every algorithm. But with 9 clusters, K-Means has a little bit of advantage in terms of RMSE, MAE and R2 score compared to the usual GMM and BIRCH. The RMSE still is below the mean for all the 5 algorithms.

In the 16 clusters case, generated from Tables A.38, A.39 and A.16 and showed with Figures 4.50, 4.51 and 4.52, the GMM and K-Means seem to be the same clustering, apart from the top 3 case, in which K-Means is a little bit better. Overall, the result is always the same as before.

In the last case with 36 clusters, represented by Figures 4.53, 4.54 and 4.55, that represent Tables A.41, A.42 and A.43, the RMSE is still close to the mean, but below and the R2 score is again very high. Still we do not have a clear winner, with the highest place in the podium contended by GMM and K-Means, and the third taken by BIRCH.

Based on Figures 4.56 and 4.57, in the RMSE case, we can see the differences between the clustering algorithms that try to create regions with a similar number of elements with the others. Squares and Mean Shift have a very high RMSE, which means that it is not very convenient to use such methods for creating areas with the Montreal dataset. The R2 score is very constant in every case, with all

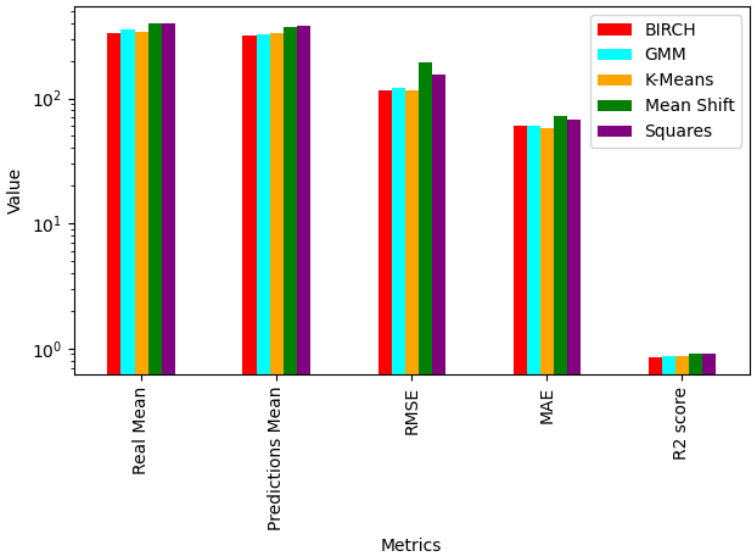

Figure 4.47: Results of top 3 of 9 clusters with XGBoost on Montreal

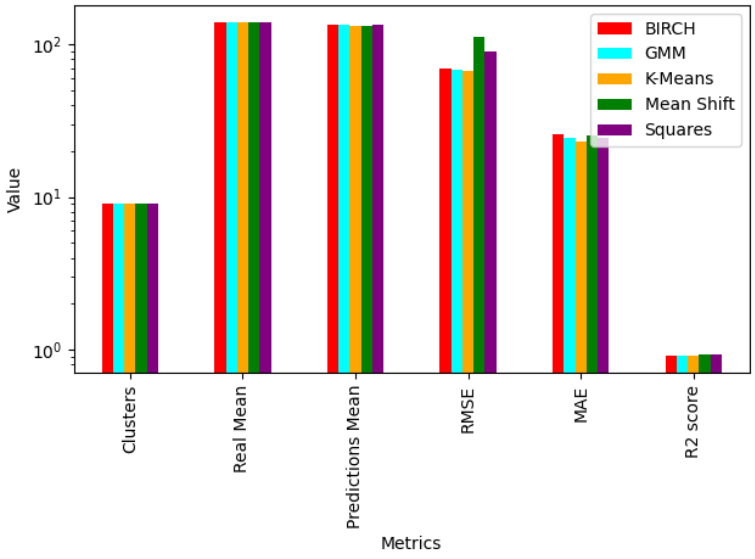

Figure 4.48: Results of regions with at least 120 items over 9 clusters with XGBoost on Montreal

the 5 algorithms very near to each other.

To conclude the Montreal section, we can confirm that XGBoost is better than the ARIMA method again, because it tends to have the RMSE always below the related mean, a lower MAE and a very high R2 score for all the clustering algorithms. But this time, we can not declare the best clustering. This is caused by the dataset. In Padua case, we have items that can be placed randomly in the

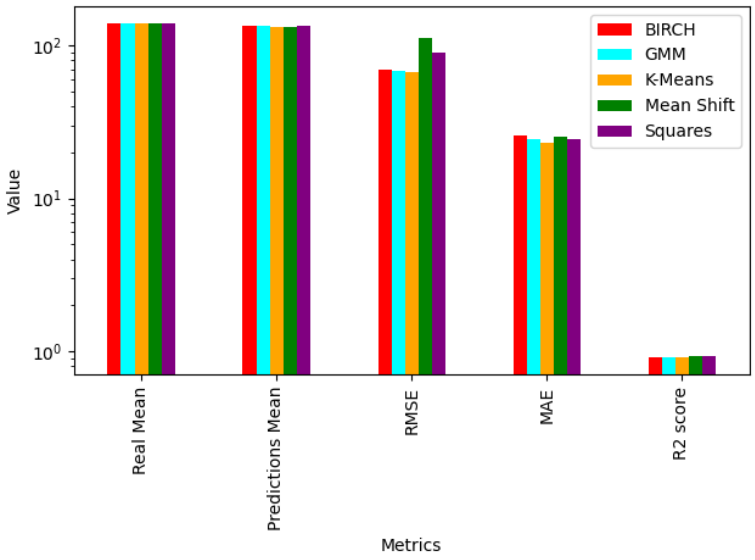

Figure 4.49: Results of 9 clusters with XGBoost on Montreal

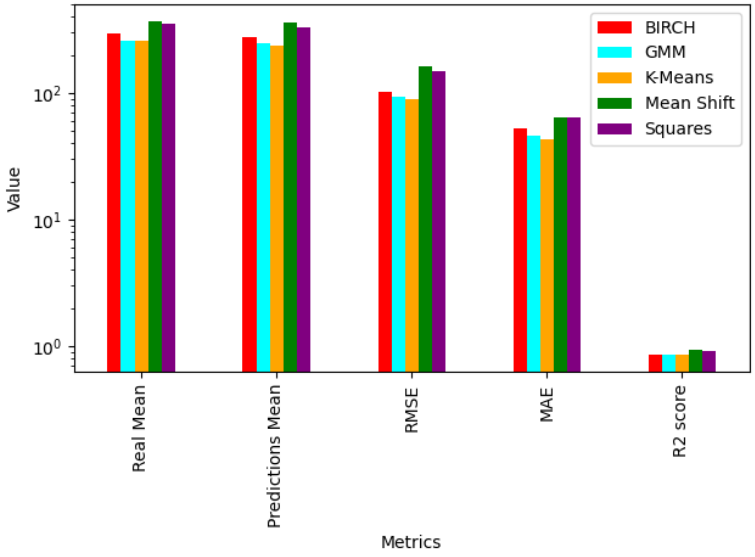

Figure 4.50: Results of top 3 of 16 clusters with XGBoost on Montreal

city, meanwhile in Montreal the starting and ending point of a bike travel is fixed, and the clustering application is a lot easier. We can however say, that GMM is the best algorithm, because with both dataset it tends to be really efficient and reliable and, apparently, works best with some noise that is the normal case in our situation using a floating bike sharing system.

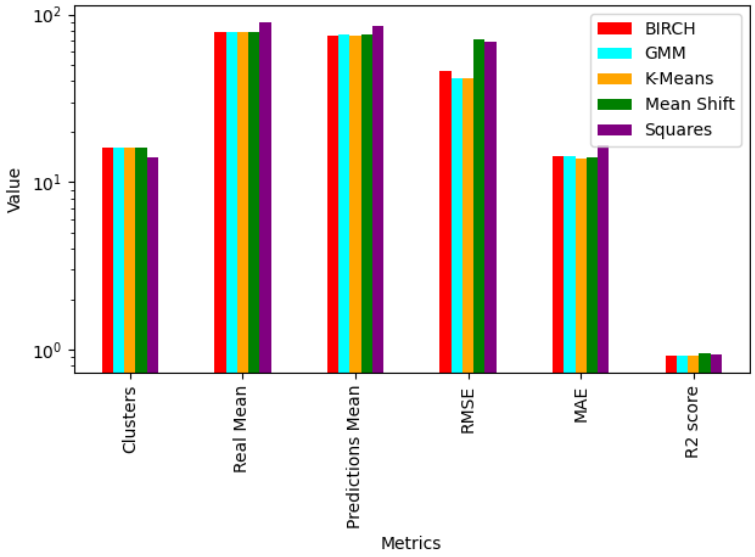

Figure 4.51: Results of regions with at least 120 items over 16 clusters with XGBoost on Montreal

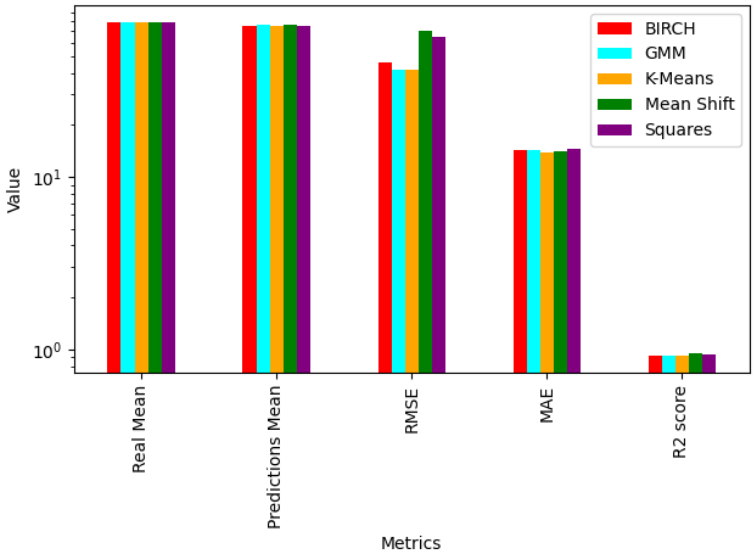

Figure 4.52: Results of 16 clusters with XGBoost on Montreal

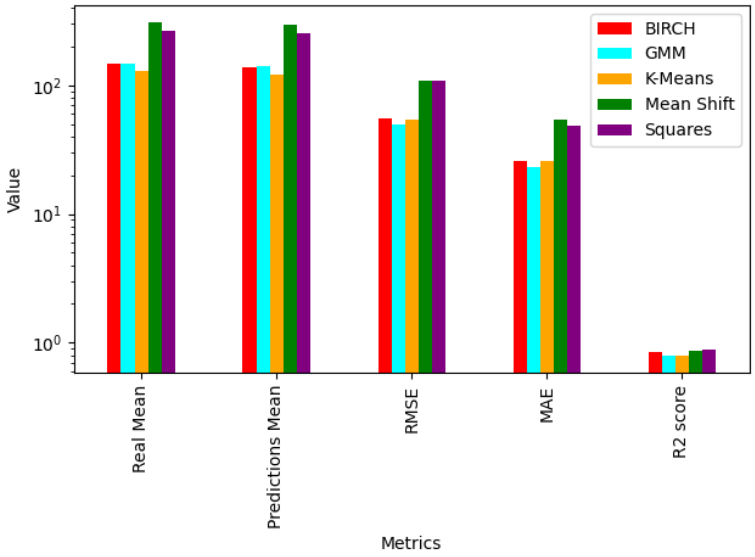

Figure 4.53: Results of top 3 of 36 clusters with XGBoost on Montreal

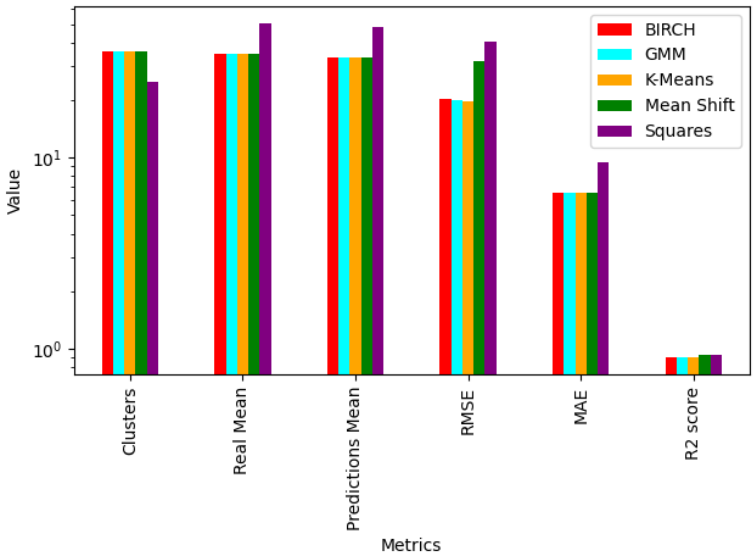

Figure 4.54: Results of regions with at least 120 items over 36 clusters with XGBoost on Montreal

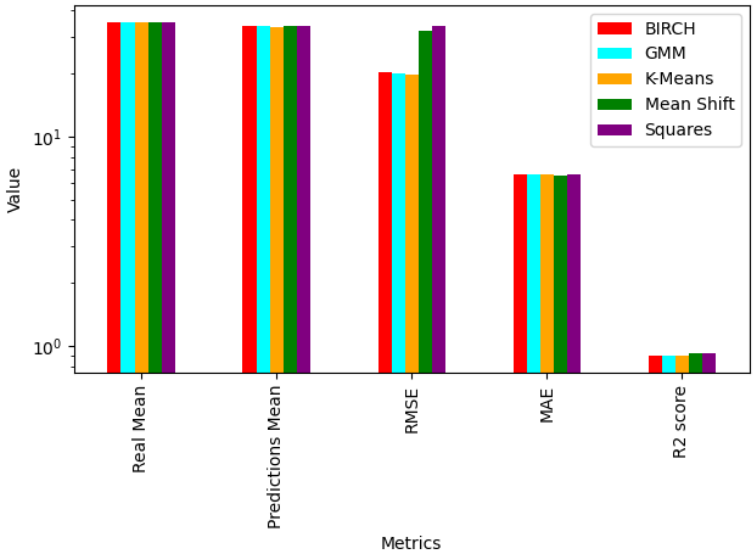

Figure 4.55: Results of 36 clusters with XGBoost on Montreal

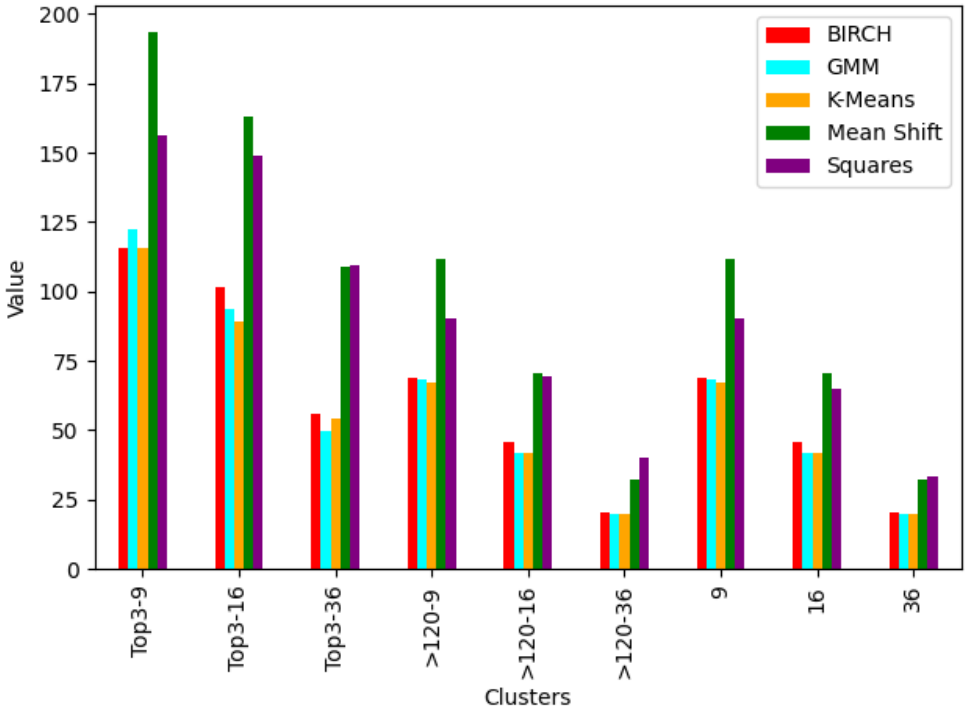

Figure 4.56: Comparison of RMSE results with XGBoost on Montreal

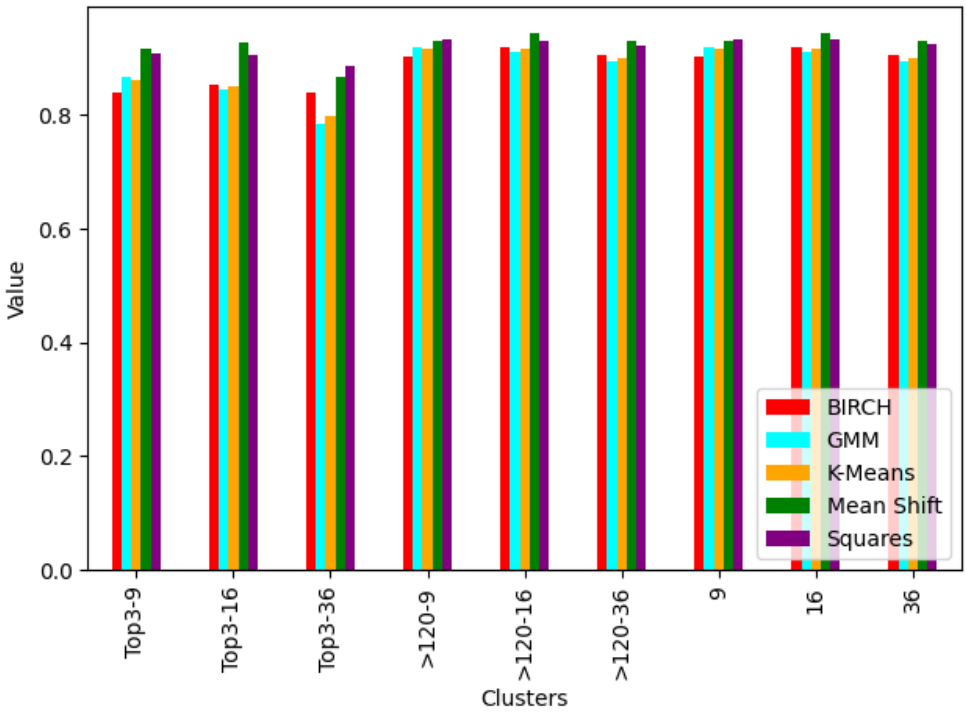

Figure 4.57: Comparison R2 score comparison with XGBoost on Montreal

5

Conclusions and Future Works

The aim of this thesis was to analyze and make more efficient the floating bike sharing system of Padua to make it more eco-sustainable. For making this happen, we used different clustering techniques, to have as many choices as possible about which method is the best. After that, we wanted to predict the number of bikes that have to be available in every cluster in every hour of a day. Using ARIMA and XGBoost, we have given very credible results, supported by different evaluation metrics that have been compared between the various clustering algorithms implemented. Using different datasets and bike sharing systems, helped us to analyzing data from different sources, in order to clarify the results and the various comparison metrics used. We showed the differences between the various clustering techniques and furthermore with the forecasting methods implemented. We reached the conclusion that XGBoost is better in terms of efficiency, execution time and quality of results with the respect to the ARIMA approach. With respect to the clustering algorithms, we consider GMM as the best, because with both dataset it tends to be really reliable and works best with some noise that is the standard case in our work environment using a floating bike sharing system. Furthermore, following our analyses and the processes carried out, we suggest to use different evaluation metrics to be able to observe the study environment from all angles, without neglecting things that we consider to be taken for granted or basic, but which in reality hide something that is falsified or that has no real meaning when considered alone. With the increasing of the clusters number, the differences between the clustering algorithms have become more clear, with the ones that

have a more homogeneous approach outdistanced the other algorithms quite clearly, above all with the use of ARIMA for forecasting. Our thesis showed how a step by step approach can lead to optimize and to make more efficient bike sharing systems. This workflow can be applied into different cities without any issue and it can be used to reduce air pollution caused by vans that move bikes or by people who are forced to take their own vehicle due to lack of them. The implementation is fast and not so difficult, mostly with the use of XGBoost as a forecasting method. It only needs a dataset of bike movements, that does not need a high number of items, because in our work we only used 3 months of data to train and test the whole environment.

During the realization, we encountered some problems with the implementations of specific clustering libraries or with the understanding of the results given by the statistical metrics. We have tried to implement more clustering techniques, like DBSCAN, OPTICS, ClustGeo and more, but our datasets do not fit very well with these type of algorithms. This was probably caused by the fact that DBSCAN is very powerful in cases with a lot of noise, such as very big cities or regions with an enormous density of items. This affects all the clustering based on the DBSCAN algorithm, just like OPTICS. Furthermore, at the start of the analysis of the predictions step, we did not understand well why some clustering algorithms that divided the datasets not very homogeneously, had a higher score compared to the K-Means or GMM. After a deep study we found out that it was given by the quantity of clusters with very few elements inside. The 90% of hours in these regions contained 0 bikes and so the predictions were easier to do. This situation falsified a lot of the results and for fixing it we applied different filtering methods to the clustering process.

For future works, we suggest to implement more clustering techniques, that can lead to different and better predictions. These can bring to a deeper understanding of how a clustering algorithm should work with this environment. Furthermore, it will be very useful to introduce a prediction system that takes into account the weather and differentiates the usual working day to the weekends or holidays. This can bring to very precise predictions for each day of the year, including all the possible cases that would lead the forecasting algorithms to be very wrong with respect to the real value.

Appendix Data tables

Clusters	Squares	K-Means	GMM	Mean Shift	BIRCH
16	0s	2.4s	3.01s	3263.91s	6.96s
36	0s	3.86s	3.45s	1655.61s	7.36s
64	0s	7.26s	16.57s	1448.71s	7.98s

Table A.1: Clustering processing times for Padua dataset

Cluster Number	Squares	K-Means	GMM	Mean Shift	BIRCH
0	124	7294	1076	57120	174
1	67	19395	6800	44341	33
2	9	10262	6554	10582	2727
3	1384	4277	15476	2364	18
4	21237	5258	14287	1577	43
5	36870	3798	4630	1385	12495
6	494	7772	5149	34	4835
7	423	6461	7667	73	2632
8	19715	7267	10717	17	9266
9	34872	2903	23939	8	20742
10	2532	4106	1010	717	11
11	20	2991	1553	46	25914
12	528	17578	1712	4	6829
13	1413	3540	3995	144	3370
14	37	13667	12189	1	6367
15	21	3177	2992	1333	24290

Table A.2: Number of bikes per region on 16 clusters on Padua dataset

Cluster Number	Squares	K-Means	GMM	Mean Shift	BIRCH
0	3	1657	333	23767	17731
1	38	15762	11125	29205	22126
2	19	2448	2439	20808	4128
3	2	1972	1850	9609	103
4	4	3795	2914	7954	109
5	7	3925	170	4235	7969
6	140	5436	7298	7462	1480
7	2706	2159	2055	2922	2534
8	1753	1325	2794	2663	50
9	24	6945	4493	4332	27
10	3	3383	15608	2515	92
11	347	1332	5377	1246	2605
12	3218	5254	3795	772	276
13	16304	1767	2997	177	12719
14	33479	2835	1020	974	3730
15	2143	2197	3554	45	2633
16	13	5460	2462	44	2066
17	49	1825	3174	22	27
18	2691	2270	3949	29	4089
19	14584	3976	1155	69	1214
20	24683	737	2074	759	5788
21	6582	1479	1239	11	164
22	182	2710	4986	18	38
23	7	2793	2530	8	53
24	242	2099	480	26	4152
25	3049	4442	9234	6	51
26	7184	868	2771	4	396
27	79	357	3391	5	5468
28	13	7944	1124	8	16
29	12	6934	2696	3	11
30	3	2665	2373	2	10
31	49	1332	4676	37	3
32	102	3211	836	3	730
33	29	2217	354	2	2299
34	0	2963	598	2	3180
35	3	1272	1822	1	11679

Table A.3: Number of bikes per region on 36 clusters on Padua dataset

Cluster Number	Squares	K-Means	GMM	Mean Shift	BIRCH
0	1	4189	15661	20961	1176

1	1	5480	12347	25229	10
2	31	14937	0	12776	4152
3	13	782	1287	12827	2568
4	4	898	256	8129	30
5	3	1838	2330	2604	1180
6	1	1305	1451	3285	11
7	1	1144	803	2976	3180
8	16	1195	2106	2336	27
9	16	2096	2343	2043	37
10	76	1319	9220	2355	1237
11	46	3274	593	5023	53
12	4	2224	2486	3219	396
13	5	2211	1298	1358	12719
14	0	1182	869	3768	266
15	16	1753	1156	1990	12662
16	439	1658	1023	882	2299
17	1368	1116	2403	3207	11679
18	3495	1118	696	825	68
19	3564	2684	1410	799	2281
20	286	1137	700	335	20
21	3	807	18	140	64
22	1	1692	5514	115	8
23	47	1361	1283	104	1638
24	882	4148	583	224	1123
25	2815	2294	8605	60	51
26	13559	1994	1347	659	134
27	29617	950	68	24	5468
28	3403	1077	3429	23	16
29	485	1932	74	1019	1443
30	5	4671	1708	19	596
31	3	281	102	19	3
32	278	2349	4797	17	1454
33	3563	1649	400	14	27
34	11065	1504	252	44	2578

35	20033	902	715	96	13
36	5363	1101	1288	12	771
37	2317	962	889	66	1910
38	38	2578	1978	8	6190
39	0	979	366	9	48
40	142	4053	146	15	39
41	2051	1223	0	9	357
42	3036	63	1617	8	19
43	9043	833	1111	26	24
44	433	1954	150	3	2988
45	172	396	1275	4	2800
46	5	92	1156	7	87
47	6	1158	0	3	3274
48	2	1478	294	3	1789
49	2	89	404	45	2300
50	486	1590	960	4	623
51	1285	21	2095	3	5391
52	84	1976	883	2	25
53	28	2033	7730	2	1179
54	8	1086	206	2	1204
55	10	265	75	1	1162
56	2	2044	1236	2	93
57	17	2378	1817	1	14679
58	23	1466	836	1	45
59	24	1056	343	2	38
60	20	849	0	1	1425
61	1	1440	2514	1	10
62	0	1517	708	1	29
63	3	3915	336	1	580

Table A.4: Number of bikes per region on 64 clusters on Padua dataset

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	9.418	5.781	6.277	9.514	7.931
Predictions mean	9.581	5.864	6.356	9.676	8.074
RMSE	6.802	5.026	5.190	7.786	6.231
MAE	4.283	3.163	3.336	4.145	3.838
R2 score	0.661	0.494	0.537	0.752	0.645

Table A.5: Evaluation results of top 5 clusters of 16 using ARIMA on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Clusters	11	16	16	9	12
Values mean	4.440	3.057	3.057	5.425	3.488
Predictions mean	4.521	3.112	3.122	5.529	3.566
RMSE	4.611	3.232	3.253	5.819	3.889
MAE	2.176	1.950	1.921	2.496	1.931
R2 score	0.744	0.536	0.591	0.784	0.709

Table A.6: Evaluation results of regions with at least 90 items over 16 clusters using ARIMA on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	3.057	3.057	3.057	3.057	3.057
Predictions mean	3.113	3.112	3.122	3.114	3.124
RMSE	3.823	3.232	3.253	4.365	3.639
MAE	1.505	1.950	1.921	1.413	1.698
R2 score	0.761	0.536	0.591	0.800	0.717

Table A.7: Evaluation results of 16 clusters using ARIMA on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	8.001	3.523	3.886	7.733	6.061
Predictions mean	8.089	3.610	3.954	7.846	6.139
RMSE	6.077	3.896	4.126	6.104	5.081
MAE	3.871	2.305	2.459	3.853	3.242
R2 score	0.607	0.424	0.444	0.594	0.501

Table A.8: Evaluation results of top 5 clusters of 36 using ARIMA on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Clusters	17	36	36	16	26
Values mean	2.869	1.358	1.358	3.252	1.876
Predictions mean	2.915	1.394	1.394	3.318	1.916
RMSE	3.434	1.935	1.951	3.698	2.500
MAE	1.625	1.115	1.120	1.881	1.267
R2 score	0.705	0.456	0.468	0.679	0.610

Table A.9: Evaluation results of regions with at least 90 items over 36 clusters using ARIMA on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	1.358	1.358	1.358	1.358	1.358
Predictions mean	1.381	1.394	1.394	1.386	1.387
RMSE	2.360	1.935	1.951	2.388	2.125
MAE	0.775	1.115	1.120	0.791	0.922
R2 score	0.733	0.456	0.468	0.720	0.632

Table A.10: Evaluation results of 36 clusters using ARIMA on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	6.941	2.732	4.447	6.706	4.861
Predictions mean	7.026	2.831	4.515	6.787	4.943
RMSE	5.532	3.550	4.376	5.443	4.497
MAE	3.488	1.954	2.693	3.466	2.797
R2 score	0.582	0.421	0.462	0.516	0.492

Table A.11: Evaluation results of top 5 clusters of 64 using ARIMA on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Clusters	25	61	56	28	39
Values mean	1.947	0.801	0.888	1.741	1.245
Predictions mean	1.991	0.828	0.914	1.782	1.280
RMSE	2.666	1.419	1.593	2.489	1.913
MAE	1.258	0.793	0.771	1.174	0.990
R2 score	0.681	0.409	0.552	0.647	0.573

Table A.12: Evaluation results of regions with at least 90 items over 64 clusters using ARIMA on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	0.764	0.764	0.764	0.764	0.764
Predictions mean	0.782	0.791	0.787	0.783	0.785
RMSE	1.668	1.386	1.477	1.647	1.496
MAE	0.499	0.758	0.664	0.519	0.614
R2 score	0.710	0.413	0.561	0.678	0.600

Table A.13: Evaluation results of 64 clusters using ARIMA on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	9.421	5.781	6.278	9.516	7.933
Predictions mean	9.148	5.594	6.102	9.169	7.683
RMSE	3.784	2.955	3.239	4.292	3.639
MAE	1.962	1.393	1.586	1.861	1.805
R2 score	0.895	0.825	0.820	0.925	0.879

Table A.14: Evaluation results of top 5 clusters of 16 using XGBoost on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Clusters	11	16	16	9	12
Values mean	4.442	3.058	3.058	5.427	3.489
Predictions mean	4.331	2.980	3.003	5.270	3.406
RMSE	2.587	2.047	1.968	3.221	2.324
MAE	1.105	1.060	1.013	1.217	1.030
R2 score	0.919	0.808	0.845	0.934	0.896

Table A.15: Evaluation results of regions with at least 90 items over 16 clusters using XGBoost on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	3.058	3.058	3.058	3.058	3.058
Predictions mean	2.984	2.980	3.003	2.974	2.985
RMSE	2.146	2.047	1.968	2.417	2.175
MAE	0.770	1.060	1.013	0.699	0.911
R2 score	0.925	0.808	0.845	0.939	0.899

Table A.16: Evaluation results of 16 clusters using XGBoost on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	8.002	3.524	3.888	7.735	6.063
Predictions mean	7.701	3.401	3.675	7.447	5.894
RMSE	3.662	2.691	2.817	3.524	3.186
MAE	1.784	1.246	1.208	1.698	1.527
R2 score	0.857	0.725	0.741	0.865	0.804

Table A.17: Evaluation results of top 5 clusters of 36 using XGBoost on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Clusters	17	36	36	16	26
Values mean	2.870	1.359	1.359	3.253	1.876
Predictions mean	2.793	1.358	1.347	3.171	1.860
RMSE	2.108	1.343	1.378	2.184	1.620
MAE	0.883	0.740	0.706	0.992	0.757
R2 score	0.889	0.738	0.745	0.888	0.836

Table A.18: Evaluation results of regions with at least 90 items over 36 clusters using XGBoost on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	1.359	1.359	1.359	1.359	1.359
Predictions mean	1.327	1.358	1.347	1.329	1.349
RMSE	1.450	1.343	1.378	1.411	1.378
MAE	0.428	0.740	0.706	0.425	0.556
R2 score	0.899	0.738	0.745	0.902	0.845

Table A.19: Evaluation results of 36 clusters using XGBoost on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	6.942	2.732	4.447	6.708	4.861
Predictions mean	6.633	2.633	4.203	6.441	4.668
RMSE	3.327	2.600	2.906	3.249	2.761
MAE	1.632	1.109	1.355	1.473	1.321
R2 score	0.849	0.690	0.763	0.828	0.809

Table A.20: Evaluation results of top 5 clusters of 64 using XGBoost on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Clusters	25	61	56	28	39
Values mean	1.948	0.801	0.887	1.741	1.246
Predictions mean	1.904	0.830	0.911	1.717	1.257
RMSE	1.678	1.070	1.124	1.556	1.267
MAE	0.764	0.599	0.551	0.694	0.668
R2 score	0.874	0.664	0.799	0.862	0.813

Table A.21: Evaluation results of regions with at least 90 items over 64 clusters using XGBoost on Padua

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	0.764	0.764	0.764	0.764	0.764
Predictions mean	0.751	0.789	0.785	0.758	0.775
RMSE	1.051	1.020	1.041	1.031	0.992
MAE	0.309	0.563	0.478	0.313	0.421
R2 score	0.885	0.683	0.785	0.874	0.824

Table A.22: Evaluation results of 64 clusters using XGBoost on Padua

Cluster Number	Squares	K-Means	GMM	Mean Shift	BIRCH
0	70726	1277747	39120	3272403	1237306
1	1866	145411	551551	298973	204491
2	503863	74373	17465	119708	832986
3	2581377	1234998	1497927	88013	101638
4	9670	10926	294601	123700	21509
5	74721	248487	142948	34370	35709
6	666904	741095	1138540	978	455909
7	19785	210559	20168	5894	1055471
8	23870	9186	250462	8743	7763

Table A.23: Number of bikes per region on 9 clusters on Montreal dataset

Cluster Number	Squares	K-Means	GMM	Mean Shift	BIRCH
0	12912	346906	273447	2948722	1087252
1	23817	465916	924909	328936	357405
2	0	14149	585424	150208	124870
3	74558	108997	123952	207287	101638
4	398133	22436	123039	90244	150054
5	312450	58549	24293	31404	637172
6	0	375491	7186	100603	157188
7	119078	946789	866119	31768	1055471
8	2446015	64563	93533	978	7763
9	444409	110715	64705	4905	27739
10	1433	978	62035	5894	79621
11	6090	1006040	21692	1601	38626
12	37020	7763	7763	38785	978
13	59552	147772	410486	8114	7970
14	10406	153282	978	1464	20531
15	6909	122436	363221	1869	98504

Table A.24: Number of bikes per region on 16 clusters on Montreal dataset

Cluster Number	Squares	K-Means	GMM	Mean Shift	BIRCH
0	0	374658	65891	1397967	98504
1	6733	80005	451652	943167	559057
2	1237	21851	63067	546539	7970
3	0	28551	36945	242364	7763
4	0	67033	55791	49552	16554
5	1869	258207	398174	155570	8565
6	15834	978	395369	114453	192357
7	13962	101958	978	75936	219446
8	48794	30847	4905	64210	479842
9	1866	73468	140353	39318	67741
10	0	295094	60271	116003	78098
11	10447	7763	67682	6512	211015
12	89547	136273	7763	978	280491
13	347953	8438	1866	11371	91297
14	284370	15279	16084	10451	25864
15	0	2197	113471	6872	65891
16	0	22597	132328	21373	34257
17	26069	9545	12093	3959	4669
18	377800	106525	69897	39861	271696
19	1618188	12093	6699	2845	85084
20	330866	2464	264560	40250	295138
21	9670	302022	90810	4905	9545
22	0	75607	33099	11880	49268
23	0	77889	40522	9336	308749
24	74721	6672	128960	5417	11880
25	505012	407793	80879	5894	978
26	153873	415379	8438	3536	19174
27	16235	173325	2281	2363	53668
28	0	187721	19958	8024	33957
29	0	9336	35005	817	27400
30	0	36162	258221	1464	71258
31	6413	59276	274363	5593	10923
32	1606	63356	11880	710	9608
33	2845	2483	266900	742	62620
34	705	328251	264277	1127	9336
35	6167	146697	71350	1423	173119

Table A.25: Number of bikes per region on 36 clusters on Montreal dataset

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	400.132	338.448	355.978	395.088	331.545
Predictions mean	400.464	338.730	356.277	395.414	331.828
RMSE	236.456	177.874	184.821	283.141	168.872
MAE	118.194	102.494	107.526	116.456	101.692
R2 score	0.789	0.707	0.693	0.824	0.657

Table A.26: Evaluation results of top 3 clusters of 9 using ARIMA on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Clusters	9	9	9	9	9
Values mean	140.579	140.579	140.579	140.579	140.579
Predictions mean	140.830	140.730	140.729	140.822	140.732
RMSE	136.724	103.182	107.448	163.703	101.581
MAE	42.890	44.396	44.563	42.567	44.277
R2 score	0.847	0.805	0.804	0.855	0.793

Table A.27: Evaluation results of regions with at least 120 items over 9 clusters using ARIMA on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	140.579	140.579	140.579	140.579	140.579
Predictions mean	140.830	140.730	140.729	140.822	140.732
RMSE	136.724	103.182	107.448	163.703	101.581
MAE	42.890	44.396	44.563	42.567	44.277
R2 score	0.847	0.805	0.804	0.855	0.793

Table A.28: Evaluation results of 9 clusters using ARIMA on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	351.930	257.139	259.027	372.128	295.793
Predictions mean	352.222	257.352	259.254	372.435	296.046
RMSE	221.240	134.758	138.049	254.918	153.337
MAE	105.354	79.972	80.765	110.193	91.203
R2 score	0.794	0.671	0.664	0.823	0.667

Table A.29: Evaluation results of top 3 clusters of 16 using ARIMA on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Clusters	14	16	16	16	16
Values mean	90.372	79.075	79.075	79.075	79.075
Predictions mean	90.505	79.150	79.303	79.291	79.194
RMSE	103.637	63.787	63.644	110.782	69.540
MAE	28.526	26.443	26.333	24.658	26.108
R2 score	0.848	0.797	0.800	0.862	0.813

Table A.30: Evaluation results of regions with at least 120 items over 16 clusters using ARIMA on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	79.075	79.075	79.075	79.075	79.075
Predictions mean	79.192	79.150	79.303	79.291	79.194
RMSE	96.944	63.787	63.644	110.782	69.540
MAE	24.960	26.443	26.333	24.658	26.108
R2 score	0.850	0.797	0.800	0.862	0.813

Table A.31: Evaluation results of 16 clusters using ARIMA on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	264.644	129.186	147.409	310.434	145.848
Predictions mean	264.901	129.295	147.533	310.692	145.980
RMSE	156.407	63.463	82.183	163.572	81.943
MAE	81.972	40.524	48.683	94.828	48.203
R2 score	0.767	0.639	0.621	0.701	0.656

Table A.32: Evaluation results of top 3 clusters of 36 using ARIMA on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Clusters	25	36	36	36	36
Values mean	50.608	35.145	35.145	35.145	35.145
Predictions mean	50.704	35.327	35.281	35.257	35.266
RMSE	57.959	30.302	29.550	48.468	31.022
MAE	16.899	13.124	13.051	12.050	13.098
R2 score	0.840	0.767	0.776	0.842	0.779

Table A.33: Evaluation results of regions with at least 120 items over 36 clusters using ARIMA on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	35.145	35.145	35.145	35.145	35.145
Predictions mean	35.211	35.327	35.281	35.257	35.266
RMSE	48.299	30.302	29.550	48.468	31.022
MAE	11.736	13.124	13.051	12.050	13.098
R2 score	0.846	0.767	0.776	0.842	0.779

Table A.34: Evaluation results of 36 clusters using ARIMA on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	400.132	338.448	355.978	395.088	331.545
Predictions mean	382.486	332.112	328.300	373.956	317.674
RMSE	156.105	115.838	122.276	193.390	115.403
MAE	68.081	58.187	60.233	71.497	60.509
R2 score	0.908	0.861	0.867	0.918	0.840

Table A.35: Evaluation results of top 3 clusters of 9 using XGBoost on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Clusters	9	9	9	9	9
Values mean	140.579	140.579	140.579	140.579	140.579
Predictions mean	134.447	133.436	134.433	133.244	134.613
RMSE	90.223	67.011	68.172	111.743	69.093
MAE	24.225	23.141	24.256	25.554	25.936
R2 score	0.933	0.918	0.920	0.932	0.904

Table A.36: Evaluation results of regions with at least 120 items over 9 clusters using XGBoost on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	140.579	140.579	140.579	140.579	140.579
Predictions mean	134.447	133.436	134.433	133.244	134.613
RMSE	90.223	67.011	68.172	111.743	69.093
MAE	24.225	23.141	24.256	25.554	25.936
R2 score	0.933	0.918	0.920	0.932	0.904

Table A.37: Evaluation results of 9 clusters using XGBoost on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	351.930	257.139	259.027	372.128	295.793
Predictions mean	332.141	235.817	250.438	360.346	279.911
RMSE	148.700	88.871	93.568	162.976	101.717
MAE	64.065	42.773	46.277	64.379	52.585
R2 score	0.907	0.851	0.845	0.928	0.854

Table A.38: Evaluation results of top 3 clusters of 16 using XGBoost on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Clusters	14	16	16	16	16
Values mean	90.372	79.075	79.075	79.075	79.075
Predictions mean	85.587	75.408	75.608	76.527	75.063
RMSE	69.416	41.889	41.923	70.791	45.880
MAE	16.573	13.834	14.303	13.972	14.315
R2 score	0.932	0.918	0.912	0.944	0.919

Table A.39: Evaluation results of regions with at least 120 items over 16 clusters using XGBoost on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	79.054	79.075	79.075	79.075	79.075
Predictions mean	74.889	75.408	75.608	76.527	75.063
RMSE	64.933	41.889	41.923	70.791	45.880
MAE	14.502	13.834	14.303	13.972	14.315
R2 score	0.933	0.918	0.912	0.944	0.919

Table A.40: Evaluation results of 16 clusters using XGBoost on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	264.644	129.186	147.409	310.434	145.848
Predictions mean	253.929	122.051	140.059	295.719	136.704
RMSE	109.595	54.430	49.557	108.978	55.790
MAE	48.407	25.772	23.236	53.782	26.031
R2 score	0.886	0.798	0.784	0.867	0.841

Table A.41: Evaluation results of top 3 clusters of 36 using XGBoost on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Clusters	25	36	36	36	36
Values mean	50.608	35.145	35.145	35.145	35.145
Predictions mean	48.585	33.303	33.498	33.555	33.507
RMSE	40.166	19.616	19.922	32.095	20.340
MAE	9.463	6.590	6.582	6.538	6.558
R2 score	0.923	0.900	0.896	0.931	0.905

Table A.42: Evaluation results of regions with at least 120 items over 36 clusters using XGBoost on Montreal

Metric	Squares	K-Means	GMM	Mean Shift	BIRCH
Values mean	35.145	35.145	35.145	35.145	35.145
Predictions mean	33.741	33.303	33.498	33.555	33.507
RMSE	33.472	19.616	19.922	32.095	20.340
MAE	6.573	6.590	6.582	6.538	6.558
R2 score	0.926	0.900	0.896	0.931	0.905

Table A.43: Evaluation results of 36 clusters using XGBoost on Montreal

References

[1] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785. URL: https://doi.org/10.1145/2939672.2939785.

[2] D. Comaniciu and P. Meer. "Mean shift: a robust approach toward feature space analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.5 (2002), pp. 603–619. DOI: 10.1109/34.1000236.

[3] David Cournapeau. *Scikit-learn Machine Learning in Python*. URL: https://scikit-learn.org/stable/. (accessed: 04.11.2022).

[4] Jean-Michel D. *Public bike sharing in north america*. URL: https://www.kaggle.com/datasets/jeanmidev/public-bike-sharing-in-north-america. (accessed: 15.12.2022).

[5] Dingsheng Deng. "DBSCAN Clustering Algorithm Based on Density". In: *2020 7th International Forum on Electrical Engineering and Automation (IFEEA)*. 2020, pp. 949–953. DOI: 10.1109/IFEEA51475.2020.00199.

[6] Jian Dong et al. "A Spatio-Temporal Flow Model of Urban Dockless Shared Bikes Based on Points of Interest Clustering". In: *ISPRS International Journal of Geo-Information* 8.8 (2019). ISSN: 2220-9964. DOI: 10.3390/ijgi8080345. URL: https://www.mdpi.com/2220-9964/8/8/345.

[7] S.L. Ho and M. Xie. "The use of ARIMA models for reliability forecasting and analysis". In: *Computers and Industrial Engineering* 35.1 (1998), pp. 213–216. ISSN: 0360-8352. DOI: https://doi.org/10.1016/S0360-8352(98)00066-7. URL: https://www.sciencedirect.com/science/article/pii/S0360835298000667.

[8] Mingzhuang Hua et al. "Large-scale dockless bike sharing repositioning considering future usage and workload balance". In: *Physica A: Statistical Mechanics and its Applications* 605 (2022), p. 127991. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2022.127991. URL: https://www.sciencedirect.com/science/article/pii/S0378437122006227.

[9] John D. Huntert. *Matplotlib: Visualization with Python*. URL: https://matplotlib.org/. (accessed: 10.11.2022).

[10] Arun Jagota. *Density-based and Graph-based Clustering*. URL: https://towardsdatascience.com/density-based-and-graph-based-clustering-a1f0d45ff5fb. (accessed: 29.10.2022).

[11] Xin Jin and Jiawei Han. "K-Means Clustering". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 563–564. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_425. URL: https://doi.org/10.1007/978-0-387-30164-8_425.

[12] Saurav Kaushik. *Clustering | Introduction, Different Methods, and Applications*. URL: https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/. (accessed: 25.10.2022).

[13] Xinyu Li et al. "Short-Term Forecast of Bicycle Usage in Bike Sharing Systems: A Spatial-Temporal Memory Network". In: *IEEE Transactions on Intelligent Transportation Systems* 23.8 (2022), pp. 10923–10934. DOI: 10.1109/TITS.2021.3097240.

[14] Cory Maklin. *Gaussian Mixture Models Clustering Algorithm Explained*. URL: https://towardsdatascience.com/gaussian-mixture-models-d13a5e915c8e. (accessed: 29.10.2022).

[15] Wes McKinney. *Pandas - Python Data Analysis Library*. URL: https://pandas.pydata.org//. (accessed: 04.11.2022).

[16] Travis Oliphant. *Numpy*. URL: https://numpy.org/. (accessed: 02.11.2022).

[17] Sakshi Patel, Shivani Sihmar, and Aman Jatain. "A study of hierarchical clustering algorithms". In: *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*. 2015, pp. 537–541.

[18] L. Rabiner and B. Juang. "An introduction to hidden Markov models". In: *IEEE ASSP Magazine* 3.1 (1986), pp. 4–16. DOI: 10.1109/MASSP.1986.1165342.

[19] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. "BIRCH: an efficient data clustering method for very large databases". In: *ACM SIGMOD Conference*. 1996.

Acknowledgments

After all this pages, I want to thank the people that have supported me through this course of study. Starting with my Family, my mother Lucia, my father Stefano and my sister Valentina who have always been close to me and helped me make the right choices. My girlfriend Marianna, who has long been my life partner. It is thanks to her that I was able to finish this journey, because she always manages to get me back up in difficult times and is the only person who can make me bring out the best in myself. My grandmothers, Emiliana and Silvana. It is mainly because of them that I have become what I am. My grandfather Luigi. It is because of him that my passion for engineering was born, seeing its skills and abilities in all manual activities. My great-grandmother Rosa who has been a constant in my growth, teaching me a lot of games that have led me to be a very curious person about everything new.

Furthermore, Nicola and Ilaria. I have to thank them because they have always encouraged me in keeping on my journey and continue to give me valuable advice. My whole group of friends, Luca, Francesco, Gianluca, Anna V., Bernardo, Pietro, Martina, Ludovico, Andrea and Anna Z. We have known each other since We were just over a metre tall. We have shared a lot of experiences and trips and I hope we will soon share even more beautiful ones.

Last but not least, Professor Matteo and PhD student Margherita, who supported me in this last step of my study career, giving me a lot of help during my thesis project in a professional but at the same time very friendly manner.

To conclude, I want to thank all the people who have been close to me and made me smile even once.