Transport Logistics and Operation Research - Project 1

Demand Forecast and Forecast based clustering for waste collection sites in Serre Ponçon

Final report

Group 4:

Sara BAZYAN, Harihareshwar KUMARAVEL, Albin TAMEWE, Aboubaker AL KADIRI, Felix BRECHT, Mohamad FLAITY

Reviewer: Van Dat CUNG

Second reviewer: Siao Leu PHOURATSAMAY

Table of Contents

List of figures

List of tables

Abstract

We perform forecasting of the fill rate of containers for four waste categories (OM, BX, Verre and Carton) at 328 pickup points. This forecasting is done in order to optimize the collection of waste by clustering the pickup points based on their distance and fill rate. Once the points have been clustered, an optimal route is determined for each cluster for a specific day. The primary objective of this project is to improve the efficiency of waste collection by identifying the most effective routes for our collection vehicles to follow.

1. Introduction

In contrast to the planet's declining ability to absorb trash, waste creation is increasing globally at an inexorable rate, making waste treatment a difficult problem that modern societies must deal with. The Earth is exhausted, and natural resources are depleted. In addition, the linear structure of our economy—extract, manufacture, use, and disposal—has reached its limitations. Due to the rising amount of garbage being produced, the unsustainable development of the majority of countries has led to an issue that needs to be resolved. There are hardly any technology tools available to optimize the administration of the garbage collecting process, despite social and governmental commitment. Municipal garbage collection requires the most resources because it is primarily produced by domestic, industrial, and commercial activities.

Serre-Poncon lake is a large artificial lake in France, which has a very high seasonality due to its touristic location. In this area, there are 328 pick-up sites of waste with more than 1000 containers in total. There is one main road to reach the disposal site center (Embrun), which is the only one for all the collection sites, along with several other mountain roads.

From the historical data of that area, we know that there are 4 types of waste which are divided between domestic waste, multi-material waste, glass waste & cardboard waste, with the majority concentrated in the range of domestic and multi-material.

This waste collection is managed by 6 trucks, and the rate of visit to each site can vary from once per day in peak season, to once every 3 months in the off-season.

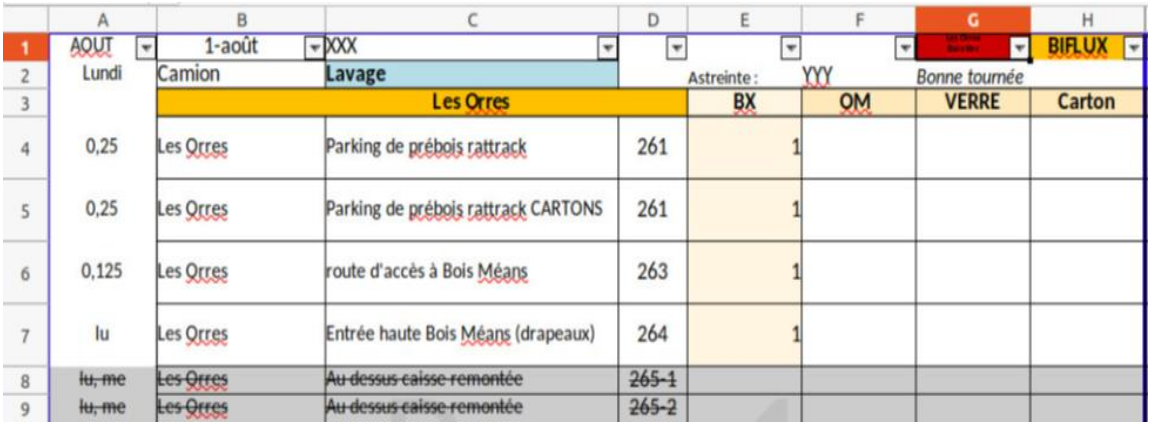
In the past, solid trash collection was done without first studying the demand or the routes used by the vehicles, and the drivers chose the routes. Even though the answers were far from ideal. The majority of the time, static plans with a predefined number of weekly visits are used to guide the collection of solid trash. Because there are so many constraints to take into account while determining the best solution, this method has significant drawbacks. Furthermore, the usage of an automatic tool is required to address these types of problems due to the stochastic nature of trash generation, traffic circumstances, and many other limitations that are insurmountable for a person to handle.

In this paper, we propose a tool to predict the behavior of filling rates of the sites based on the historical data in our hands. This tool is “Time-series forecasting” which gives the ability to model the underlying patterns and trends of those filling rates with great accuracy. In other words, it would allow us to address the problem of what containers should be visited first with their dates.

Different time-series forecasting was used: Exponential smoothing, Autoregressive method, ARIMA, ARMA & SARIMA. The choice of the technique was made based on the nature and characteristics of the given data of the filling rates and the resources available.

2. Problem description Overview

The current solutions proposed by the experts working on waste collection in that area are mainly based on human expertise, which can increase the rate of error and costs. In other words, the frequencies, dates & clusters of the sites visits are based on experience without a certain algorithm to anticipate the results from the historical data of the filling rates. Below is a small example of the methodology used:



	A	B	C	D	E	F	G	H
1	AOUT	1-août	XXX					BIFLUX
2	Lundi	Camion	Lavage		Astreinte :	YYY	Bonne tournée	
3		Les Orres			BX	OM	VERRE	Carton
4	0,25	Les Orres	Parking de prébois rattrack	261	1			
5	0,25	Les Orres	Parking de prébois rattrack CARTONS	261	1			
6	0,125	Les Orres	route d'accès à Bois Méans	263	1			
7	lu	Les Orres	Entrée haute Bois Méans (drapeaux)	264	1			
8	~~lu, me~~	~~Les Orres~~	~~Au dessus caisse remontée~~	~~265-1~~				
9	~~lu, me~~	~~Les Orres~~	~~Au dessus caisse remontée~~	~~265-2~~				

Figure 1: Input data indicating waste collection

The problem with this methodology, is that it is slow and not very effective:

- Mostly made by hand
- Adjusted every day, redone from the start by every period
- Clustering of the Tours is only based on number of sites
- No advantage taken of the historical data of the container's filling rates

We can summarize the problems that we have in this article, for the waste collection service management in the Serre-Poncon area, with two major questions:

1) What containers should be collected and at what date?

2) What are the sites clustered for a given period depending on that demand to increase effectiveness?

To address these two problems, we built an algorithm using python to generate a demand forecasting of the filling rates of the sites, in order to anticipate the dates in which each site should be visited for a given period of time. In addition, based on this forecast, and using k-means methodology, the algorithm would cluster the sites within a feasible solution.

3. Literature Review

Time series forecasting is a statistical method used to predict future values of a time series based on its past values, and it plays an important role in demand forecasting. A number of different time series forecasting models have been developed to address the complexities of forecasting demand, including ARIMA, SARIMA, exponential smoothing, autoregressive, and ARMA models.

ARIMA (Auto-Regressive Integrated Moving Average) models are one of the most widely used methods for time series forecasting. According to *Hyndman and Khandakar (2008),* ARIMA models are suitable for non-stationary time series and they have shown to be effective in many practical applications. On the other hand, SARIMA (Seasonal Auto-Regressive Integrated Moving Average) models are an extension of ARIMA models that take into account the presence of seasonality in the time series data. According to *Chen and Liu (2016),* SARIMA models are particularly useful for forecasting demand for products or services that have a clear seasonal pattern.

Exponential smoothing models, as stated by *Hyndman and Athanasopoulos (2018),* are another popular method for time series forecasting. These models are based on the idea of using a weighted average of past observations to make predictions about future values. They are particularly useful for short-term forecasting, as they are able to quickly respond to changes in the underlying data.

Autoregressive and ARMA (Autoregressive Moving Average) models are also commonly used in time series forecasting. According to *Box and Jenkins (1970),* autoregressive models use past values of the time series to make predictions about future values, while ARMA models take into account both past values and past errors in the predictions. These models are useful for medium-term forecasting, and are typically used in conjunction with other models to provide more accurate predictions.

In general, it can be seen that time series forecasting plays an important role in demand forecasting, and a number of different models have been developed to address the complexities of forecasting demand. These include ARIMA, SARIMA, exponential smoothing, autoregressive, and ARMA models. Each model has its own advantages and limitations, and the choice of model will depend on the specific characteristics of the time series data and the forecasting horizon.

It is important to note that a good practice for time series forecasting is to use multiple models and compare their performance, this way one can select the best model for a specific problem. Also, regular evaluation of the model's performance and adjusting the model accordingly can help improve the accuracy of the predictions. It is also worth noting that there are many other types of time series forecasting models that have been proposed in the literature, such as state space models, neural networks, and machine learning

methods. These models can also be used to improve the accuracy of demand forecasting, and are being actively researched by many scholars in the field.

Model Evaluation is an important step in forecasting. It is important to evaluate forecasting models because accurate forecasts are essential for making informed business decisions. By evaluating forecasting models, you can ensure that your forecasts are as accurate as possible, which can help you make better decisions and improve your business operations. The RMSE error, a frequently used indicator of the discrepancy between anticipated values and actual values, is utilized in this research to compare several models. It is easier to interpret and is easy to calculate for various different types of data, including continuous, discrete and categorical data. It is also in the same scale as the data and it makes it easier to interpret and compare different models. *Saurabh Suradhaniwar et al. (2021)* used RMSE value to evaluate One-step and Multi-step ahead forecasting strategies.

There are several different methods for clustering products or demand patterns, including:

K-means clustering: This is a popular method for clustering data into a specific number of groups (k). It works by iteratively assigning data points to the closest group and updating the group centroids based on the data points in each group.

Hierarchical clustering: This method involves creating a hierarchy of clusters, with each cluster being defined as a subset of the larger cluster above it. There are two main types of hierarchical clustering: Agglomerative clustering, which starts with individual data points and combines them into larger clusters, and divisive clustering, which starts with all data points in a single cluster and divides them into smaller clusters.

Density-based clustering: This method involves identifying clusters based on the density of data points within a given area. Clusters are formed around areas of high density, with points that are farther away from these dense areas being considered as outliers or noise.

4. Related Terminologies

- **Time Series Forecasting models:**

Time series forecasting is the process of using historical data to make predictions about future events. It is a common task in many fields, including finance, economics, and engineering, and it can be used to make informed decisions about resource allocation, risk management, and other important issues. One important aspect of time series forecasting is the ability to accurately model the underlying patterns and trends in the data. This can be challenging because real-world data is often noisy and may contain multiple levels of trend and seasonality. There are many different techniques for time series forecasting, ranging from simple methods like moving averages and exponential smoothing to more complex techniques like autoregressive integrated moving average (ARIMA) models and neural networks. The choice of technique depends on the characteristics of the data, the goals of the forecast, and the resources available. In conclusion, time series forecasting is a valuable tool for predicting future events based on historical data. It requires careful analysis of the data, selection of appropriate forecasting techniques, and evaluation of the forecasts to ensure their accuracy and usefulness.

To do the forecasting of fill rate, the following time series forecasting algorithms were used.

- Exponential Smoothing
- Autoregressive Model
- Auto Regressive Moving Average (ARMA) Model
- Autoregressive integrated moving average (ARIMA) model
- Seasonal autoregressive integrated moving average (SARIMA) Model.

The explanation of each model is given in the following section.

- **Exponential smoothing**

Exponential smoothing is a popular time series forecasting technique that is used to smooth out short-term fluctuations and make predictions about the future. It works by assigning exponentially decreasing weights to past observations, with the most recent observations given the highest weight. This allows the model to capture the most recent trends in the data while still incorporating information from earlier observations. One of the key benefits of exponential smoothing is that it is relatively simple to implement and interpret. It is also relatively robust to variations in the data, making it a good choice for forecasting data that exhibits a wide range of patterns and trends. There are several different variations of exponential smoothing, including single exponential smoothing, double exponential smoothing, and triple exponential smoothing. Single exponential smoothing is the most basic form, and it only considers the current and previous time steps when making predictions. Double and triple exponential smoothing add additional terms

to the prediction formula to account for trends and seasonality in the data. Overall, exponential smoothing is a powerful and widely used technique for forecasting time series data. It can be used for a wide range of applications, including sales forecasting, financial forecasting, and demand forecasting, among others.

Here, only single exponential smoothing is implemented because it is the most basic form of exponential smoothing and it only considers the current and previous time steps when making predictions. It is based on the following formula:

$$Forecast(t) = \alpha * Y(t) + (1 - \alpha) * Forecast(t-1)$$

Where:

- **Forecast(t)** is the forecast for time step t
- **Y(t)** is the actual value at time step t
- **Forecast(t-1)** is the forecast for the previous time step
- **α** is the smoothing factor, which determines the weight assigned to the most recent observations. A value of 0 means that the forecast will not be updated based on new data, while a value of 1 means that the forecast will be fully based on the most recent observation.

- **Auto Regressive Model (AR) Model**

An autoregressive (AR) model is a type of time series model that uses past values of the time series to make predictions about the future. It is based on the assumption that the current value of the time series is related to its past values.

The basic form of an autoregressive model of order p is given by the following formula:

$$Y(t) = c + \Phi 1 Y(t-1) + \Phi 2 Y(t-2) + \ldots + \Phi p * Y(t-p) + \varepsilon(t)$$

Where:

- **Y(t)** is the value of the time series at time step t
- **c** is a constant term
- **Φ1, Φ2, …, Φp** are the autoregressive coefficients, which determine the influence of past values on the current value
- **Y(t-1), Y(t-2), …, Y(t-p)** are the past values of the time series
- **ε(t)** is the error term, which represents the unpredictable part of the time series that is not explained by the model.

Autoregressive models are widely used for time series forecasting because they are relatively simple to implement and interpret. They are also relatively robust to variations in the data, making them a good choice for forecasting data that exhibits a wide range of patterns and trends. However, they may not be

suitable for data that exhibits more complex patterns, such as long-term trends or seasonality, and may require the use of more advanced models.

- **Auto Regressive Moving Average (ARMA) Model**

An autoregressive moving average (ARMA) model is a type of time series model that combines the autoregressive (AR) and moving average (MA) models. It is based on the assumption that the current value of the time series is related to both its past values and the errors from past predictions.

The basic form of an autoregressive moving average model of orders p and q is given by the following formula:

$$Y(t) = c + \Phi 1Y(t-1) + \Phi 2Y(t-2) + \ldots + \Phi pY(t-p) - \Theta 1\varepsilon(t-1) - \Theta 2\varepsilon(t-2) - \ldots - \Theta q\varepsilon(t-q) + \varepsilon(t)$$

Where:

- **Y(t)** is the value of the time series at time step t
- **c** is a constant term
- **Φ1, Φ2, …, Φp** are the autoregressive coefficients, which determine the influence of past values on the current value
- **Y(t-1), Y(t-2), …, Y(t-p)** are the past values of the time series
- **Θ1, Θ2, …, Θq** are the moving average coefficients, which determine the influence of the errors on the current value
- **ε(t-1), ε(t-2), …, ε(t-q)** are the past errors
- **ε(t)** is the error term, which represents the unpredictable part of the time series that is not explained by the model.

Autoregressive moving average (ARMA) models are widely used for time series forecasting because they are relatively simple to implement and interpret, and can capture both short-term dependencies and long-term trends in the data. They are particularly well-suited for data that exhibits a combination of autocorrelation and moving average patterns, and are relatively robust to variations in the data. Overall, ARMA models can be a useful tool for forecasting time series data and are an important part of the time series analysis toolkit.

- **Autoregressive integrated moving average (ARIMA) model**

Autoregressive integrated moving average (ARIMA) is a type of time series model that combines the autoregressive (AR) and moving average (MA) models with the concept of integration. It is used to model time series data that exhibits a trend or seasonality, and is particularly well-suited for data that is not stationary (i.e., the mean, variance, or autocorrelation structure of the data changes over time).

The basic form of an autoregressive integrated moving average model of orders (p,d,q) is given by the following formula:

$$Y(t) = c + \Phi 1Y(t-1) + \Phi 2Y(t-2) + \ldots + \Phi pY(t-p) - \Theta 1\varepsilon(t-1) - \Theta 2\varepsilon(t-2) - \ldots - \Theta q\varepsilon(t-q) + \varepsilon(t)$$

Where:

- **Y(t)** is the value of the time series at time step t
- **c** is a constant term
- **Φ1, Φ2, …, Φp** are the autoregressive coefficients, which determine the influence of past values on the current value
- **Y(t-1), Y(t-2), …, Y(t-p)** are the past values of the time series
- **Θ1, Θ2, …, Θq** are the moving average coefficients, which determine the influence of the errors on the current value
- **ε(t-1), ε(t-2), …, ε(t-q)** are the past errors
- **ε(t)** is the error term, which represents the unpredictable part of the time series that is not explained by the model.

The orders of the model, (p,d,q), determine the number of past values, the degree of differencing, and the number of past errors that are considered when making predictions. The parameter d represents the degree of differencing, which is the number of times that the time series is different to make it stationary. A higher degree of differencing may be necessary for data that exhibits a strong trend or seasonality. To fit an autoregressive integrated moving average model to a time series, the model parameters (i.e., the constant term c, the autoregressive coefficients Φ1, Φ2, …, Φp, and the moving average coefficients Θ1, Θ2, …, Θq) must be estimated from the data. This can be done using a variety of techniques, such as maximum likelihood estimation or least squares regression. Once the model is fit, it can be used to make predictions about future time steps by plugging in the predicted values for Y(t), Y(t-1), …, Y(t-p), ε(t-1), ε(t-2), …, ε(t-q). Overall, autoregressive integrated moving average models are a powerful tool for forecasting time series data and are widely used in many different fields.

- **Seasonal autoregressive integrated moving average (SARIMA) Model**

Seasonal autoregressive integrated moving average (SARIMA) is a type of time series model that combines the autoregressive integrated moving average (ARIMA) model with the concept of seasonality. It is used to model time series data that exhibits both trend and seasonality, and is particularly well-suited for data that exhibits periodic patterns on a seasonal time scale (e.g., daily, weekly, monthly).

The basic form of a seasonal autoregressive integrated moving average model of orders (p,d,q)(P,D,Q)m is given by the following formula:

$$Y(t) = c + \Phi 1Y(t-1) + \Phi 2Y(t-2) + \ldots + \Phi pY(t-p) - \Theta 1\varepsilon(t-1) - \Theta 2\varepsilon(t-2) - \ldots - \Theta q\varepsilon(t-q) + \Psi 1Y(t-m) + \Psi 2Y(t-2m) + \ldots + \Psi PY(t-Pm) - \Omega 1\varepsilon(t-m) - \Omega 2\varepsilon(t-2m) - \ldots - \Omega Q\varepsilon(t-Qm) + \varepsilon(t)$$

Where:

- **Y(t)** is the value of the time series at time step t
- **c** is a constant term
- **Φ1, Φ2, …, Φp** are the autoregressive coefficients, which determine the influence of past values on the current value
- **Y(t-1), Y(t-2), …, Y(t-p)** are the past values of the time series
- **Θ1, Θ2, …, Θq** are the moving average coefficients, which determine the influence of the errors on the current value
- **ε(t-1), ε(t-2), …, ε(t-q)** are the past errors
- **m** is the seasonal period (e.g., m=12 for monthly data)
- **Ψ1, Ψ2, …, ΨP** are the seasonal autoregressive coefficients, which determine the influence of past values at the same seasonality on the current value
- **Y(t-m), Y(t-2m), …, Y(t-Pm)** are the past values of the time series at the same seasonality
- **Ω1, Ω2, …, ΩQ** are the seasonal moving average coefficients, which determine the influence of the errors at the same seasonality.

As with the non-seasonal ARIMA model, the orders of the model, (p,d,q)(P,D,Q)m, determine the number of past values, the degree of differencing, and the number of past errors that are considered when making predictions. The parameter d represents the degree of differencing, which is the number of times that the time series is differenced to make it stationary. A higher degree of differencing may be necessary for data that exhibits a strong trend or seasonality. The parameter D represents the degree of seasonal differencing, which is the number of times that the time series is differenced with respect to the seasonal period (m). To fit a seasonal autoregressive integrated moving average model to a time series, the model parameters (i.e., the constant term c, the autoregressive coefficients Φ1, Φ2, …, Φp, the moving average coefficients Θ1, Θ2, …, Θq, the seasonal autoregressive coefficients Ψ1, Ψ2, …, ΨP, and the seasonal moving average coefficients Ω1, Ω2, …, ΩQ) must be estimated from the data. This can be done using a variety of techniques, such as maximum likelihood estimation or least squares regression. Once the model is fit, it can be used to make predictions about future time steps by plugging in the predicted values for Y(t), Y(t-1), …, Y(t-p), ε(t-1), ε(t-2), …, ε(t-q), Y(t-m), Y(t-2m), …, Y(t-Pm), ε(t-m), ε(t-2m), …, ε(t-Qm).

Overall, seasonal autoregressive integrated moving average models are a powerful tool for forecasting time series data with seasonality and are widely used in many different fields.

5. Methodology

- **Data Cleaning**

Data cleaning is an essential step in the data pre-processing process. It involves identifying and correcting or removing inconsistencies or errors in the data in order to prepare it for further analysis or visualization. In this project, the data cleaning process began by merging data from various sheets of different waste types with the filling rate into one sheet for each type of waste, as this data was necessary for time series forecasting. The data was grouped based on the type of waste, such as BX, Carton, Verre, and OM. Each sheet had four columns with important information, including the date, the commune, the point number, and the percentage rate. To ensure that the data was consistent and could be easily analyzed, the column names and data types were standardized. The **date** column was changed to a **date data type**, the **commune** column was changed to a **string data type**, the **n_point** column was changed to a **string data type**, and the **taux** column was changed to a **double** data type from percentage. After the data was merged and standardized, it was checked for duplicates and any duplicates were removed. The resulting dataset consisted of four sheets, one for each type of waste. This cleaned and organized dataset was then ready for further analysis.

Column Names – Before Cleaning	Column Name – After cleaning
Date	date
Commune	commune
Points OM* or N° point	n_point
Taux	taux

** - Changes based on type of waste*

Table 1: Colum names before and after cleaning

Data types of each column after cleaning

Date	datetime64[ns]
commune	object
n_point	object
taux	float64

Table 2 : Column datatype

The cleaned data is stored in the different worksheets of the same spreadsheet file for the ease of use in the future

The missing values found in 'taux' columns were filled with mean fill rate during the final code execution. This is a useful strategy when working with datasets that have a large number of missing values, as it allows to retain as much of the available data as possible while still addressing the issue of missing values. The potential outliers with fill rate more than 100% were filtered out to avoid skewing the results.

- **Methodology for Forecasting:**

To predict the fill rate, a python script was used. The step-by-step methodology is given below.

1. Select the data frame from cleaned data based on user inputs – n_point and type of waste
2. Split the selected data frame into a training set and a test set. (80% Train data and 20% Test data)
3. Fit the following 5 time series forecasting models to the training set:
    a. Exponential Smoothing
    b. Autoregressive model
    c. Autoregressive Moving Average model
    d. ARIMA model
    e. SARIMA model
4. For each model, Use the fitted model to make predictions on the testing set and calculate the RMSE by comparing the predicted values to the actual values in the test set.
5. Select the model with the lowest RMSE.
6. Use the selected model to make future predictions on new data.

- **Forecasting Evaluation:**

There are metrics that can be used to evaluate the performance of time series forecasting models. Some common ones include:

Mean absolute error (MAE): This measures the average absolute difference between the predicted values and the actual values. It is calculated as the sum of the absolute differences divided by the number of samples. MAE is less sensitive to large errors than RMSE, but it may be less robust when the error distribution is skewed.

Mean absolute percentage error (MAPE): This measures the average absolute percentage difference between the predicted values and the actual values. It is calculated as the sum of the absolute percentage differences divided by the number of samples. MAPE is useful when the values

in the time series vary significantly, as it adjusts for the scale of the data. However, it can be sensitive to outliers and may produce misleading results when the actual values are very small.

Mean squared error (MSE): This measures the average squared difference between the predicted values and the actual values. It is calculated as the sum of the squared differences divided by the number of samples. MSE is sensitive to large errors and is commonly used in regression problems.

Median absolute error (MedAE): This measures the median absolute difference between the predicted values and the actual values. It is calculated as the median of the absolute differences. MedAE is less sensitive to outliers than the mean absolute error, but it may not be as robust when the error distribution is skewed.

In this particular case, Root mean squared value is used.

Root mean squared error (RMSE) is a common metric used to evaluate the performance of time series forecasting models. It is calculated as the square root of the mean squared error (MSE), which is defined as the average of the squared differences between the predicted and actual values.

RMSE = sqrt (MSE) = sqrt (mean ((predicted – actual)^2))

where:

- **predicted** is the vector of predicted values.
- **actual** is the vector of actual values.

There are several reasons why RMSE is a useful metric for evaluating time series forecasting models. First, RMSE is easy to understand and interpret, as it is in the same units as the original data. This makes it easy to compare the performance of different models or different configurations of the same model. Second, RMSE is a continuous, positive-valued metric, which means that it is always greater than or equal to zero. A lower RMSE value indicates a better fit between the predicted values and the actual values, and a higher RMSE value indicates a worse fit.

In conclusion, RMSE is a useful metric for evaluating the performance of time series forecasting models because it is easy to understand and it is in the same units as the original data

- **Methodology for Clustering:**

K-means clustering is a popular and simple method for grouping a set of objects into clusters based on distance measures between the objects. It is an unsupervised learning algorithm, meaning that it is used to find patterns in data without being given any labeled examples. The basic idea behind k-means clustering is to define a distance measure between the objects to be clustered, and then use an algorithm to partition the

objects into k clusters such that the distance between objects within a cluster is minimized, while the distance between objects in different clusters is maximized. The number of clusters (k) is a user-specified parameter that needs to be chosen in advance. The k value can also be calculated using a certain method called the Elbow Method. It is a heuristic used to determine the optimal number of clusters in a K-Means clustering analysis. It is based on the idea that the optimal number of clusters is the one that minimizes the within-cluster sum of squared errors (SSE). The within-cluster SSE is calculated as the sum of the squared distances between each data point and the centroid of its cluster. It is a measure of the compactness and cohesiveness of the clusters, and a lower SSE indicates that the clusters are more tightly grouped and have a smaller dispersion. To apply the elbow method, the K-Means algorithm is run for a range of values of k (the number of clusters), and the SSE is calculated for each value of k. The resulting values of SSE are plotted on a graph, with k on the x-axis and SSE on the y-axis. The optimal number of clusters is then determined by looking for an "elbow" in the graph, where the rate of change in SSE begins to level off. This point is considered to be the optimal value of k, as it corresponds to a balance between minimizing the SSE and maximizing the number of clusters.

One of the advantages of k-means clustering is that it is relatively fast and easy to implement, and it can scale to large datasets. However, it can be sensitive to the initial choice of centroids, and it may not always find the "true" underlying clusters in the data. It is also limited to dividing the data into clusters of roughly equal size and shape, and it is sensitive to outliers. These limitations should be kept in mind when using k-means clustering, and other clustering algorithms may be more suitable for certain situations. In context of the project the idea is to cluster the predictions based on ‘distance of the point from depot’ and ‘predicted fill rate’

- **Methodology for Vehicle Routing Problem**

The Vehicle Routing Problem (VRP) is a well-known optimization problem in which a set of vehicles must visit a given set of locations, while minimizing the total distance traveled. In this case, the VRP is being applied to a list of lists of locations, which is the result of the clustering where each inner list represents a set of locations that must be visited by a single vehicle

To solve VRP using Python, follow these steps:

- Define the problem: Identify the locations that need to be visited and the capacity of each vehicle. Also, specify the distance between each pair of locations.
- Formulate the problem as an optimization problem: Choose a suitable mathematical formulation, such as linear programming or integer programming, to represent the VRP.
- Solve the optimization problem: Use a Python library, such as Pulp, Pyomo, Gurobi, or Google OR Tools to solve the optimization problem.

- Evaluate the solution: Check that the solution is feasible and meets the constraints of the VRP, such as ensuring all locations are visited and that the vehicles are not overloaded.
- Implement the solution: Create a plan for each vehicle's route and provide instructions for the drivers. This may involve creating a map of the routes.

6. Implementation

Various python scripts were written to solve different parts of the problem. An overview how each script is implemented is given in this following section

Forecasting Script

The code is implemented in 4 python scripts. The first is the forecasting script which is used to forecast the fill rate based on the location and date. The program starts by importing various libraries to perform the prediction. Below are the libraries that were used

- **statsmodels**: This library contains functions for implementing exponential smoothing and autoregressive integrated moving average (ARIMA) models. These models were used for forecasting the time series data.
- **sklearn**: This library contains the **mean_squared_error** function, which was used to evaluate the root mean squared error (RMSE) between the predicted values and the actual values.
- **pmdarima**: This library contains the **auto_arima** function, which was used to find the best ARIMA model for a given data frame. The **auto_arima** searches for the optimal combination of autoregressive (AR), moving average (MA), and differencing (I) parameters for an ARIMA model. It works by fitting a range of ARIMA models with different combinations of these parameters to the data, and selecting the model that minimizes the information criterion (such as Akaike's Information Criterion or Bayesian Information Criterion).
- **pandas** and **numpy**: Pandas is a library for data manipulation and analysis in Python, while NumPy is a library for numerical computing in Python. Both libraries are widely used in scientific computing, data analysis, and machine learning.

The code begins by selecting the data frame by using user inputs such as **'d'-** Waste bin location and **'m'** - waste type from the excel file previously cleaned. This dataframe is passed to various evaluation functions of different time series models previously discussed. The script then chooses the model with the least RMSE value and uses it to predict the 10 future readings. This number can be changed inside the script to get more predictions. Keep in mind that this code returns the output.

Multiple Forecasting Script

A second script is used to iterate the forecast for all the locations and store it in a separate excel sheet. A 'for' loop iterates the prediction script for all the values of **'d'** and returns a spreadsheet containing the prediction values for all the locations. A point to be noted here is this process takes a long time to run due to the fact that there are a large number of locations. So, In context of this project the script was run four times

for four different types of wastes and stored in different spreadsheets. These were then merged to make a single file, **'Predictions.xlsx'** for the ease of use in Clustering.

- Clustering Script

To implement the k-means clustering the following libraries were used.

- **'sklearn'** (Scientific Kit for Python's Learning Environment) is a library for machine learning in Python. It provides a range of supervised and unsupervised learning algorithms and is widely used in the field of data science. The 'KMeans' class from 'sklearn' is used in this code to implement the K-Means clustering algorithm.
- **'yellowbrick'** is a library for visualizing machine learning models and algorithms in Python. It provides a range of visualizers for evaluating and comparing models, as well as for understanding their behavior. The **'KElbowVisualizer'** class from **'yellowbrick'** is used in this code to visualize the data and determine the optimal number of clusters for the K-Means algorithm.
- **'pandas'** is a library for data manipulation and analysis in Python. It provides a range of functions and data structures for efficiently storing and manipulating data in a tabular format. The **'read_excel'** function from pandas is used in this code to read data from an Excel file.
- **'numpy'** is a library for numerical computing in Python. It provides a range of functions for efficient manipulation of large arrays of data. The unique function from numpy is used in this code to extract the unique values in an array.

The way the script works is it takes an input **'date'** and type of waste **'m'.** Then, the script uses the 'Selectpoints' function to select the data points for a single date from an Excel file containing the predicted waste generation at each location. It takes the date as input and returns a dataframe containing the date, location, prediction, and distance from the depot for each location. The Script uses the **'KElbowVisualizer'** class to determine the optimum number of clusters for this data frame. This optimum number of clusters is used as an input to 'KMeans' function which then clusters the data and returns a list of lists containing unique locations around Serre-Poncon. This is the input of the Vehicle routing problem.

- Vehicle Routing Problem script

The VRP script takes the list of lists from the previous clustering algorithm as the user input 'loc'. The VRP solution is implemented using the following libraries

- **'routing_enums_pb2'** is a module from the Google OR-Tools library that provides a set of enumeration classes for specifying various options and parameters for routing models. It is used in this code to define the search strategy for the VRP.

- **'pywrapcp'** is a module from the Google OR-Tools library that provides a Python wrapper for the Constraint Programming (CP) solver. It is used in this code to create a routing model and solve the VRP.
- **'numpy'** and **'pandas'** are used for basic data selection and manipulation

The overview of how the script works is as below:

1. The code begins by importing the necessary libraries and modules, including routing_enums_pb2' and 'pywrapcp' from OR-Tools, and pandas for reading data from an Excel file.
2. The VRP is defined by a list of locations, loc, which is a list of lists containing the locations that need to be visited. The function 'distMat' is then defined to extract a distance matrix for the specified locations from an Excel file. The distance matrix is stored in a Pandas dataframe, which is then converted to a NumPy array and returned by the function.
3. The 'create_data_model' function is defined to store the data for the VRP. It takes the distance matrix as an input and defines the number of vehicles and the depot location (which is always the first location in the distance matrix).
4. The 'print_solution' function is defined to print the solution to the console. It takes the manager, routing model, and solution as inputs and prints the route distance and the list of locations visited in the order they were visited. The list of locations is stored in a global list a, which is returned by the function.
5. The 'vrp' function is defined to solve the VRP for each list of locations in loc. It first creates a list pl_list of distance matrices for each list of locations in 'loc'. Then, it iterates over each list of locations and calls the create_data_model and print_solution functions to define the data model and solve the VRP.
6. The 'distance_callback' function is defined as a nested function within the 'vrp' function. It takes the indices of the starting and ending locations as inputs and returns the distance between them using the distance matrix.
7. The 'transit_callback_index' variable is defined as the result of registering the 'distance_callback' function with the routing model. This allows the routing model to use the 'distance_callback' function to compute the cost of each arc.
8. The default search parameters for the VRP are defined and the first solution strategy is set to the "PATH_CHEAPEST_ARC" strategy, which finds the cheapest route among all feasible routes.
9. The VRP is solved using the 'SolveWithParameters' method of the routing model, passing in the search parameters as an argument.

10. If a solution is found, it is printed to the console using the print_solution function. The original list of locations is then reordered based on the solution using the ‘sort_list’ function.
11. Finally, the script returns the optimal route for each cluster and a list of lists to check the routing in the routing tools provided as input of the project.

7. Results

This segment gives an overview of the results that were obtained using the python script.

- **Forecasting result**

Let's say a user wants to predict the fill rate at location '99-1' and waste type 'BX'. The user should change the value on the python script 'Forecasting.py'

```
# ///USER INPUTS///
d = '99-1' #Enter the point where the prediction has to be made. Make sure it is in 'str' format
m = 0 # m = 0,1,2,3 for BX,Verre,OM,Carton respectively
```

Figure 2: User input to predict the Fill rate

The result after executing the script will be

```
✓ 1m 27.5s
RMSE using Exponential smoothing:  0.23413021442737067
RMSE using AR Model:  0.22036481019402726
RMSE using ARMA Model:  0.22036367700529086
RMSE using ARIMA Model:  0.23452897587605886
RMSE using SARIMA Model:  0.23373901612781983
minimun mse was found using ARMA Model
min_mse = 0.22036367700529086
   prediction       date n_point waste_type
0    0.786391 2022-06-15    99-1         BX
1    0.787338 2022-06-17    99-1         BX
2    0.788253 2022-06-19    99-1         BX
3    0.789138 2022-06-21    99-1         BX
4    0.789993 2022-06-23    99-1         BX
5    0.790821 2022-06-25    99-1         BX
6    0.791621 2022-06-27    99-1         BX
7    0.792395 2022-06-29    99-1         BX
8    0.793143 2022-07-01    99-1         BX
9    0.793866 2022-07-03    99-1         BX
```

Figure 3 : Prediction Results

- **Clustering Results**

The User Inputs the type of waste and date on which the clustering has to be made.

```
# ///USER INPUTS///
date = '2022-07-19' #Date on which the clustering need to be done
m = 0 # Input type of waste - 0,1,2,3 for BX,Verre,OM,Carton respectively
```

Figure 4: User input to Cluster sites on a particular date

The script runs and returns a cluster in form of list of lists with locations.

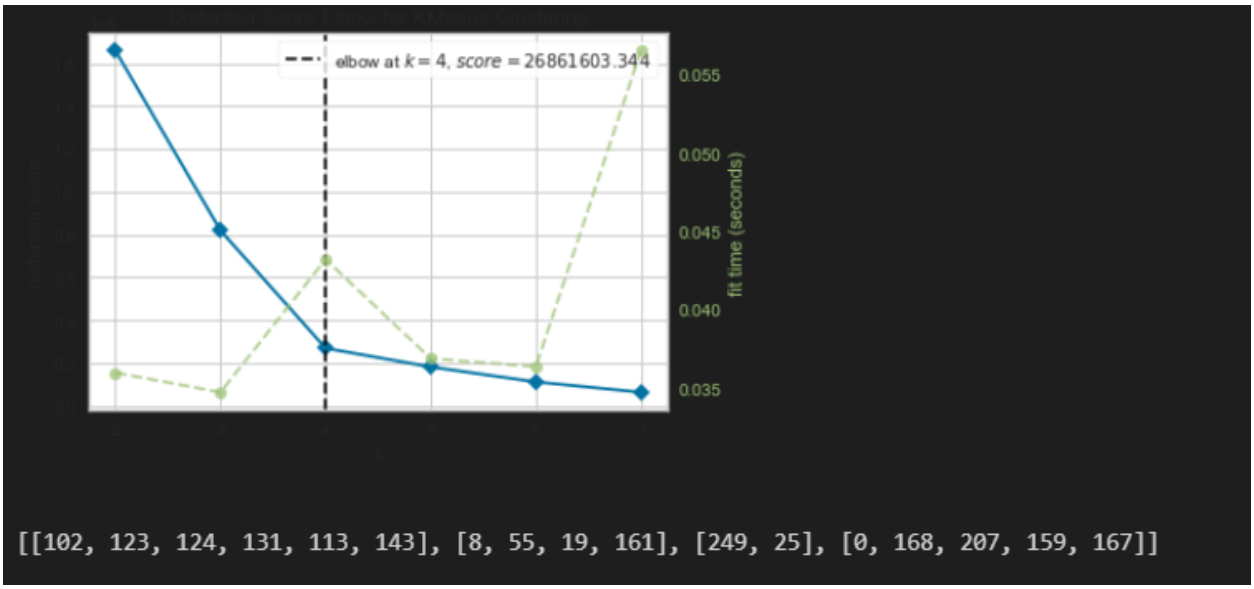


Figure 5 : Clustering Output along with

The graph shows the optimum number of clusters found using the elbow method by the 'KElbowVisualizer' class. Here, k = 4 and the clusters of cities are [[102, 123, 124, 131, 113, 143], [8, 55, 19, 161], [249, 25], [0, 168, 207, 159, 167]].

- **VRP results**

Using the output from the clustering algorithm and running the VRP script gives the following output

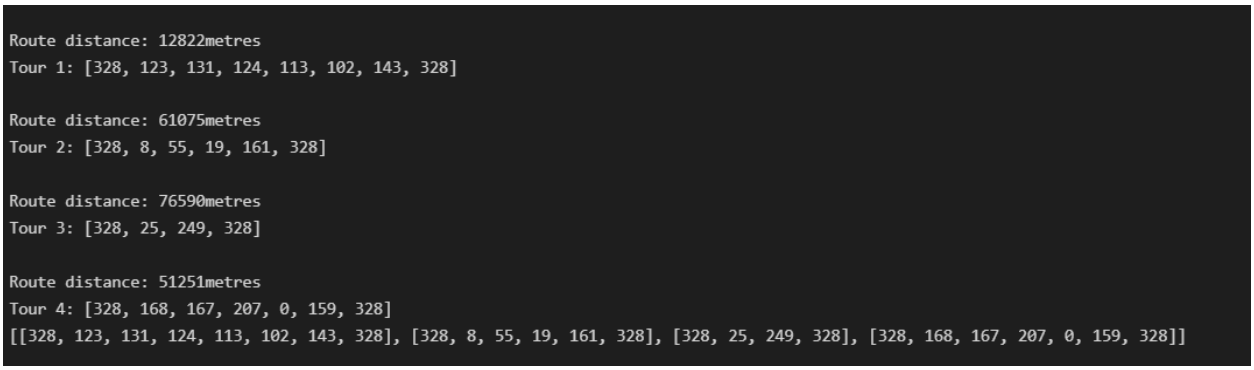


Figure 6 : VRP results

The last line of the output will be used to display the routes on the map using the tool that was provided during the project beginning. The result of the routing on '19-07-2022' is given below

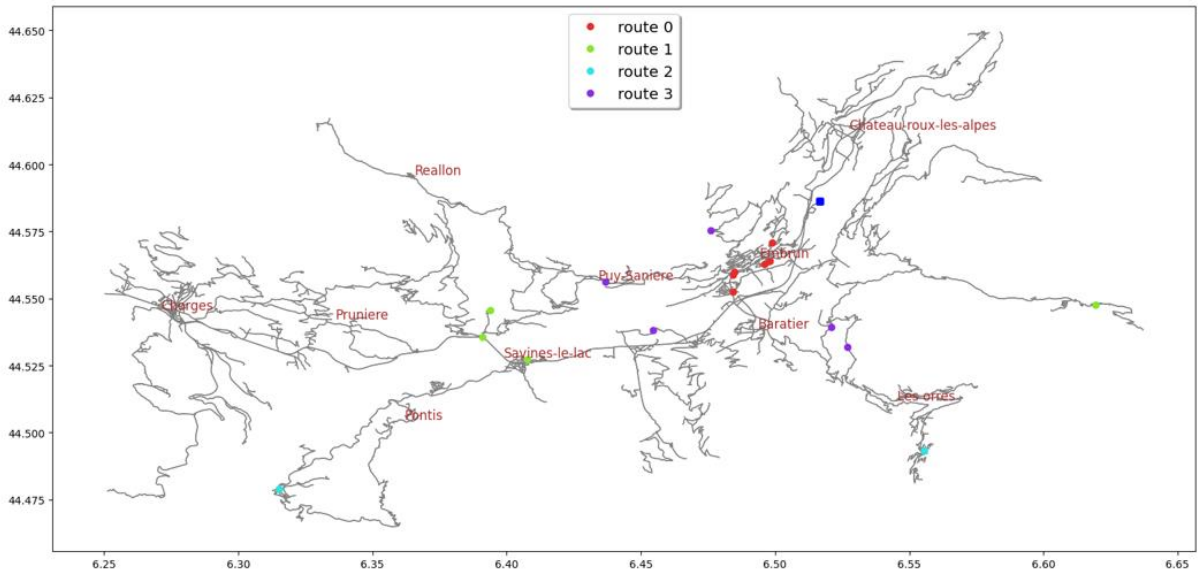


Figure 7: Offline Map output indicating the clustered site

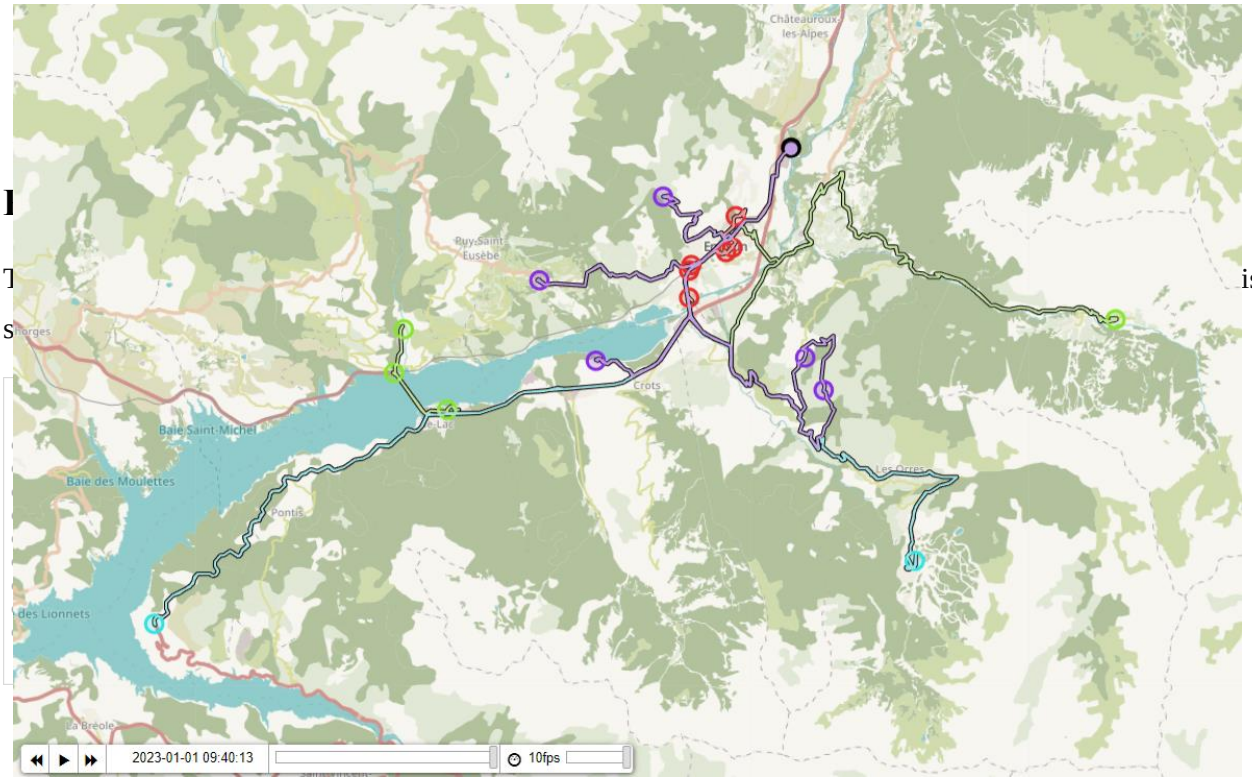


Figure 8 : Routes for the clustered sites

We found that the RMSE values for waste types varied between 0.05 to 0.3, with the exception of Carton, for which certain points had RMSE values greater than 0.5. The presence of outliers in the data set can also contribute to a higher RMSE value. RMSE is sensitive to outliers, which means that extreme values can have a greater impact on the overall score. This highlights the importance of identifying and addressing outliers in the data set.

To improve the model's predictions and lower the RMSE value, one could gather more data points for the specific waste type in regular intervals of time.

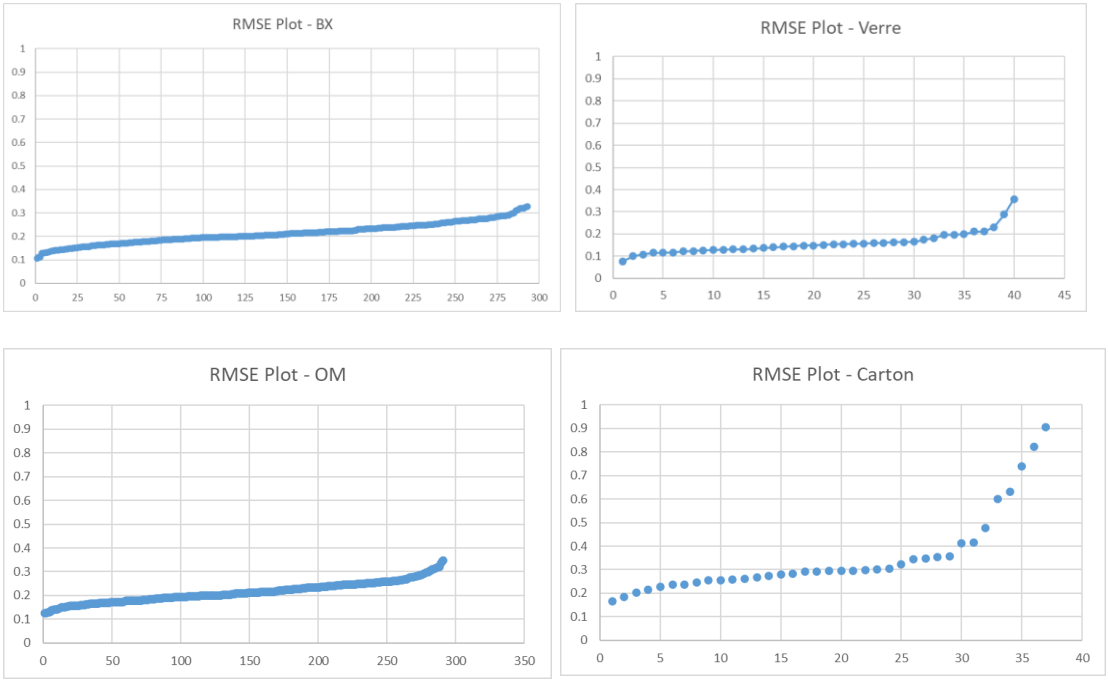


Figure 9: RMSE values graphs for the 4 types of waste

8. Conclusion

In this project, we conducted a study over the demand analysis of waste collection of the commune of Serre Ponçon by defining clusters and then a routing of vehicles over the denied clusters. The main goal was to analyze the given data and conclude a demand based on the location, the capacity and the different constraints of the waste collection. To be able to do that we first went over the forecasting part of the project by using 5 forecasting models and chose the one with the lowest RMSE. Then for the clustering, we directed a K-means clustering model to group a defined collection of locations in the Serre Ponçon commune. Finally, the vehicle routing was set by VRP method to give an example of an optimized routing between locations of each cluster. Like any other project, the study faced multiple issues that limited our work and we state:

- Inconsistent data or not enough data at some places to make the forecast. Some places had just one reading.
- Only fill rate is taken into consideration for the forecast and the predicted date is just the average of days before the last reading added to the latest reading.
- The Multiple Forecasting Script takes a long time to run as the locations are huge in number. For reference, refer to Image 2. Prediction on one location takes 97s
- The clustering is done using a basic unsupervised machine learning model which may not give the 'true' clusters within the data points.
- The VRP does not optimize the routes for the entire list of cities on a particular date. It rather optimizes the route of each cluster.

Further work can be done based on these limitations and as a continuity to the work considering the cited points.

References

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2016), *Time Series Analysis: Forecasting and Control* (Fifth edition), Hoboken, New Jersey: John Wiley & Sons, Inc. Available at *https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=1061322*

Hyndman, R. J., and Athanasopoulos, G. (2018] -), *Forecasting: Principles and Practice* (2nd edition), Melbourne: OTexts. Available at *https://otexts.com/fpp2/*

Hyndman, R. J., and Khandakar, Y. (2008), "Automatic Time Series Forecasting: The Forecast Package for R," *Journal of Statistical Software* [online], 27, 1–22, DOI: 10.18637/jss.v027.i03. Available at *https://www.jstatsoft.org/article/view/v027i03*

Perron, Laurent, Vincent Furnon (2019), "Or-Tools," Author={Perron, Laurent} [online]. Available at *https://scholar.google.com/citations?user=umrglaiaaaaj&hl=en&oi=sra*

Suradhaniwar, S., Kar, S., Durbha, S. S., and Jagarlapudi, A. (2021), "Time Series Forecasting of Univariate Agrometeorological Data: A Comparative Performance Evaluation via One-Step and Multi-Step Ahead Forecasting Strategies," *Sensors (Basel, Switzerland)* [online], 21, DOI: 10.3390/s21072430. Available at : *https://www.researchgate.net/publication/350546596_Time_Series_Forecasting_of_Univariate_Agrometeorological_Data_A_Comparative_Performance_Evaluation_via_One-Step_and_Multi-Step_Ahead_Forecasting_Strategies*

OR-Tools v9.5. Laurent Perron and Vincent Furnon. https://developers.google.com/optimization/ - Travelling Salesman problem