

Snap4Waste

AMSTERDAM WASTE CONTAINER

Studenti

- Mirco Ceccarelli (7042018)
- Pietro Zarri (7042627)

Docente

- Prof. Paolo Nesi

Supervisori

- Luciano Alessandro Ipsaro Palesi
- Enrico Collini



Indice

1. Introduzione.....	pag. 2
2. Raccolta dei Dati.....	pag. 3
1. Amsterdam Data Services.....	pag. 3
2. REST API Utilizzate.....	pag. 3
1. Container REST API.....	pag. 4
2. Cluster REST API.....	pag. 4
3. Clusterfractie REST API.....	pag. 4
4. Weging REST API.....	pag. 5
3. Data Ingestion	pag. 6
1. Snap4City.....	pag. 6
2. Container Ingestion.....	pag. 7
3. Measures Ingestion.....	pag. 8
4. Dashboard.....	pag. 9
4. Data Analysis.....	pag. 10
1. Importazione dei Dati da Snap4City.....	pag.10
2. Statistiche sui Rifiuti.....	pag. 10
1. Mappatura dei Container per Dimensione.....	pag. 13
3. Analisi dei Percorsi di Raccolta dei Camion.....	pag. 14
1. Animazione Registrazione Misure Giornaliere.....	pag. 14
2. Algoritmo per la Suddivisione dei Percorsi.....	pag. 14
3. Visualizzazione dei Percorsi.....	pag. 16
4. Statistiche sui Percorsi.....	pag. 17
5. Conclusioni.....	pag. 19
6. Sviluppi Futuri.....	pag. 19
7. Bibliografia.....	pag. 20

1 - Introduzione

L'obiettivo di questo elaborato per l'esame di Big Data Architectures è stato quello di collezionare e analizzare i dati sulla raccolta dei rifiuti nella città di Amsterdam.

La prima fase ha riguardato la raccolta dei dati presenti sul sito *open data* del comune di Amsterdam, che riguardano le misurazioni effettuate, da parte degli operatori ecologici, allo svuotamento dei cassonetti interrati.

La seconda fase si è concentrata sull'analisi dei dati e sul calcolo di varie statistiche, suddivise sia per tipologia di rifiuto che per intervalli temporali.

Infine sono state realizzate alcune tecniche per determinare possibili percorsi effettuati dai mezzi di raccolta dei rifiuti e per stimare il numero di veicoli impiegati per lo svolgimento di tali operazioni.

2 - Raccolta dei Dati

Nella fase preliminare di questo progetto sono stati analizzati i dati riguardanti la raccolta dei rifiuti presenti sulle piattaforme *open data* di varie città di tutto il mondo, tra cui: Amsterdam, Anversa, Parigi, Helsinki, Milano, Copenaghen, Londra, Berlino, Buffalo (New York).

Amsterdam è però l'unica città, tra quelle sopra elencate, che mette a disposizione i dati riguardanti le misurazioni effettuate alla raccolta del singolo cassonetto.

2.1 - Amsterdam Data Services

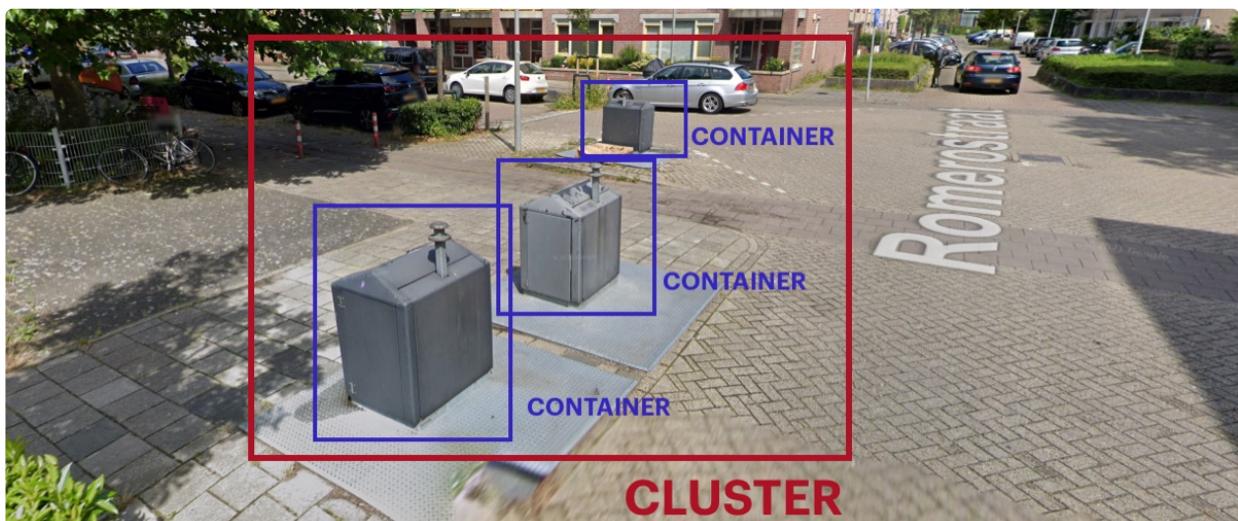
La piattaforma *open data* di Amsterdam viene aggiornata quotidianamente con i dati su diverse tipologie di rifiuti: [1]

- Indifferenziata (**Rest**);
- Carta e Cartone (**Papier**);
- Vetro (**Glas**);
- Plastica (**Plastic**);
- Tessuti e Scarpe (**Textiel**).

La categoria *Textiel* però non è stata presa in considerazione per tale elaborato a causa della scarsità delle misure registrate.

Due concetti fondamentali da tenere in considerazione durante l'utilizzo delle API sono i seguenti:

- **Container** → si intende un singolo contenitore interrato dove vengono raccolti i rifiuti di una stessa tipologia;
- **Cluster** → si intende un insieme di container collocati vicini tra loro.



All'interno della piattaforma sono registrati 5701 cluster e 17329 container.

2.2 - REST API Utilizzate

Per accedere ai dati sono a disposizione le API sui seguenti endpoint: [2]

- **Container REST URI:** <https://api.data.amsterdam.nl/v1/huishoudelijkafval/container/>
- **Cluster REST URI:** <https://api.data.amsterdam.nl/v1/huishoudelijkafval/cluster/>
- **Clusterfractie REST URI:** <https://api.data.amsterdam.nl/v1/huishoudelijkafval/clusterfractie/>
- **Weging REST URI:** <https://api.data.amsterdam.nl/v1/huishoudelijkafval/weging/>

2.2.1 - Container REST API

API che elenca tutti i **container** della città di Amsterdam. I campi più rilevanti sono i seguenti:

Nome	Descrizione	Tipo
id	Identificativo univoco dell'oggetto	String
serienummer	Numero di serie rilasciato dal produttore	String
clusterId	Identificato univoco del cluster di cui fa parte il container	String
eigenaarId	Identificativo del proprietario	String
eigenaarNaam	Nome del proprietario	String
status	Stato del contenitore: 0-inattivo, 1-attivo, 2-pianificato	Integer
fractieCode	Tipologia container: 1-Rest, 2-Glas, 3-Papier, 4-Plastic, 5-Textiel	String
fractieOmschrijving	Descrizione della frazione del contenitore fornita dal fornitore	String
datumCreatie	Data in cui l'oggetto è stato creato nel sistema di gestione dei container	String
datumPlaatsing	Data in cui l'oggetto è stato posizionato	String
datumOperationeel	Data da cui il container è operativo	String
locatielid	Identificativo della fossa in cui è posizionato il container	String
geometrie	Localizzazione del container (in coordinate RD)	Point

2.2.2 - Cluster REST API

API che elenca tutti i **cluster** della città di Amsterdam. I campi più rilevanti sono i seguenti:

Nome	Descrizione	Tipo
id	Identificativo univoco del cluster	String
subclusterIndicatie	Indica se si tratta di un cluster diviso da una strada	Boolean
geometrie	Localizzazione del cluster (in coordinate RD)	Point
datumOntstaan	Data di inserimento del cluster nel sistema	String
status	Stato del cluster (0-inattivo, 1-attivo)	Integer

2.2.3 - Clusterfractie REST API

API che elenca la **divisione di ogni singolo cluster nei suoi container**, specificando la tipologia di rifiuto del container. I campi più rilevanti sono i seguenti:

Nome	Descrizione	Tipo
id	Identificativo univoco della suddivisione del cluster	String
clusterId	Identificativo univoco del cluster a cui è collegato il container	String
aantalContainers	Numero di container per cluster	Integer
volumeM3	Somma dei volumi (m^3) dei container per cluster	Number
code	Tipologia container: 1-Rest, 2-Glas, 3-Papier, 4-Plastic, 5-Textiel	String
omschrijving	Tipologia dei rifiuti del container	String

2.2.4 - Wegin REST API

API che elenca tutte le **pesate** effettuate sui container della città di Amsterdam. I campi più rilevanti sono i seguenti:

Nome	Descrizione	Tipo
id	Identificativo univoco della pesata	String
clusterId	Identificativo univoco del cluster	String
clusterSubclusterIndicatie	Indica se il cluster è diviso da una strada	Boolean
weegsysteemId	Identificativo del sistema di pesatura (associato al veicolo)	Integer
weegsysteemOmschrijving	Descrizione del sistema di pesatura	String
volgnummer	Numerazione crescente delle pesate per sistema di pesatura	Integer
datumWeging	Data della pesatura (Y - M - D)	String
tijdstipWeging	Ora della pesatura (H - M - S)	String
locatienummer	Numero della posizione	String
fractieCode	Codice frazione di rifiuto fornito da Welvaarts, 1: Rest, 2: Glas, 3: Papier, 4: Plastic, 5: Textiel	Integer
fractieOmschrijving	Descrizione della frazione di rifiuto	String
eersteWeging	Peso Lordo (Kg)	Number
tweedeWeging	Peso Tara (Kg)	Number
nettoGewicht	Peso Netto (Kg)	Number
geometrie	Localizzazione della pesatura (in coordinate RD)	Point (x, y)

È necessario aggiungere le seguenti spiegazioni:

- Alcune misurazioni possono presentare delle **anomalie** nel valore del peso, che accadono quando il container è pesato troppo velocemente durante il sollevamento, senza che sia completamente appeso al braccio meccanico.
- Ogni pesata (*Weging*) ha come attributo l' **id** di un cluster, ma concretamente si riferisce a uno dei container in cui il cluster è suddiviso.

3 - Data Ingestion

Al fine di processare e analizzare i dati, essi sono stati caricati tramite le REST API della città di Amsterdam all'interno della piattaforma **Snap4City**. [3]

3.1 - Snap4City

Tramite la piattaforma smart *Snap4City*, il primo passo è stato quello di creare gli **IOT Device Model**, che per questo elaborato si riferiscono ai container.

È stato quindi necessario eseguire un'analisi preliminare per capire cosa dovesse essere incluso all'interno dei *Device Model*.

Sono stati creati quattro diversi Device Model, uno per ogni tipologia di rifiuto:

- **AmsterdamRestContainer**;
- **AmsterdamGlasContainer**;
- **AmsterdamPapierContainer**;
- **AmsterdamPlasticContainer**.

Ciascuno di essi presenta gli stessi due attributi statici:

- La tipologia del rifiuto che il cassetto contiene;
- La capacità massima del cassetto.

Sono stati inoltre assegnati i seguenti valori:

- *dateObserved* → valore obbligatorio sui modelli di *Snap4City* ed indica il momento in cui è stata eseguita una misurazione su un cassetto;
- *weight* → indica il peso in Kg di una misurazione su un cassetto al momento dello svuotamento.

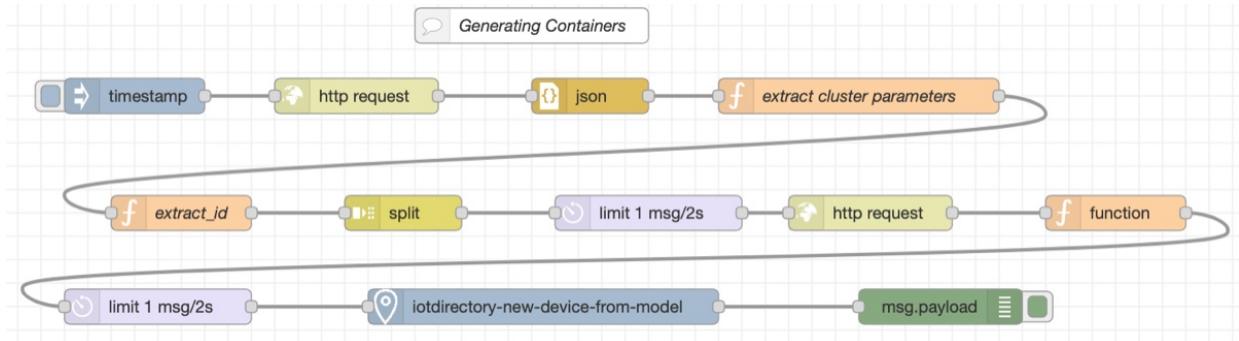
The screenshot shows the Snap4City interface. On the left, there is a list of '192 MODELS' with a search bar and a table showing four entries: 'AmsterdamRestContainer', 'AmsterdamGlasContainer', 'AmsterdamPapierContainer', and 'AmsterdamPlasticContainer'. The table includes columns for 'Device Model', 'Producer', 'Healthiness Criteria', and 'Key Generation'. On the right, a detailed view dialog for 'AmsterdamRestContainer' is open. The dialog has tabs for 'General Info', 'IoT Broker', 'Static Attributes', and 'Values'. Under 'General Info', it says 'Rest waste container in Amsterdam'. Under 'Static Attributes', it lists 'Name' (AmsterdamRestContainer), 'Description' (Rest waste container in Amsterdam), 'WeightSensor' (Sensor), 'Device Type' (Kind), 'Producer' (Amsterdam city), 'Frequency' (600), 'Healthiness Criteria' (Refresh Rate, 300), 'Healthiness Value' (300), 'Key Generation' (Automatically generated), and 'Edge-Gateway Type'. At the bottom of the dialog is a 'Cancel' button. To the right of the dialog, there is a grid of buttons for 'Edit', 'Delete', and 'View' for each model in the list.

| Modello **AmsterdamRestContainer** sulla piattaforma Snap4City.

Una volta aggiunti correttamente i quattro modelli alla piattaforma *Snap4City*, è stata creata una **IOT Application** per eseguire, tramite un flusso Node-RED [4], sia la *Container Ingestion* che la *Measures Ingestion*.

3.2 - Container Ingestion

L'obiettivo di questa fase è stato quello di creare un **IOT Device** per ciascun container su cui andar poi a caricare le misurazioni. Per fare ciò è stato utilizzato il seguente flusso Node-RED:



Flusso per eseguire la creazione dei container.

I passaggi eseguiti sono i seguenti:

1. Tramite il nodo "*http request*" viene eseguita una chiamata GET all'endpoint:

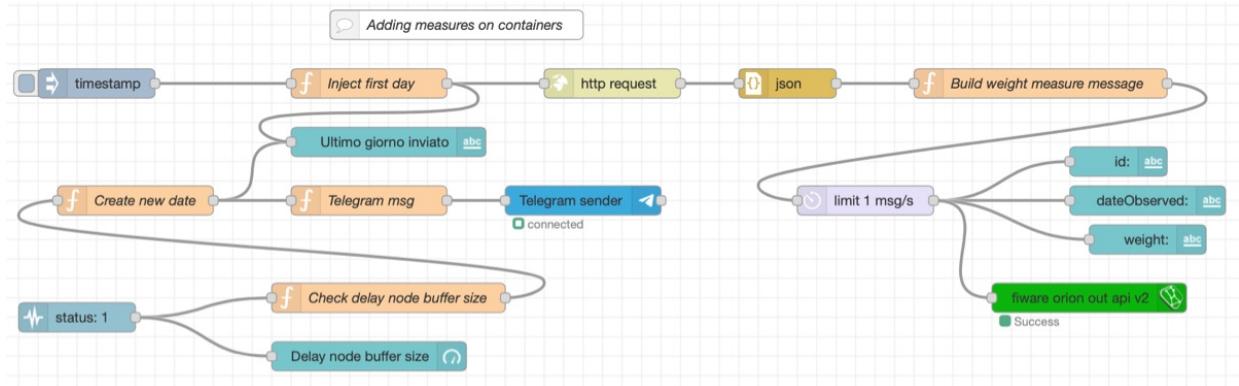
https://api.data.amsterdam.nl/v1/huishoudelijkafval/cluster/?_pageSize=100000&_format=json.

Il parametro `pageSize` è stato aggiunto per evitare la paginazione dei risultati.

2. Il nodo "*extract cluster parameters*" estraie dal json di risposta i parametri necessari per identificare il cluster: id e posizione. Le coordinate sono state converte dal sistema olandese RD (Rijks-Driehoek) a wgs_84.
3. Tramite l'*id* di ogni cluster viene poi eseguita una nuova richiesta GET all'endpoint:
<https://api.data.amsterdam.nl/v1/huishoudelijkafval/clusterfractie?clusterId={{id}}>.
4. Combinando i valori delle due richieste GET viene composto l'*id* univoco del *Device* concatenando l'*id* del cluster, a cui il cassonetto appartiene, con la tipologia di rifiuto. Viene assegnato quindi ogni *Device* al proprio *Device Model* di appartenenza.
5. Infine la creazione vera e propria del *Device* viene eseguita all'interno del nodo "*iot-directory-new-device-from-model*".

3.3 - Measures Ingestion

L'obiettivo di questa fase invece è stato quello di caricare le misurazioni registrate allo svuotamento dei singoli container, assegnandole ognuna a ciascun Device. In particolare sono stati presi in considerazione i dati dal 1 Gennaio 2020 al 31 Dicembre 2021. Per fare ciò è stato utilizzato il seguente flusso Node-RED:



Flusso per eseguire il caricamento delle misurazioni ai container.

I passaggi eseguiti sono i seguenti:

1. Tramite il nodo "http request" viene eseguita una chiamata GET all'endpoint:

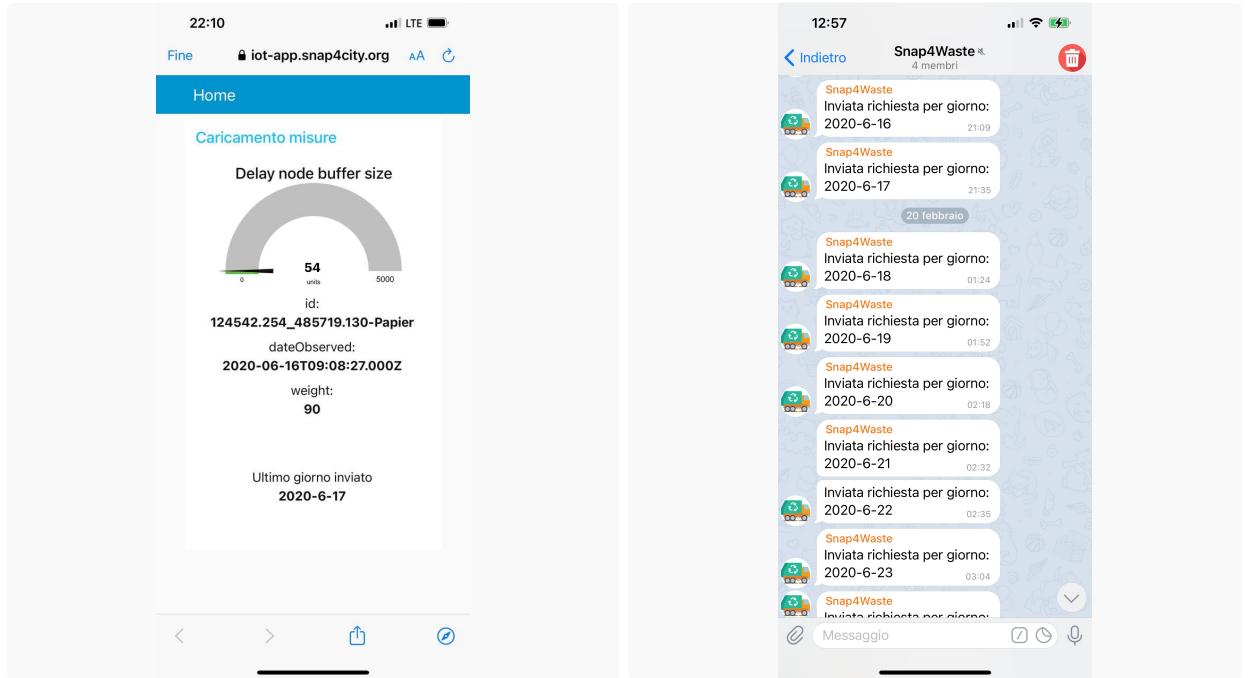
https://api.data.amsterdam.nl/v1/huishoudelijkafval/weging/?datumWeging={{date}}&_pageSize=5000&_format=json.

Per non sovraccaricare il flusso con troppi dati le richieste vengono eseguite separatamente per ciascun giorno sull'intervallo dei due anni.

2. Con il nodo "build weight measure message" vengono estratti dal json di risposta i valori necessari alla creazione della misura: id del cluster, peso e data.
3. Il messaggio con tali valori viene mandato quindi in ingresso a un buffer (1 msg/sec) e successivamente al nodo "fiware orion out api v2", che carica sulla piattaforma Snap4City le singole misure.
4. Quando il buffer, monitorato tramite il nodo "status", si svuota, attiva il nodo "create new date" che manda in input al nodo "http request" la data successiva da cui estrarre le misure.

Per monitorare l'andamento del processo di caricamento in tempo reale, sono stati creati una

Dashboard e un **Bot Telegram**:



| Dashboard e Bot Telegram per il monitoraggio *real-time* del caricamento delle misure.

3.4 - Dashboard

Usando il *Dashboard Builder* integrato in Snap4City è stata realizzata una **Dashboard** per visualizzare su di una mappa i container suddivisi nelle quattro tipologie di rifiuto e tramite una *timeline* l'andamento delle misure effettuate su un singolo cassetto selezionato.



| Dashboard AmsterdamWasteContainer.

4 - Data Analysis

Al termine della fase di *Data Ingestion*, sono state eseguite una serie di analisi statistiche sui dati per mettere in evidenza l'andamento sulla produzione dei rifiuti della città di Amsterdam nell'intervallo da noi considerato (2020-2021). Con queste informazioni sono stati inoltre stimati dei possibili percorsi seguiti dai camion durante i turni di raccolta dei rifiuti urbani.

4.1 - Importazione dei Dati da Snap4City

Le analisi sono state eseguite in **Python** su **Jupyter Notebook**, utilizzando le librerie *open source* **NumPy** [5] e **Pandas** [6].

Per ottenere i dati caricati sulla piattaforma Snap4City è stato necessario essere in possesso dell'**Auth Token** relativo al profilo *Snap4Waste*:

```
url = "https://www.snap4city.org/auth/realms/master/protocol/openid-connect/token/"
data = {"client_id": "jupyterhub-pontdugard",
        "grant_type": "password",
        "username": "***",
        "password": "***"}

r=requests.post(url, data)
responseToken=json.loads(r.text)
auth_token=responseToken['access_token']
hed = {'Authorization': 'Bearer ' + auth_token}
```

Dopo aver ottenuto la lista degli *id* di tutti i container sono stati creati quattro file csv (uno per ciascuna tipologia di rifiuto) contenenti tutte le misure registrate sulla piattaforma.

```
url = "https://www.snap4city.org/superservicemap/api/v1?serviceUri=http://
www.disit.org/km4city/resource/iot/orionUNIFI/
DISIT/" + id + "&fromTime=2020-01-01T00:00:00"
response = requests.get(url, headers=hed).json()
```

| API Snap4City per recuperare le misure registrate su un determinato device tramite id.

Per semplicità le operazioni che verranno descritte nei paragrafi successivi sono state eseguite solamente sulle misurazioni relative ai cassonetti dell'indifferenziata in quanto è la tipologia di rifiuto con il maggior numero di misure caricate. Ad ogni modo le stesse operazioni descritte possono essere ripetute anche per le altre tre tipologie.

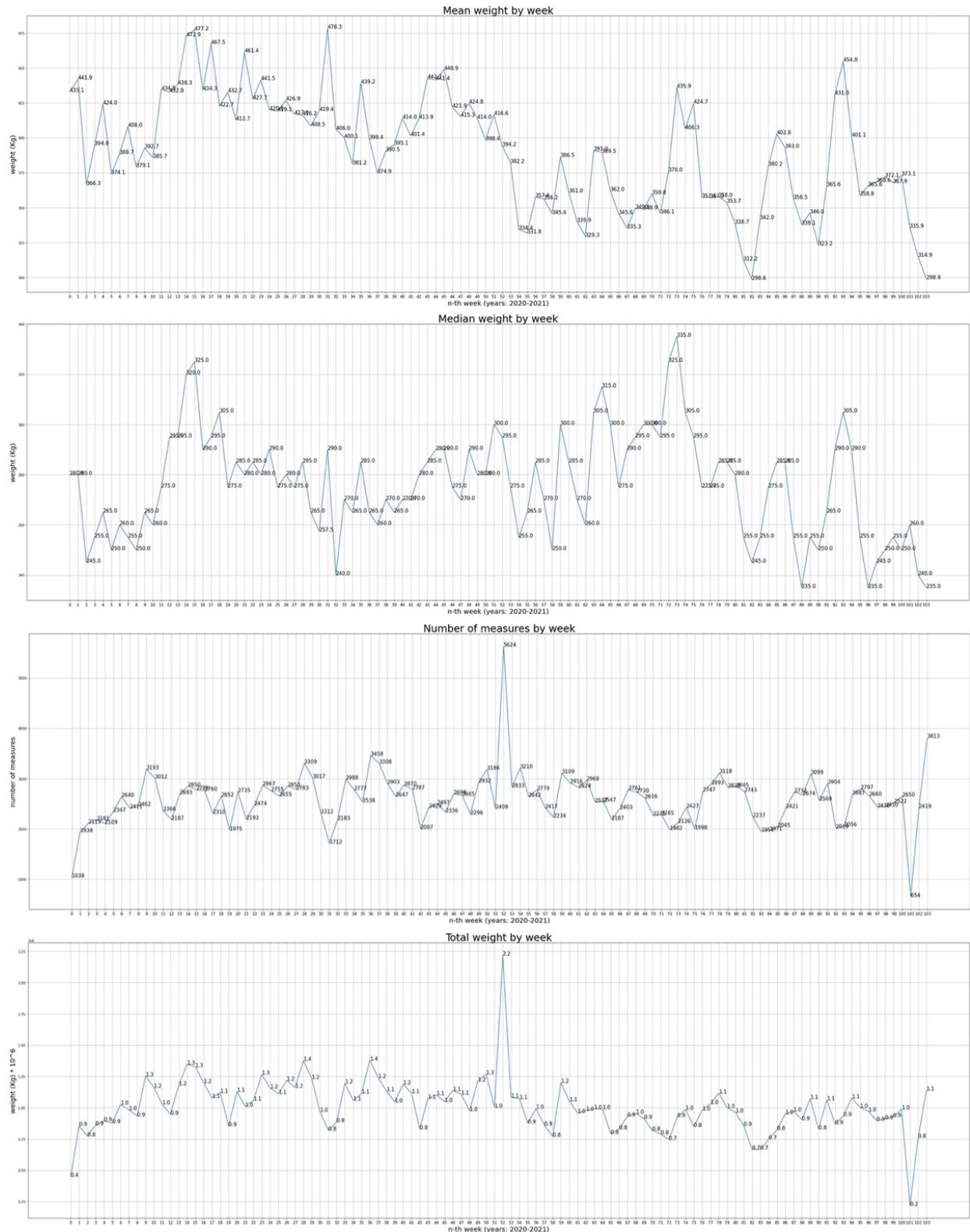
4.2 - Statistiche sui Rifiuti

Dopo aver recuperato le misure, come descritto nel paragrafo precedente, sono state calcolate le seguenti statistiche (sia settimanali che mensili):

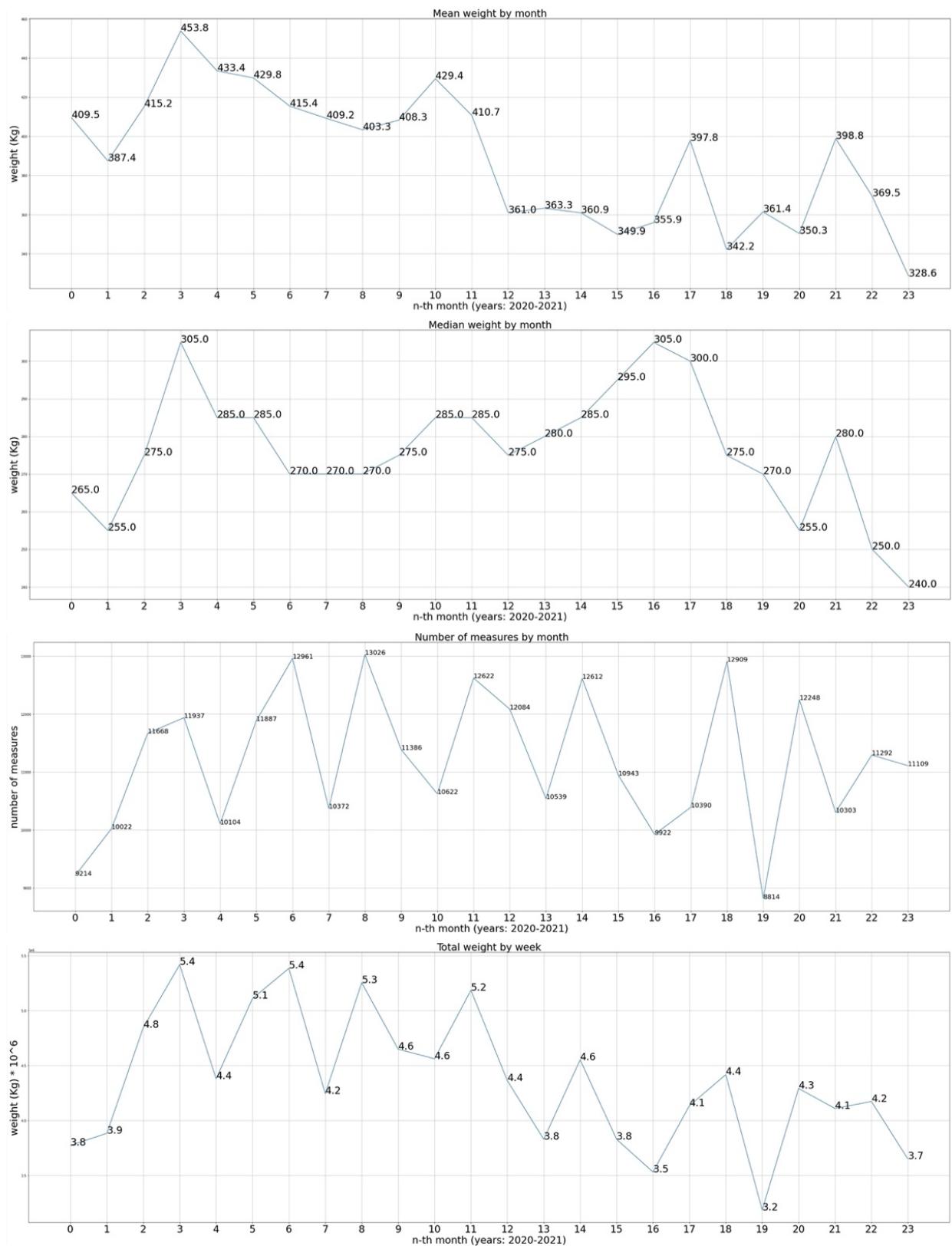
- Media del peso dei rifiuti raccolti;
- Mediana del peso dei rifiuti raccolti;
- Numero di misurazioni effettuate;

- Peso totale dei rifiuti raccolti.

Successivamente tramite la libreria **Matplotlib** [7] tali valori sono stati rappresentati su dei grafici per capirne meglio l'andamento.

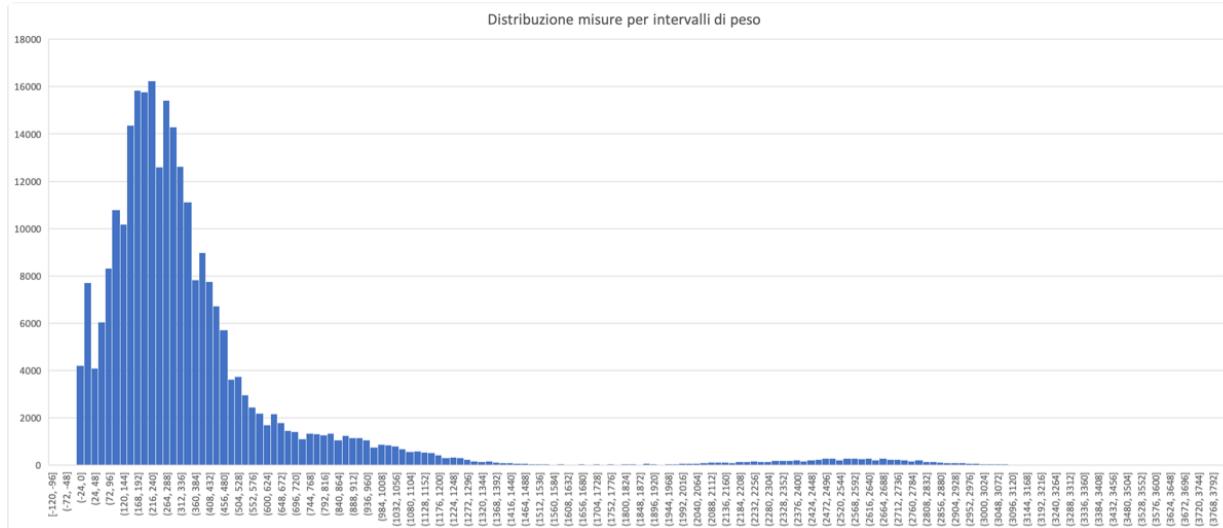


Media, mediana, numero di misure e peso totale raccolto settimanalmente per l'indifferenziata (Rest).



| Media, mediana, numero di misure e peso totale raccolto mensilmente per l'indifferenziata (Rest).

Tramite Excel è stato ricavato anche il grafico della distribuzione per intervalli di peso delle misure registrate:

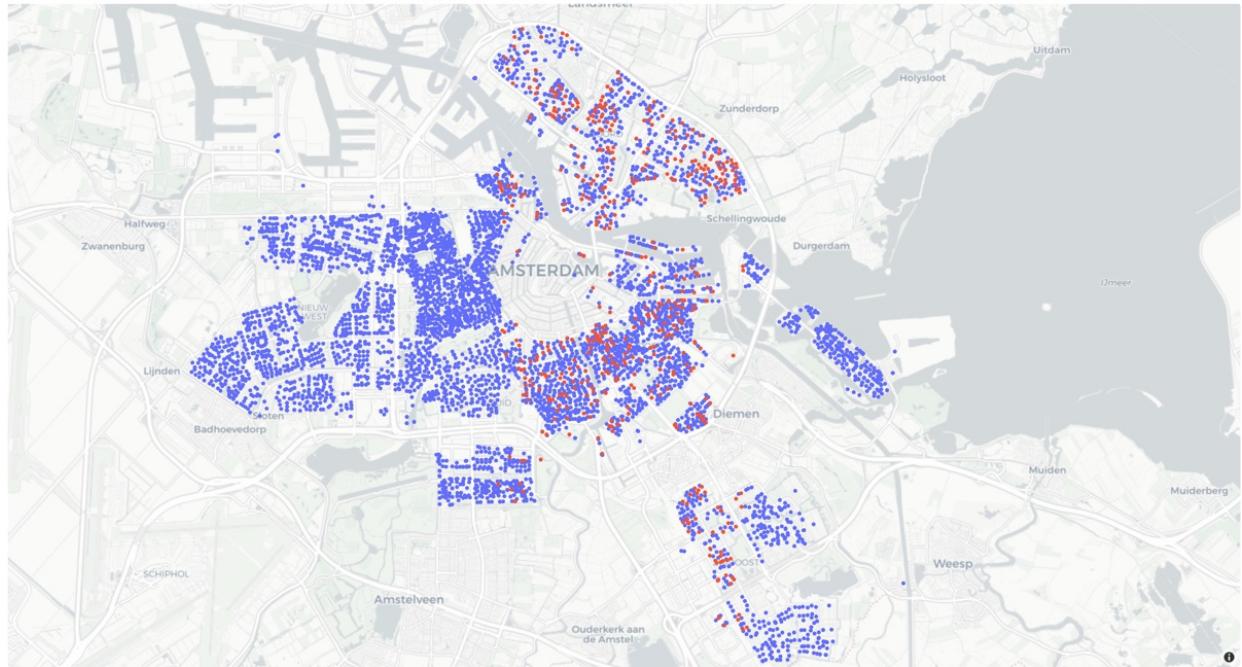


Come si evince dall'istogramma, l'andamento è bimodale: un picco maggiore di misure si registra intorno ai 230 Kg un altro minore, ma comunque notabile, intorno ai 2500 Kg.

Questo suggerisce la presenza di due tipologie di cassonetto nella città di Amsterdam. La prima tipologia in numero molto maggiore con una capacità di poco superiore ai 1000 Kg, la seconda invece in numero minore con una capacità intorno ai 3000 Kg.

4.2.1 - Mappatura dei Container per Dimensione

Le due tipologie sopra elencate sono state rappresentate su una mappa, tramite l'utilizzo della libreria **Plotly** [8], per visualizzarne la distribuzione sul territorio della città di Amsterdam.



| Mappa dei container interrati suddivisi per capienza.

In **blu** sono evidenziati i container di dimensione più piccola basandosi sul loro peso medio raccolto sui due anni di dati, mentre in **rosso** sono evidenziati i cassonetti di capacità maggiore. Come valore di soglia per la distinzione tra le due tipologie è stato preso il valore di 1500 Kg.

4.3 - Analisi dei Percorsi di Raccolta dei Camion

L'ultima fase di questo elaborato è stata la realizzazione di un algoritmo per identificare i possibili percorsi seguiti dai mezzi di raccolta dei rifiuti urbani all'interno della città di Amsterdam, utilizzando unicamente le misurazioni effettuate nei due anni presi in analisi.

4.3.1 - Animazione Registrazione Misure Giornaliera

Per comprendere in che modo le misurazioni si susseguono nell'arco di una giornata, prima della scrittura dell'algoritmo, è stata creata una mappa interattiva tramite la libreria **Folium** [9].



| Andamento delle misurazioni nella giornata del 30 Dicembre 2021.

Questa mappa indica quindi la necessità di eseguire un raggruppamento che tenga conto non solo della disposizione spaziale delle misure, ma anche del momento della giornata in cui il cassonetto è stato svuotato.

Infatti è frequente che due misurazioni eseguite su due cassonetti vicini tra di loro siano però registrate in due momenti diversi della giornata.

4.3.2 - Algoritmo per la Suddivisione dei Percorsi

Tenuto conto di quanto detto al termine del paragrafo precedente, nella creazione dell'algoritmo per la ricostruzione dei percorsi, sono state prese in considerazione le seguenti regole.

Due misure si considerano appartenenti allo stesso percorso eseguito da un camion se si verifica una delle due seguenti condizioni:

- I due cassonetti si trovano a meno di una distanza geodetica (libreria **GeoPy** [10]) $x < \bar{x}$ e le due misure sono state eseguite a una distanza temporale $t < \bar{t}$;

- $x \leq v \cdot t$ con x la distanza spaziale tra i due cassonetti, t la distanza in secondi tra le due misurazioni e v la stima della velocità del camion e $x < \tilde{x}$.

Le due precedenti condizioni nell'implementazione dell'algoritmo sono entrambe soggette al valore di capacità massima C del camion che effettua la raccolta dei rifiuti.

La capacità massima dei camion è stata stimata tra i 12000 e 14000 Kg. [11][12][13][14]

Poichè ad ogni iterazione ogni misura viene confrontata con tutti i percorsi già creati secondo il loro ordine di creazione, è stato necessario introdurre un punteggio ($score = 2 \cdot x + t$) per valutare quale tra i percorsi che soddisfano le condizioni sia il più adatto.

Di seguito è riportata la porzione di codice che implementa quanto descritto:

```
def check_measure_in_path(self, measure):
    # get distance between measures
    last_measure = self.measures[-1]
    position1 = (last_measure.lat, last_measure.long)
    position2 = (measure.lat, measure.long)
    dst = geopy.distance.distance(position1, position2).km * 1000

    # get time between measures
    date1 = datetime.strptime(last_measure.date, "%Y-%m-%dT%H:%M:%S.%fZ")
    date2 = datetime.strptime(measure.date, "%Y-%m-%dT%H:%M:%S.%fZ")
    diff = (date2 - date1).seconds

    max_distance_allowed = diff * speed_ms

    score = 2 * dst + diff

    """
    Check that a measure is part or not of an existing path.
    The following parameter are used:
        C -> maximum weight capacity of the truck.
        x_tilde -> maximum allowed distance between two consecutive measures.
    """
    if dst < 1000 and diff < 3600 and (self.tot_weight + measure.weight) < C:
        return True, score
    if dst <= max_distance_allowed and (self.tot_weight + measure.weight) < C and
       dst < x_tilde:
        return True, score
    else:
        return False, score
```

Durante tutte le implementazioni sono stati mantenuti costanti i seguenti parametri:

- $v = 4.16 m/s$ (15 Km/h) \rightarrow `speed_ms` ;
- $\bar{x} = 1000 m$;
- $\bar{t} = 3600 sec$.

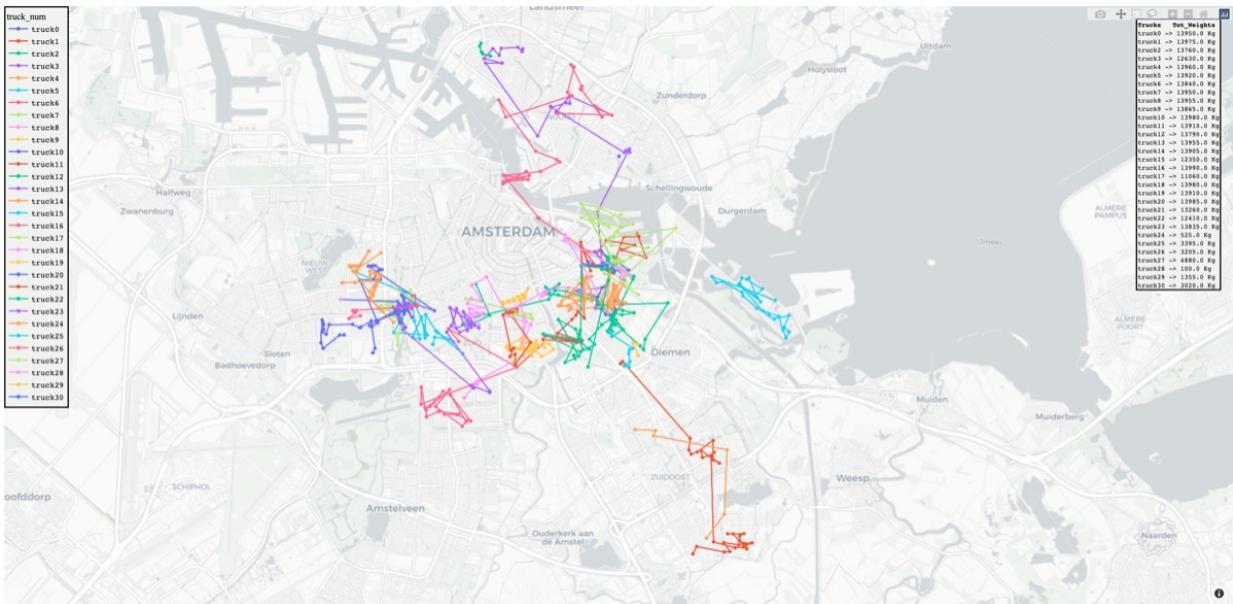
4.3.3 - Visualizzazione dei Percorsi

L'algoritmo descritto nella sezione precedente, una volta eseguito, produce un **file csv** che specifica per ogni misura il camion a cui è stata assegnata.

```
id,lat,long,truck_num,date,weight,sum_weight
124589.884_485235.611-Rest,52.35417210388034,4.9408326130590225,truck0,2021-05-27T06:37:15.000Z,130.0,130.0
124376.436_485335.106-Rest,52.35508210389528,4.93764261291122,truck0,2021-05-27T06:42:36.000Z,385.0,515.0
...
127405.785_485866.553-Rest,52.359952103774845,4.98227261510169,truck15,2021-05-27T10:03:58.000Z,450.0,1745.0
127977.419_485564.938-Rest,52.357192103730085,4.990522615481003,truck15,2021-05-27T10:11:10.000Z,225.0,1970.0
127986.352_485450.446-Rest,52.35620210372309,4.99094261549383,truck15,2021-05-27T10:14:54.000Z,345.0,2315.0
...
125008.823_484114.909-Rest,52.34410210380788,4.947302613304544,truck29,2021-05-27T13:55:49.000Z,390.0,1120.0
125008.823_484114.909-Rest,52.34410210380788,4.947302613304544,truck29,2021-05-27T13:56:46.000Z,235.0,1355.0
124518.816_489402.920-Rest,52.391612104120526,4.939512613236199,truck30,2021-05-27T14:09:40.000Z,3020.0,3020.0
```

| Esempio del file di output prodotto.

Per visualizzare su una mappa i dati è stata sfruttata sempre la libreria **Plotly**, che data una lista di misurazioni consente di tracciare il percorso eseguito dai mezzi di raccolta dei rifiuti.



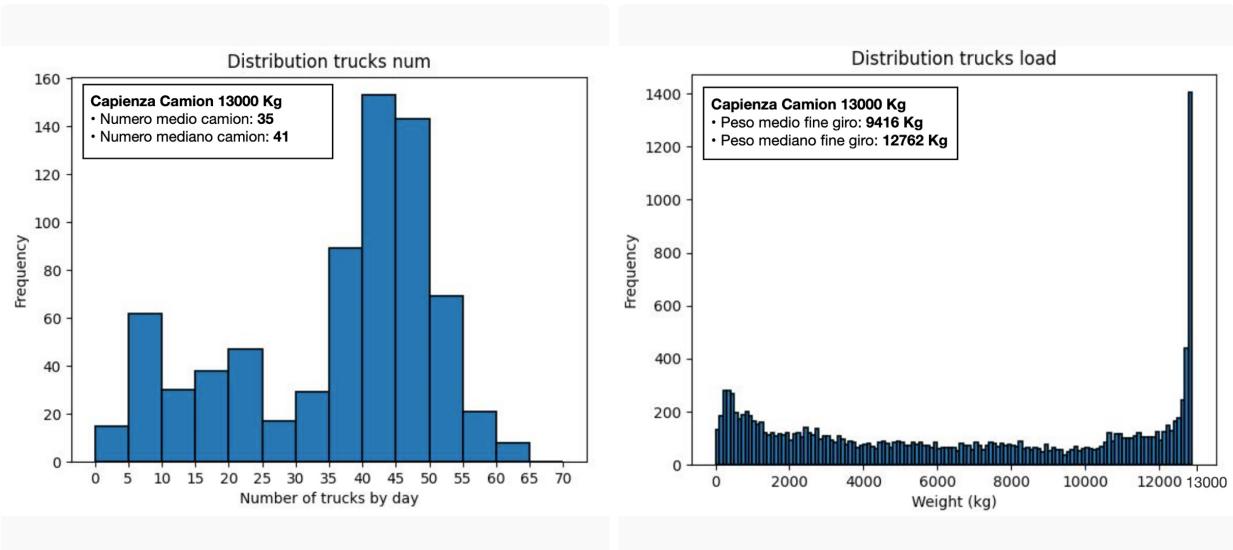
| Percorsi ricavati nella giornata di 27 Maggio 2021.

4.3.4 - Statistiche sui Percorsi

Per la prima esecuzione della procedura analizzata nella sezione 4.3.2 sono stati scelti i seguenti parametri:

- $\tilde{x} = 2000 \text{ m};$
- $C = 13000 \text{ Kg}.$

Iterando la procedura su tutti i giorni dei due anni presi in considerazione, è stato possibile visualizzare graficamente la distribuzione del numero di camion impiegati ogni giorno per la raccolta rifiuti e del rispettivo carico a fine giro.



I Distribuzione del numero di camion e relativo carico a fine giro con $\tilde{x} = 2000 \text{ m}$ e $C = 13000 \text{ Kg}.$

Prendendo in considerazione la distribuzione del carico dei camion a fine giro è possibile notare un andamento bimodale.

Oltre ad un picco attorno al valore di 13000 Kg (capacità massima), è presente anche un altro picco significativo nella parte sinistra della distribuzione, ciò indica che un numero rilevante di camion termina il giro di raccolta con pochi rifiuti all'interno del veicolo.

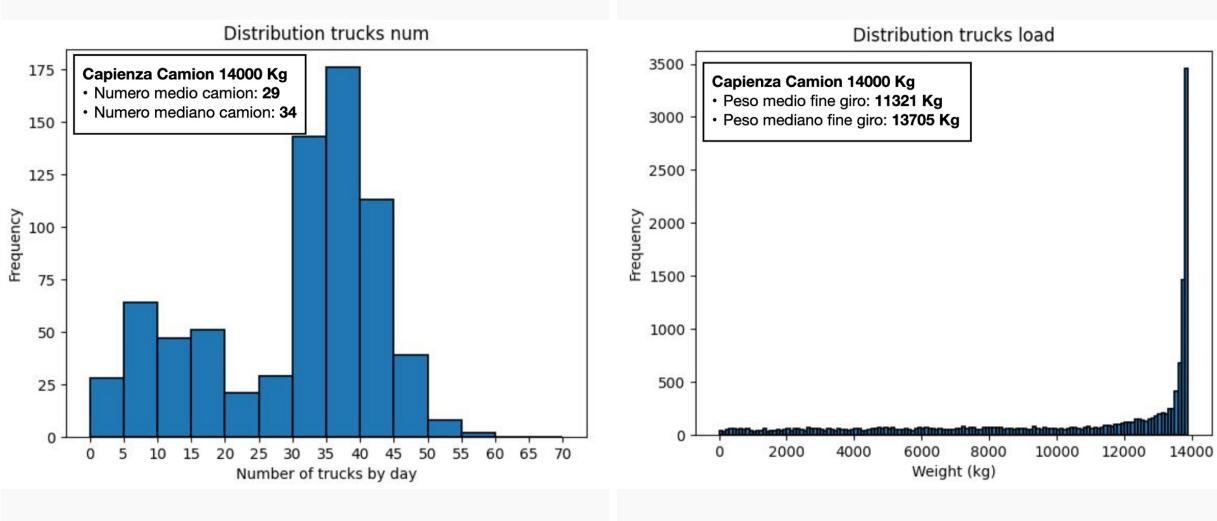
Tale comportamento è probabilmente irrealistico e suggerisce la necessità di aggiustare i parametri \tilde{x} e C .

Sapendo che i valori di capacità massima dei camion per la raccolta dei rifiuti si attestano tra i 12000 e i 14000 Kg, sono state eseguite ulteriori prove modificando i due parametri.

Il risultato migliore è stato ottenuto con i seguenti valori:

- $\tilde{x} = 3000 \text{ m};$
- $C = 14000 \text{ Kg}.$

Di seguito sono riportati i grafici sempre per le due distribuzioni:



I Distribuzione del numero di camion e relativo carico a fine giro con $\tilde{x} = 3000 \text{ m}$ e $C = 14000 \text{ Kg}$.

Dal secondo grafico è quindi possibile notare che con questi nuovi valori dei parametri, l'andamento bimodale sparisce ed è presente un solo picco in corrispondenza della capacità massima, segno che la maggior parte dei camion termina il suo giro sfruttando al massimo la sua capacità di raccolta.

Nella seguente tabella sono confrontati i dati dei due casi presi in considerazione nei due anni:

	$\tilde{x} = 2000$	$\tilde{x} = 3000$
	$C = 13000$	$C = 14000$
Numero camion carichi (capacità $\geq 75\%$)	16257 (63.3%)	15515 (72.6%)
Numero camion semi-carichi ($50\% \leq \text{capacità} < 75\%$)	2154 (8.4%)	2165 (10.1%)
Numero camion semi-vuoti ($25\% \leq \text{capacità} < 50\%$)	2511 (9.8%)	1923 (9.0%)
Numero camion vuoti (capacità $< 25\%$)	4762 (18.5%)	1761 (8.3%)

Le percentuali presenti nella tabella confermano dunque quanto mostrato dai grafici.

Con i valori dei parametri del secondo esperimento è possibile eseguire una stima più accurata del numero di camion impiegati dalla città di Amsterdam. Dal primo grafico si evince quindi che il numero **medio di veicoli** utilizzati è pari a **29**, mentre il valore **mediano** si attesta intorno a **34**.

5 - Conclusioni

Utilizzando la piattaforma **Snap4City** è stato possibile raccogliere, processare e visualizzare grandi quantità di dati relativi alla raccolta dei rifiuti della città di Amsterdam.

Attraverso gli strumenti e le librerie **Python** sono state eseguite delle analisi statistiche per comprendere meglio l'andamento e le caratteristiche della produzione dell'immondizia nel 2020 e nel 2021.

Infine è stato sviluppato un algoritmo per identificare i percorsi della raccolta dei rifiuti e stimare il numero di camion impiegati durante le procedure.

6 - Sviluppi Futuri

Gli sviluppi futuri su questo progetto possono riguardare due principali aree:

- Implementare una versione aggiornata dell'algoritmo per il raggruppamento delle misurazioni per identificare possibili tragitti svolti dai camion durante la raccolta dei rifiuti.
In particolare sarebbe interessante utilizzare tecniche avanzate di clusterizzazione di traiettorie, che tengano in considerazione comunque i vincoli spaziali, temporali e di capacità del veicolo.
- Utilizzare tecniche di *Machine Learning* per predire i percorsi di raccolta dei rifiuti più efficienti per ottimizzare la quantità di camion utilizzati e i tragitti svolti.

7 - Bibliografia

- [1] City of Amsterdam, "Household Waste" <https://www.amsterdam.nl/en/waste-recycling/household-waste/>.
- [2] Amsterdam API Docs <https://api.data.amsterdam.nl/v1/docs/datasets/huishoudelijkafval.html#>.
- [3] Snap4City <https://www.snap4city.org>.
- [4] Node-RED <https://nodered.org>.
- [5] NumPy <https://numpy.org>.
- [6] Pandas <https://pandas.pydata.org>.
- [7] Matplotlib <https://matplotlib.org>.
- [8] Plotly <https://plotly.com>.
- [9] Folium <https://python-visualization.github.io/folium/>.
- [10] GeoPy <https://geopy.readthedocs.io/en/stable/>.
- [11] How Landfills Work <https://scdhec.gov/environment/land-and-waste-landfills/how-landfills-work>.
- [12] 4 Major Types of Garbage Trucks <https://routereadytrucks.com/blogs/know-4-major-types-garbage-trucks/>.
- [13] How Much Weight Can a Garbage Truck Lift <https://www.quora.com/How-much-weight-can-a-garbage-truck-lift-in-Australia>.
- [14] DAF CF FUN Refuse Collection <https://www.daf.co.uk/en-gb/applications/municipal/garbage-disposal-trucks/daf-cf-fan-6x2-rear-loader>.