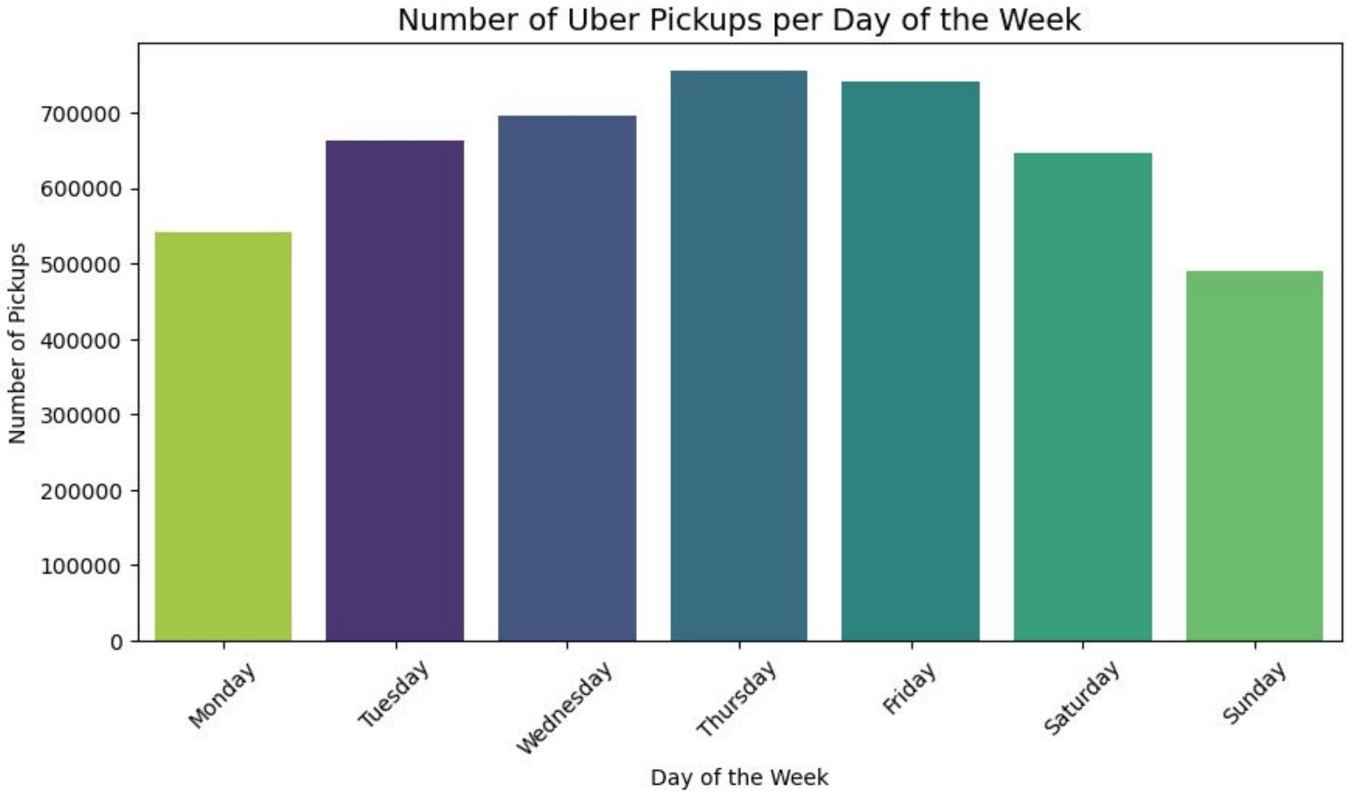
Uber

Where are the hot-zones drivers should be in (NYC)?

Based on 2014 data.

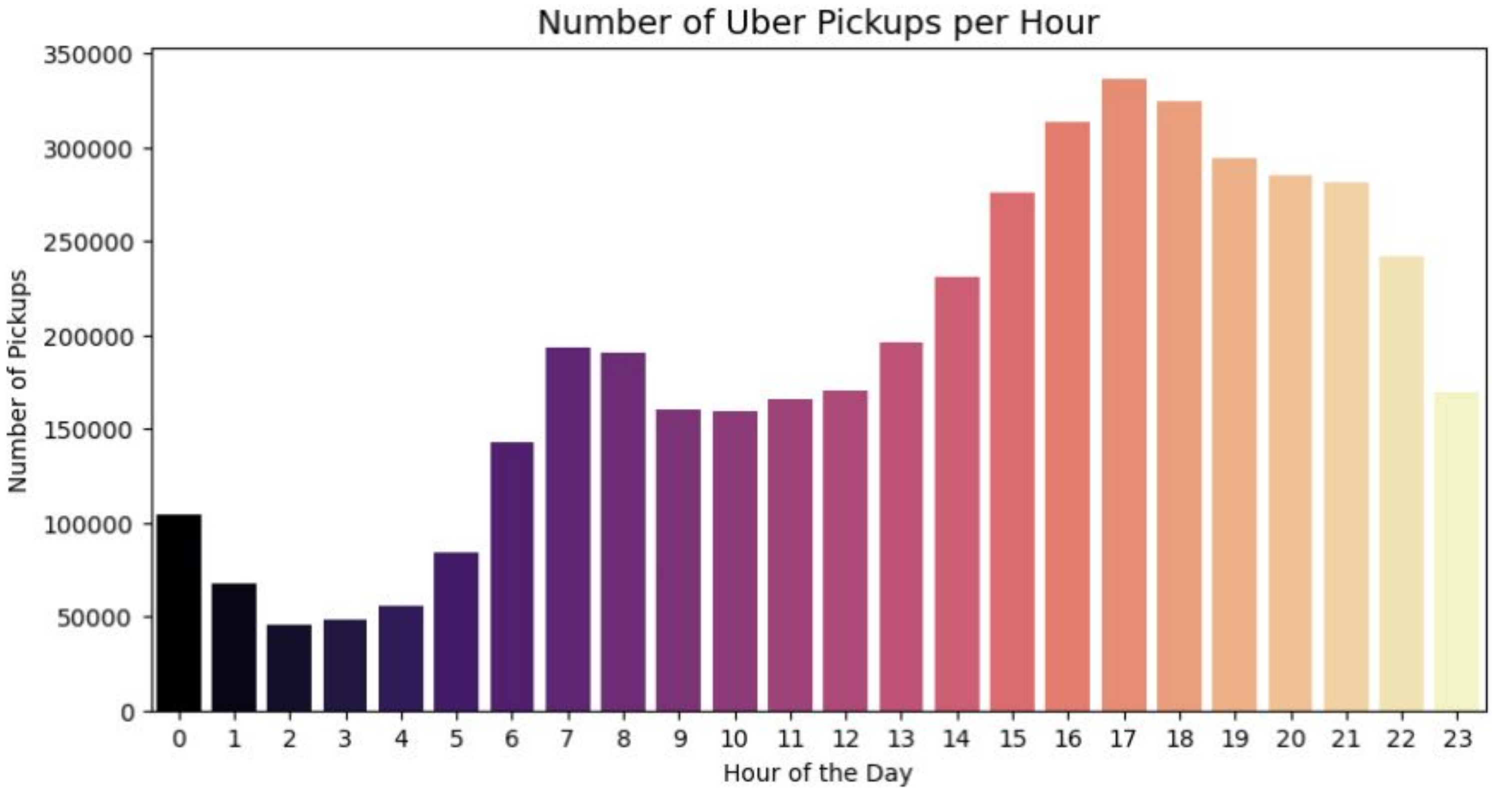
(Data from 2015 has changed and the absence of exact location with latitude and longitude doesn’t allow us to exploit the data with the 2014’s data.)

Pick-ups timeframes analysis

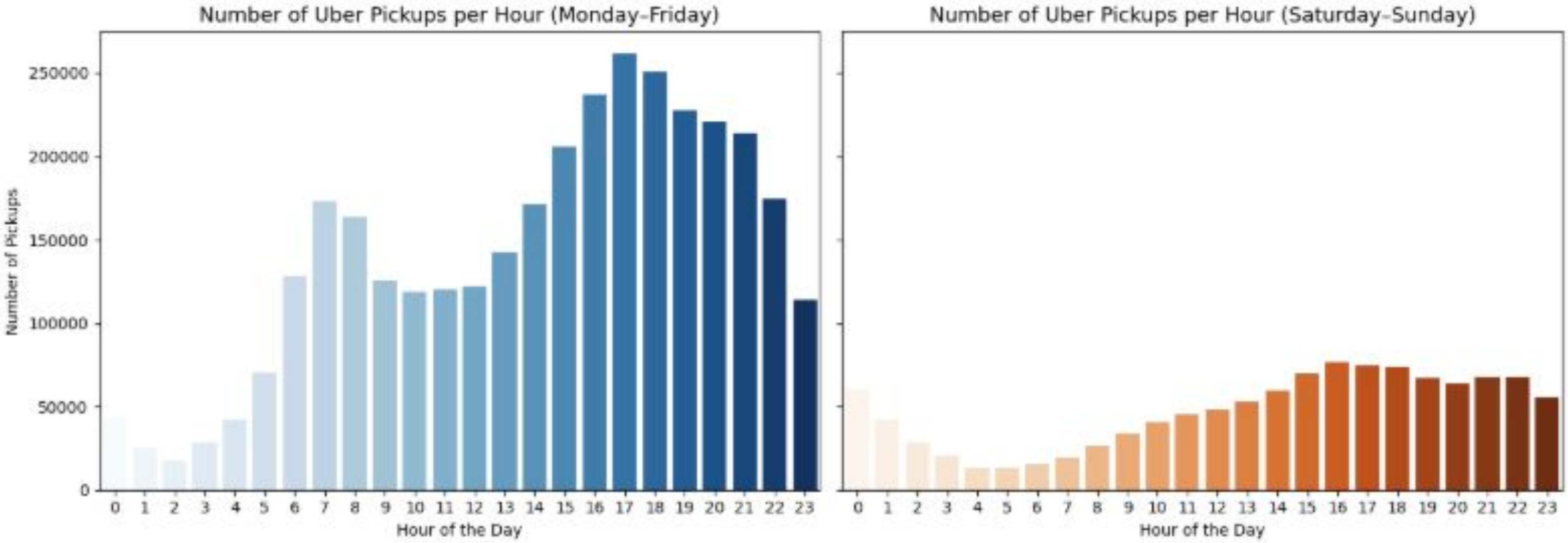


Uber

Pick-ups timeframes analysis (from Monday to Sunday)

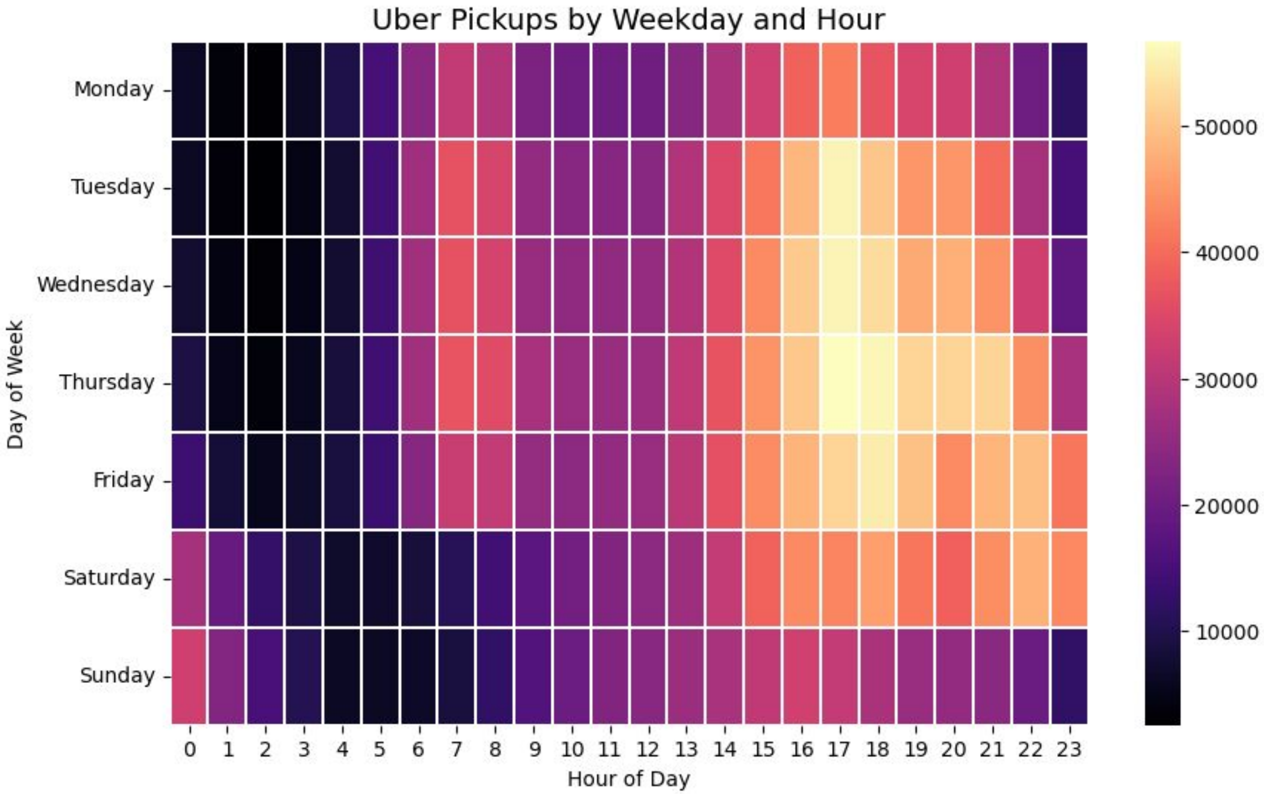


Pick-ups timeframes analysis

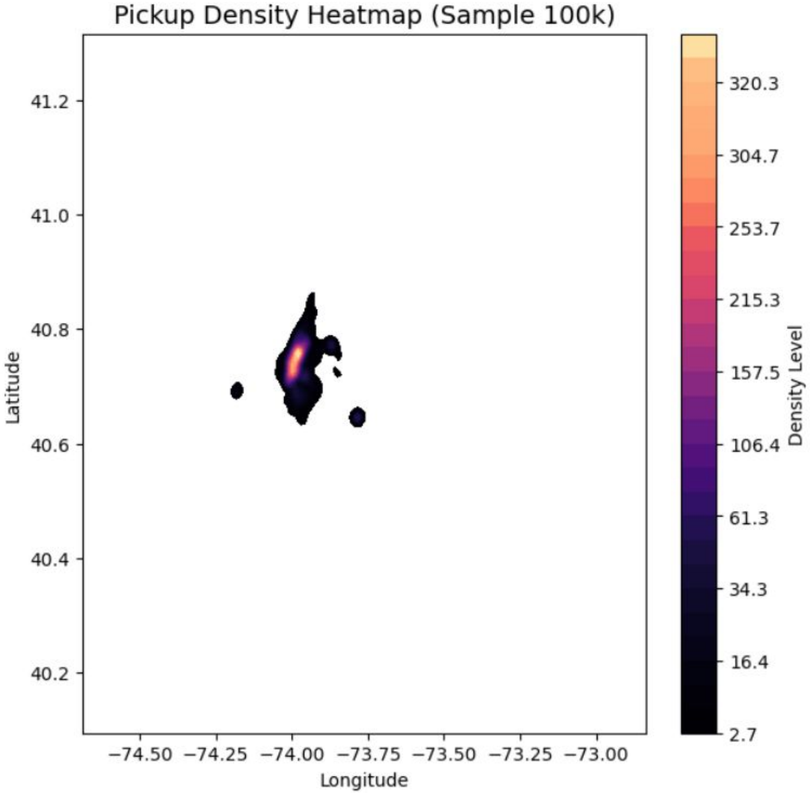


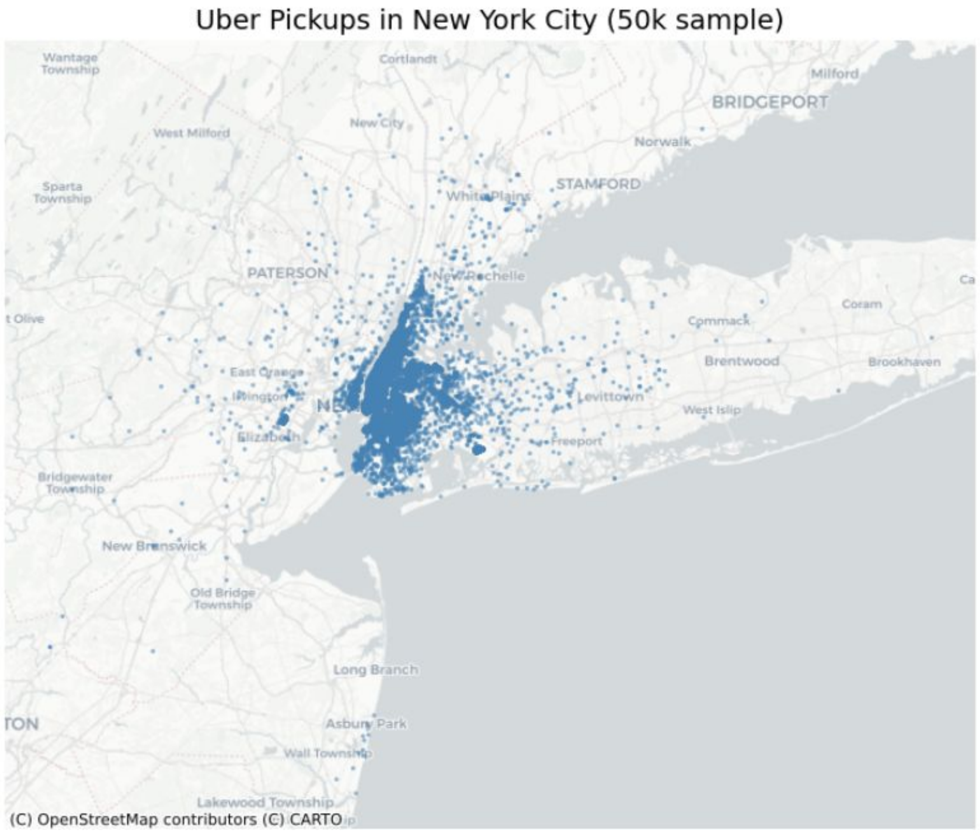
Uber

Uber pick-ups by weekday and hour

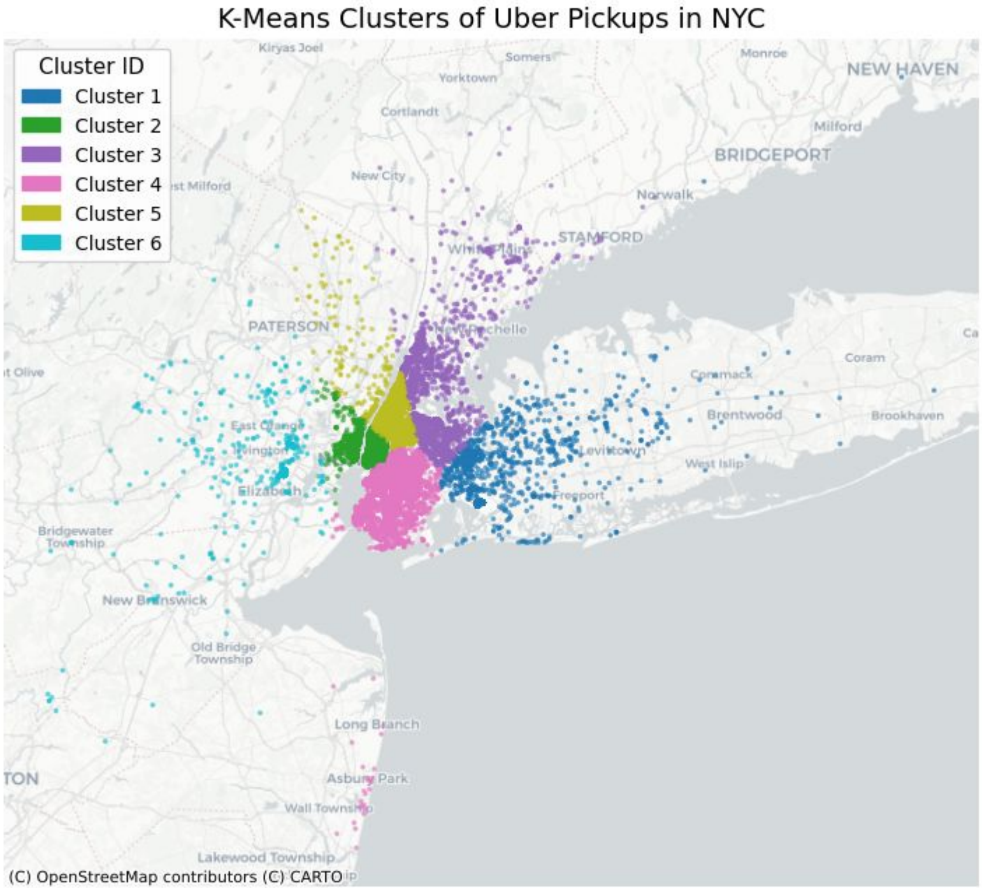


Pick-ups Overview





Pick-ups clusters (K-Means)



Pick-ups clusters (DBSCAN)

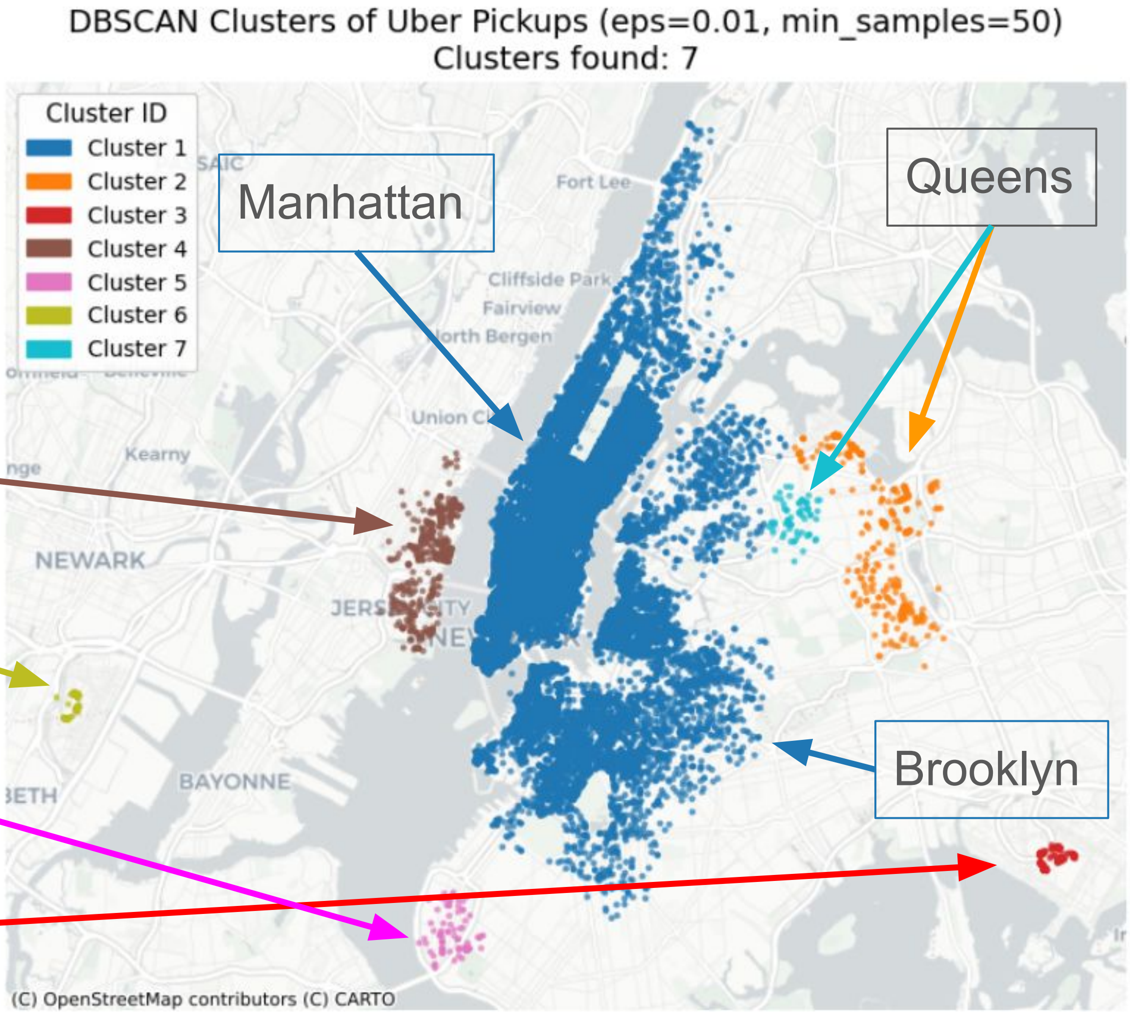
from Monday to Sunday

Jersey City

Newark Liberty International Airport

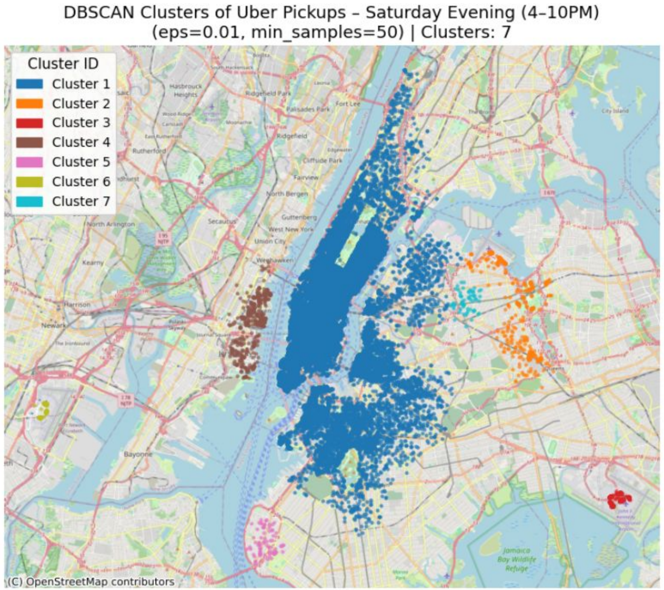
Bay Ridge - Brooklyn

John F Kennedy International Airport

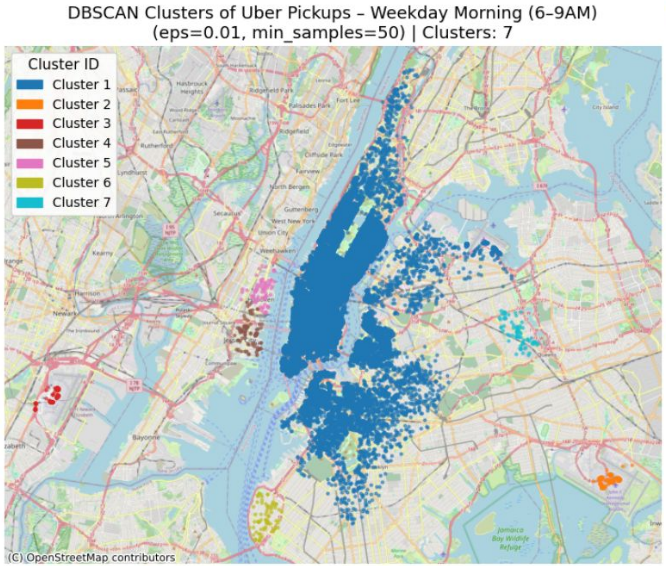


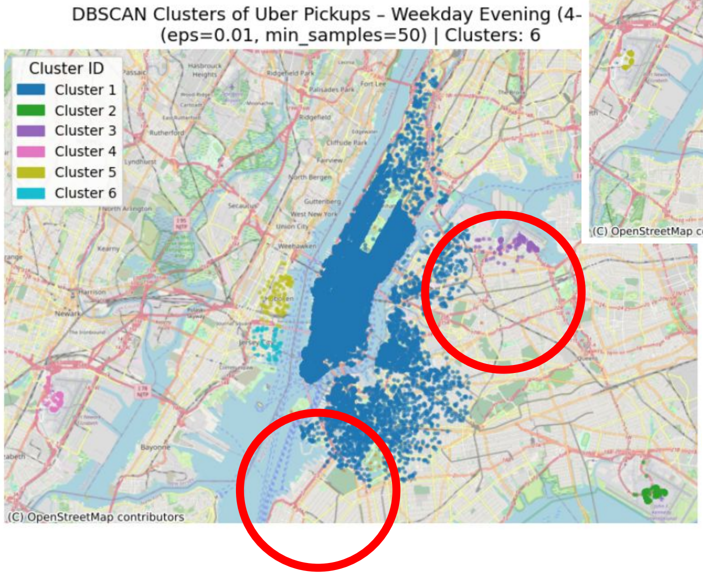
Pick-ups cluster

Saturdays evening



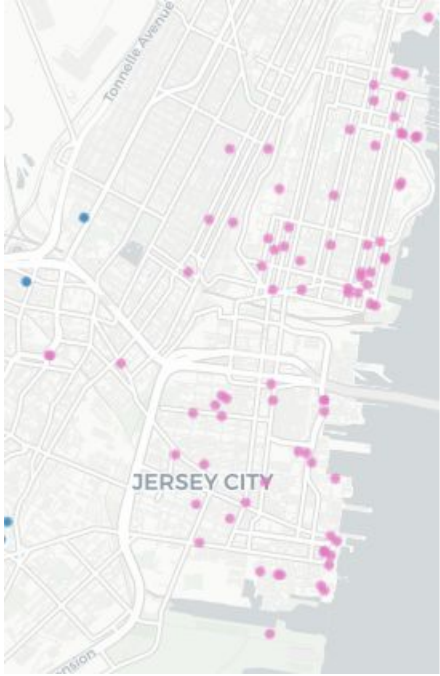
Weekdays morning



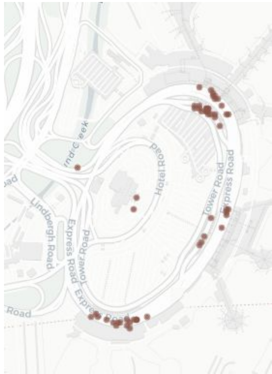


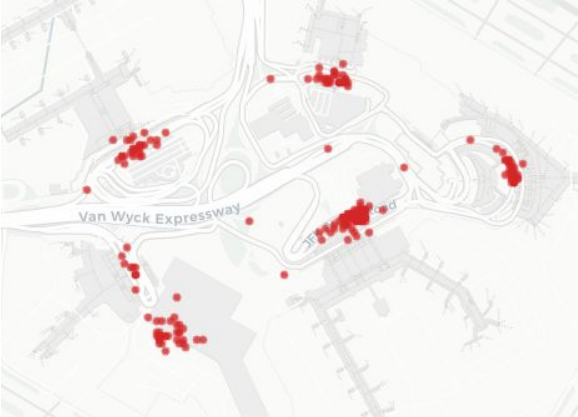
Weekdays evening

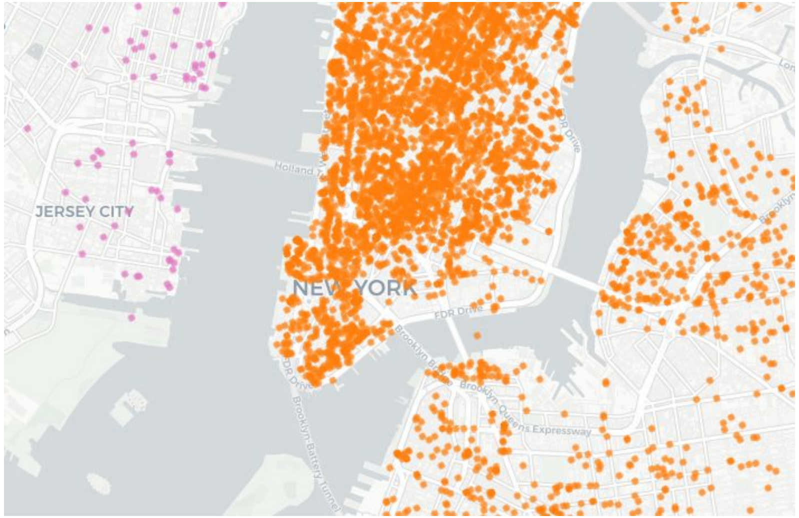
Folium interactive map available



You can zoom-in the clusters

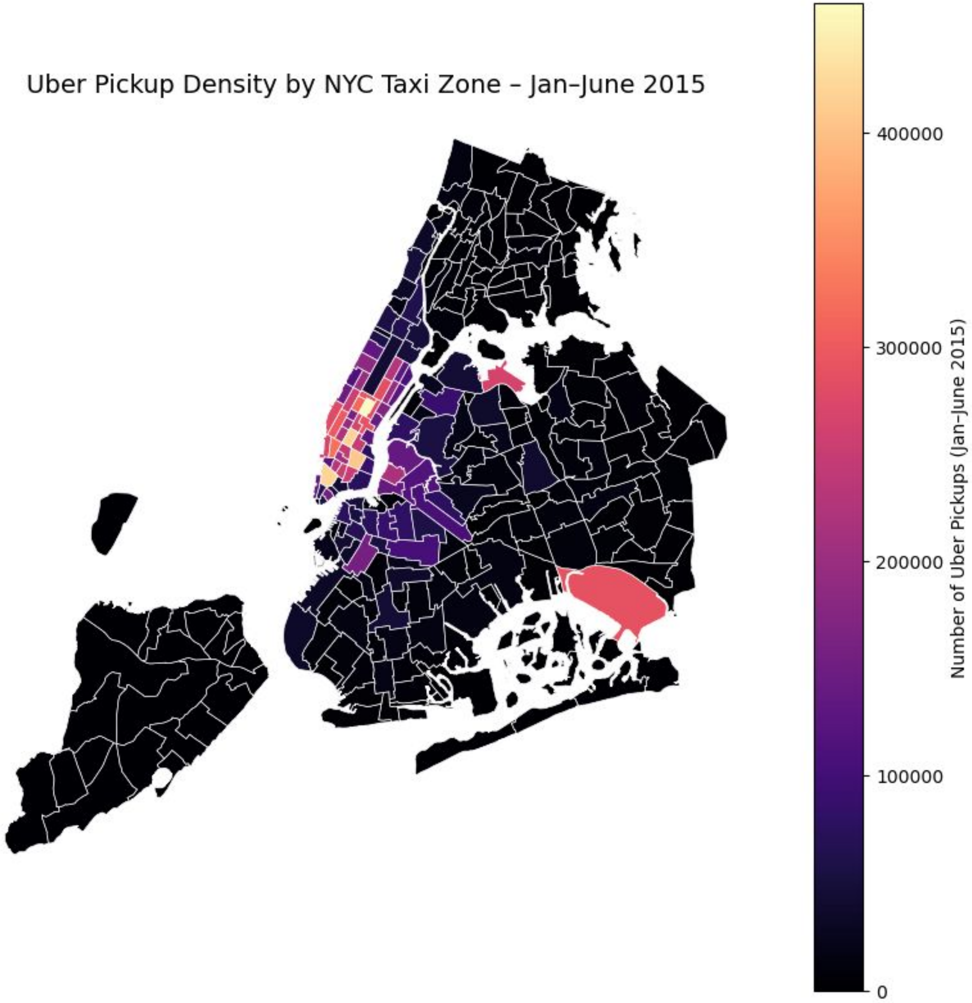






Map animation: evolution of Uber pickup Clusters per hour

Uber pick-ups
Overview area 2015



Uber

Thank you for your attention – any questions?

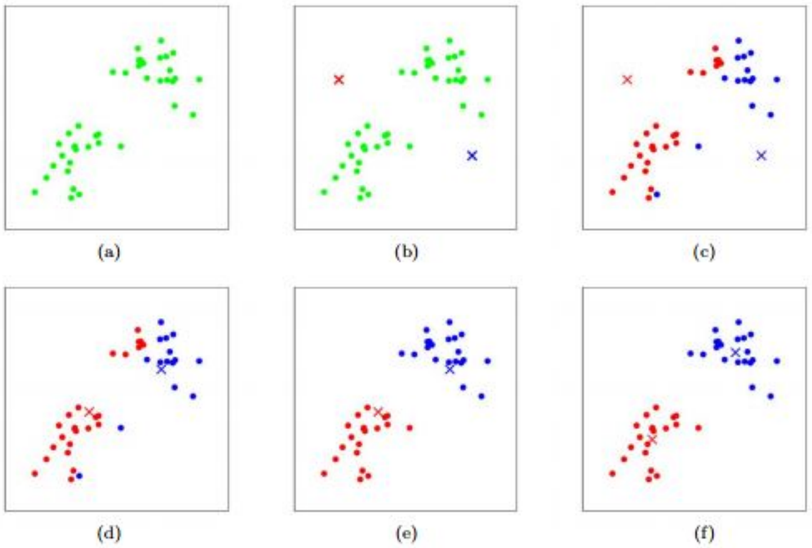
Jedha

K-Means – Centroid-based Clustering

Concept: K-Means divides the data into *k* clusters by minimizing the distance between points and their cluster center (centroid).

How it works:

a) Choose a number of clusters (*k*).
b) Randomly place *k* centroids.
c) Assign each point to the nearest centroid.
d) Move centroids to the mean of their assigned points.
e) Repeat until stable.



Pros:

- Fast and simple to implement.
- Works well for compact, spherical clusters.

Cons:

- You must choose *k* beforehand.
- Sensitive to outliers and non-linear shapes.

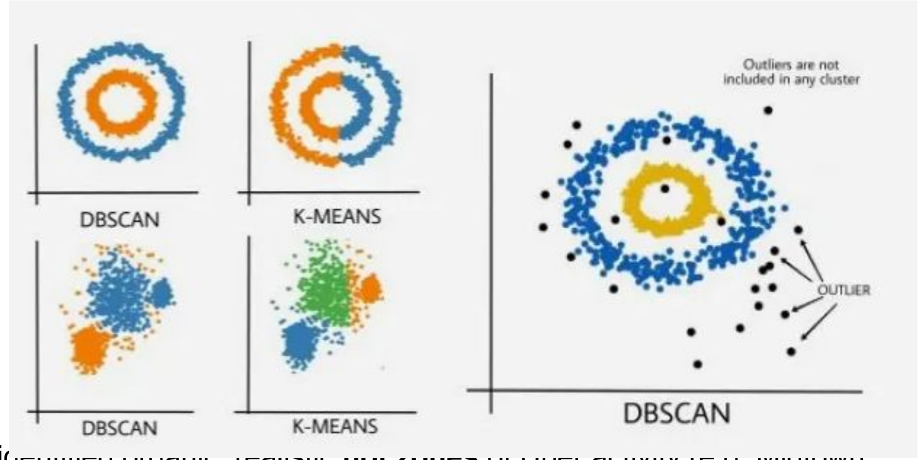
In our project: Used for a **first, simple segmentation** of Uber pickup locations — it revealed broad areas of high activity in NYC (e.g. Manhattan, airports).

DBSCAN – Density-based Clustering

Concept: DBSCAN (Density-Based Spatial Clustering of Applications with Noise) groups points that are **close together in dense regions**, and labels isolated points as *noise*.

How it works:

- A point belongs to a cluster if it has at least `min_samples` neighbors within a distance `eps`.
- Clusters grow from these dense “core points”.
- Sparse areas remain unclustered.



Pros:

- Automatically finds the number of clusters.
- Handles irregular shapes and noise well.

Cons:

- Sensitive to parameter choice (`eps`, `min_samples`).
- Struggles with varying densities.

In our project: Used for **geographical pattern detection** — it identified organic, realistic **hot zones** of Uber activity (e.g. Midtown, Downtown, JFK, LaGuardia).