Analysis of Uber Pickup Data in New York City

Introduction

This analysis aims to harness Uber pickup data from New York City to identify high-demand zones for ride pickups. By understanding where and when demand is highest, Uber can strategically position vehicles, reducing wait times and enhancing service efficiency. This project uses data science methodologies to analyze pickup patterns and offer insights into optimal vehicle deployment across the city.

Problem Statement

The challenge addressed by this project is the inefficient distribution of Uber vehicles across New York City, leading to potential increases in customer wait times and decreases in service satisfaction. By identifying high-demand areas, this analysis seeks to provide a data-driven foundation for improving ride-sharing logistics and overall customer experience.

Objectives

Primary Objective: To identify distinct high-demand zones for Uber pickups within New York City using clustering algorithms.

Secondary Objectives: To analyze temporal trends to understand how demand varies over different times of the day, days of the week, and months and provide actionable insights that could help Uber enhance its operational strategies for vehicle deployment.

Methodology

Data Collection and Handling

- Sources: Uber pickup data from April to September 2014, including date/time, latitude, longitude, and dispatching base code.(Dataset)
- Preprocessing: The data underwent cleaning steps such as handling missing values, removing duplicates, and standardizing formats. It was then aggregated for comprehensive analysis.

Clustering Technique

- Algorithms Used: K-Means and DBSCAN were applied to detect clusters based on spatial data points. The optimal number of clusters was determined using the Elbow Method and silhouette scores.
- Tools and Libraries: The analysis was performed using Python, with libraries including pandas, NumPy, seaborn, matplotlib, plotly for visualization, and scikit-learn for machine learning tasks.

Code Explanation

The project's codebase is structured to carry out data manipulation, analysis, and visualization systematically:

- Data Loading and Cleaning: Scripts to load data from CSV files, followed by data cleaning processes.

```
[ ]  # data handling
     import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt
     import matplotlib.image as mpimg
     import plotly.express as px
     import datetime

     # machine learning
     from sklearn.preprocessing import StandardScaler
     from sklearn.cluster import KMeans
     from sklearn.cluster import DBSCAN
     from sklearn.metrics import silhouette_score
     from scipy.spatial.distance import cdist

     import warnings
     warnings.filterwarnings("ignore")
```

```
taxi = pd.read_csv("taxi-zone-lookup.csv")
april_14 = pd.read_csv("uber-raw-data-apr14.csv")
may_14 = pd.read_csv("uber-raw-data-may14.csv")
june_14 = pd.read_csv("uber-raw-data-jun14.csv")
july_14 = pd.read_csv("uber-raw-data-jul14.csv")
august_14 = pd.read_csv("uber-raw-data-aug14.csv")
sept_14 = pd.read_csv("uber-raw-data-sep14.csv")
```

- Feature Engineering: Date and time were extracted and converted into separate columns to facilitate temporal analysis.

```
[ ] # create individual columns for date and time for exploratory analysis
    df["Day"] = df["Date/Time"].dt.day
    df["Month"] = df["Date/Time"].dt.month
    df["Year"] = df["Date/Time"].dt.year
    df["Time"] = df["Date/Time"].dt.hour

    # drop Date/Time column
    df.drop(columns="Date/Time", inplace=True)

    df.head()
```

	Lat	Lon	Base	Day	Month	Year	Time
0	40.7690	-73.9549	B02512	1	4	2014	0
1	40.7267	-74.0345	B02512	1	4	2014	0
2	40.7316	-73.9873	B02512	1	4	2014	0
3	40.7588	-73.9776	B02512	1	4	2014	0
4	40.7594	-73.9722	B02512	1	4	2014	0

- Normalization and Clustering: Features like latitude and longitude were normalized, and clustering algorithms were applied to this normalized data.

```
# normalize X
scaler = StandardScaler()
X_norm = scaler.fit_transform(X)

# visualize random sample
X_norm[48]
```

```
# create cluster centers, or the average of each cluster
cluster_centers = scaler.inverse_transform(kmeans.cluster_centers_)
display(cluster_centers)
```

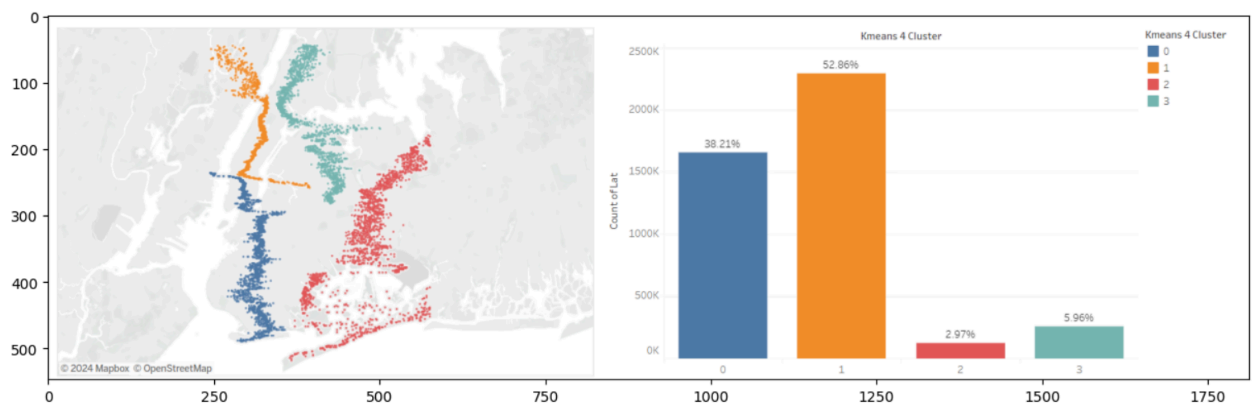
```python
[ ] # try different number of clusters, from 1 to 10
    # mean distortions = average euclidean distance

    clusters = range(1, 10)
    mean_distortions = []

    # make loop to find ideal number of clusters
    for k in clusters:
        model = KMeans(n_clusters=k)
        model.fit(X_norm)
        # assign clusters
        pred = model.predict(X_norm)
        mean_distortions.append(
            sum(
                np.min(cdist(X, model.cluster_centers_, "euclidean"), axis=1)
            )
            / pd.DataFrame(X).shape[0]
        )
```

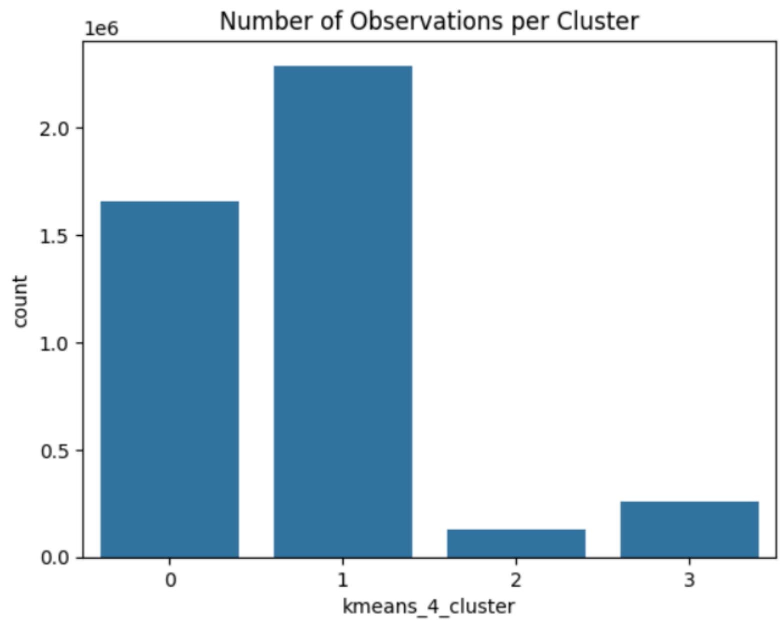
```python
# average euclidean distance from centroid
plt.plot(clusters, mean_distortions, "bx-")
plt.xlabel("K")
plt.ylabel("Average Distortion")
plt.title("Find K with Elbow Method")
plt.show()
```

- Visualization: Code to generate plots like heatmaps and scatter plots to visually depict data clusters and their geographic distribution.

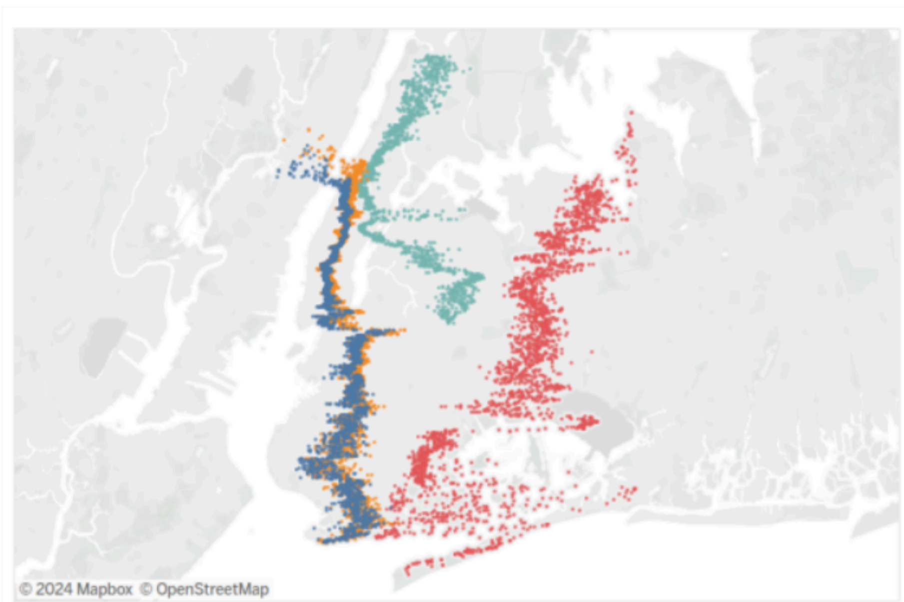


Results

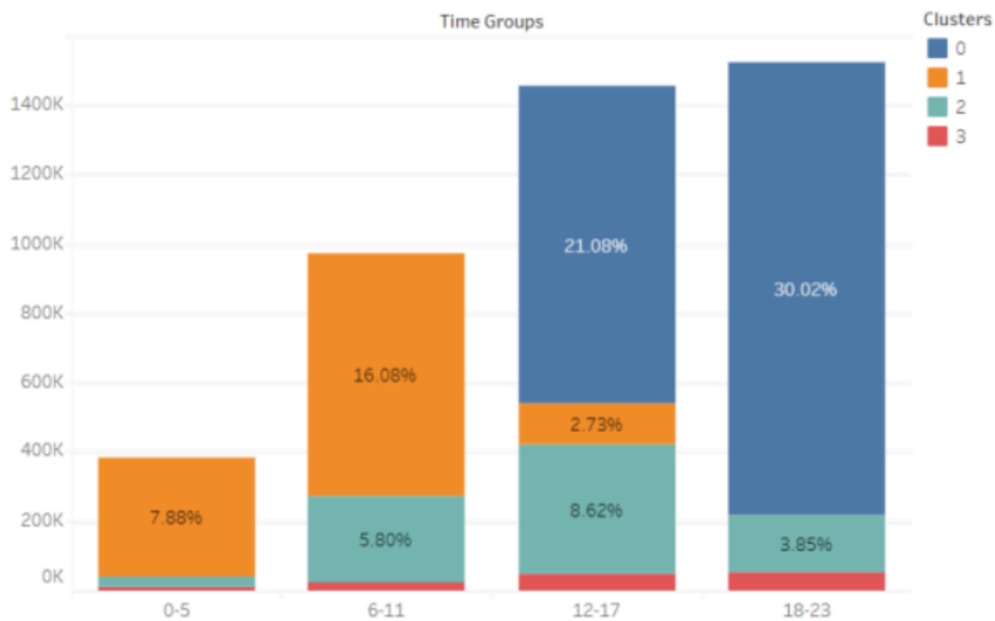
- Cluster Identification: Four main clusters were identified, with each cluster representing a geographically and demographically distinct pattern of demand.



- Geographic Insights: Maps showing these clusters highlighted the areas with the highest demand, particularly in midtown and downtown Manhattan.



- Temporal Patterns: Analysis of demand across different times showed significant peaks during rush hours and weekends, indicating variable demand patterns that can influence deployment strategies.



Limitations

- Data Coverage: The dataset covers only a few months of a single year, which may not fully capture annual variability in demand.
- Algorithm Sensitivity: The sensitivity of clustering algorithms to outliers and the choice of parameters can significantly affect the outcomes, necessitating careful tuning and validation.

Inferences

The insights derived from this analysis can significantly impact Uber's operational strategies by:

-Optimizing Vehicle Allocation: Directing more vehicles to high-demand zones during peak times.
- Strategic Planning: Assisting in long-term strategic decisions regarding fleet management and customer relationship enhancements.

Conclusion

This project illustrates the potential of data-driven approaches in optimizing ride-sharing operations. The findings provide a foundation for Uber to refine its service delivery in New York City, ensuring better availability and quicker service times. Further research could integrate additional variables such as weather conditions or special events to enhance predictive accuracy and operational planning.