NYC Transit Ridership Pattern Prediction and Classification

Sanchit Sahay, Karthik Krapa, Kumuda Aggarwal
New York University
{ss19723, kk5754, ka3535}@nyu.edu

***Abstract*—The efficient management of urban transit systems requires accurate forecasting of ridership patterns and associated metrics. This project focuses on analyzing the Metropolitan Transit Authority (MTA) dataset to predict subway ridership, delays and fare categories. Utilizing advanced machine learning techniques like XGBoost for regression and classification tasks, we preprocess and analyze the dataset to develop high-performing models. This work aims to provide actionable insights to transit authorities for effective scheduling, maintenance, and commuter service enhancements.**

I. Introduction

Urban transit systems are vital for millions of commuters, especially in cities like New York, where the subway faces challenges such as congestion that impact service quality and urban mobility. This study leverages Metropolitan Transportation Authority (MTA) datasets to develop predictive models aimed at enhancing track maintenance, scheduling, and resource allocation.

Using advanced machine learning techniques, the research focuses on a high-performance framework to forecast ridership patterns, addressing complex, non-linear relationships in the data. Additionally, clustering methods identify usage patterns across stations and time periods, enabling targeted service improvements.

Beyond theoretical contributions, this work demonstrates practical applications of data-driven insights to enhance transit system efficiency, reduce commuter stress, and support sustainable urban growth. By addressing these challenges, the study aims to promote smarter, more responsive transit systems for modern cities.

II. Literature Review

Several studies have explored methods for analyzing and predicting subway ridership patterns. We draw inspiration from the following key works:

Ding et al. (2016) [1] applied Gradient Boosting Decision Trees(GBDT) to predict short subway ridership for select stations in Beijing. This model not only achieved high accuracy in ridership forecasting but also provided insights into the relative importance of various factors influencing subway usage. The authors found that the immediate previous ridership was the most significant predictor, while the impact of bus transfer activities varied across stations.

Paparrizos and Gravano (2015) [3] introduced k-Shape, a novel algorithm for shape based time-series clustering. It utilizes a scalable iterative refinement procedure to create homogenous and well-separated clusters. The algorithm employs a normalized version of cross-correlation to consider the shapes of time-series while comparing them. k-Shape demonstrated superior performance in terms of accuracy and efficiency compared to other state-of-the-art clustering approaches.

Yujin Park et al. (2022) [4] focused on clustering weather stations based on their rainfall patterns. While their end goal was weather-related, their methodology provided valuable insights into clustering stations based on time-series data. This approach can be adapted to cluster subway stations based on their ridership patterns, offering a foundation for understanding similarities and differences in usage across a transit network.

These studies collectively inform our approach to analyzing subway ridership patterns, combining advanced predictive modeling techniques with innovative clustering methods to gain deeper insights into urban transit dynamics.

A. Data Sources

The source of the data for these models is the MTA Subway Hourly Ridership data that provides us with the fields below starting July 2020 to December 2024 -

transit_timestamp - Timestamp payment took place in local time. All transactions here are rounded down to the nearest hour. For example, a swipe that took place at 1:37pm will be reported as having taken place at 1pm.

transit_mode - Distinguishes between the subway, Staten Island Railway, and the Roosevelt Island Tram

station_complex - The subway complex where an entry swipe or tap took place. Large subway complexes, such as Times Square and Fulton Center, may contain multiple subway lines. The subway complex name includes the routes that stop at the complex in parenthesis, such as Zerega Av (6)

borough - Represents one of the boroughs of New York City serviced by the subway system (Bronx, Brooklyn, Manhattan, Queens)

payment_method - Specifies whether the payment method used to enter was from OMNY or MetroCard.

fare_class_category - The class of fare payment used for the trip. The consolidated categories are: • MetroCard – Fair Fare; • MetroCard – Full Fare; • MetroCard – Other; • MetroCard – Senior & Disability; • MetroCard – Students; • MetroCard – Unlimited 30-Day; • MetroCard – Unlimited

7-Day; • OMNY – Full Fare; • OMNY – Other; • OMNY – Seniors & Disabilities

ridership - Total number of riders that entered a subway complex via OMNY or MetroCard at the specific hour and for that specific fare type.

transfers - Number of individuals who entered a subway complex via a free bus-to-subway, or free out-of-network transfer. This represents a subset of total ridership, meaning that these transfers are already included in the preceding ridership column. Transfers that take place within a subway complex (e.g., individuals transferring from the 2 to the 4 train within Atlantic Avenue) are not captured here.

latitude - Latitude for the specified subway complex

longitude - Longitude for the specified subway complex

Based on the individual models we use, we pick what features will be the most pertinent, the exact preprocessing steps that we use for each model is described in the subsections below.

Before we do this, we plot ridership numbers on a graph to look for any cyclical patterns and growth trends.

From Fig. 1, we can see that ridership through a station depends heavily on the time of the day.

From Fig. 2, we can see the need to undersample our training data from before the year of 2022, since the COVID-19 pandemic caused unnaturally low ridership through the years of 2020-2022. We can also see that post 2022, there's a steady increase in ridership traffic. What we also see is that the range of ridership traffic depends heavily on the station. We would expect our prediction model to map all these features.

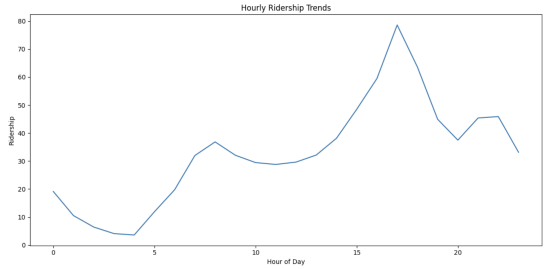

Fig. 1. Hourly Pattern for a Random Station

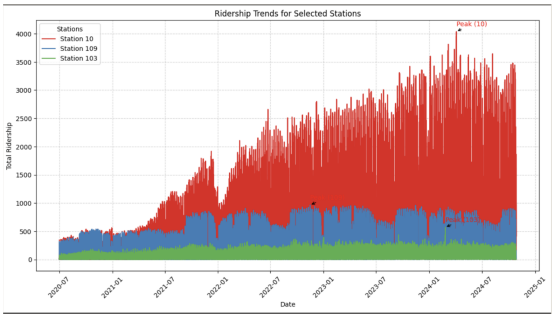

Fig. 2. Overall Ridership Trends

III. Ridership Prediction Models

For ridership prediction, we utilize several Time-Series Forecasting techniques, given that we aim to predict future ridership based on past trends. These techniques are also helpful because the ridership pattern is cyclic on day-of-the-week and hour-of-the-day, and it also shows a steady increase as time progresses. For example, the ridership for the same station in the same month might be similar for the same time of the day and the same day of the week, but it's more likely to be higher than the last year's ridership all else being equal.

Given that the data is available to use in a tabular format without any scope for missing data, the best approach to train a model on this was to select a tree-based model [2]. We go into the details of two such ensemble models - RandomForest Regressor and XGBoost Regressor.

To select the hyperparameters for these models, we use BayesSearchCV. This approach uses Bayesian optimization principles to find optimal hyperparameters. This is more efficient than GridSearch as it doesn't need exhaustive searching and RandomSearch since it makes intelligent decisions about which hyperparameters to select next.

A. Data Preprocessing

We convert transit_timestamp into the tuple (hour_of_day, day_of_week, day_of_month, month_of_year, year) in order to get more meaningful features out of the original dataset.

We use OneHotEncoding to encode station_complex_id, and fare_class_category. Even though station_complex_ids are numerical, they need to be encoded rather in order to avoid adding any unwanted hierarchies into these stations.

For ridership prediction based on individual stations, we can drop strictly correlated values such as borough, location, and the lines that the station operates. These are unlikely to add any meaningful information to our models.

Finally, we scale all other values so that features such as year don't end up having an inherently higher weight than the hour of the day.

Since we are interested in a time series prediction, our training data is set to the rows pertaining to the years 2022 and 2023, while the testing data is from the year 2024. A purposeful decision was made to exclude data from the years of 2020 and 2021 since the ridership during this time was abnormally low due to the COVID-19 pandemic (refer to Fig. 2), but it was pertinent to include multiple years in our training data so that the model can identify increasing trends based on the year.

B. Random Forest Regressor

As a baseline for our tree-based models, we ran a random forest regressor. This helped us gain insight into whether or not the tabular data we have is likely to perform well for our predictions. To get optimal hyperparameters for this, we used BayesSearchCV.

C. XGBoost Regressor

As our final model for predicting ridership, we chose to utilize XGBoost (Extreme Gradient Descent-Boosted Trees) because it is known to perform well on tabular data, time-series forecasting, and in our literature review XGBoost turned out to be the state-of-the-art for several other studies on ridership predictions in local trains and temporal data. [1], [2]

XGBoost works on the following principles -

- It Combines multiple weak decision trees.
- It uses gradient boosting to sequentially correct errors in previous trees.
- This model works well on tabular data. Often state-of-the-art for time-series forecasting.
- Easy to fine-tune since we can easily see feature importances in the final model. This will be helpful to see which temporal patterns matter and are captured.

IV. Station Clustering Models

The inspiration for clustering stations based on ridership patterns was comparing the hourly ridership trends of several different stations. From the graphs, it's obvious that the peaks for these graphs are different depending on the borough and the lines that they operate. (refer to Fig 3). Clustering these stations will give us an insight into the distribution of cities in terms of major commercial and corporate hotspots, and in helping the city plan out which lines to run more frequently. This approach for clustering stations is also present in this paper [4], where they ran the model for the city of Seoul.

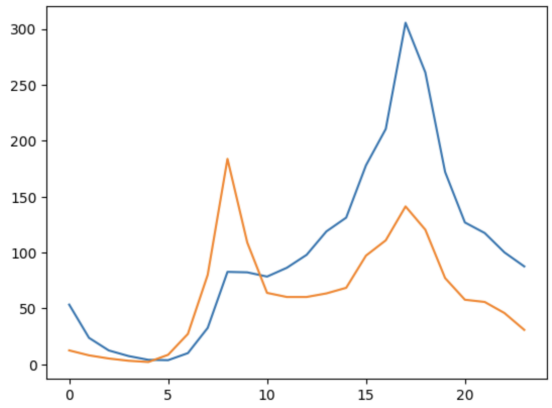

Fig. 3. Ridership Patterns for two stations.

A. Notes on Time-Series Clustering

This problem falls under the category of time-series clustering, which can be performed based on several intended outcomes. The simplest model to use is the TimeSeriesKMeans model which can use several different algorithms to compute distances between two signals. Two such being the Euclidean distance algorithm, which simply finds the euclidean distances between each point in the graph, and Dynamic Time Warping (DTW) which is useful if the frequency of shape changes is less of an issue when matching two signals.

In our case, we found a relatively new approach called K-Shape clustering, which is detailed in this paper [3]. We found that it works better than K-Means using euclidean and DTW algorithms which we will discuss now.

DTW ignores the frequency of the time-series which is a useful metric for us to preserve. For example stations with really sharp peaks and spread out peaks should ideally be clustered differently given that these represent two different ridership patterns. The Euclidean algorithm is prone to ignore shapes, and focus more on the amplitudes of two time-series, which is also not ideal in our scenario, since two stations might mimic each other's ridership patterns perfectly but see very different baseline traffic due to several factors.

K-Shape helps us preserve and ignore the exact features that would be useful to us. It uses normalised cross-correlation as its distance metric, which is a technique which tries to find the distances between two signals by removing the dependence on their amplitudes. K-Shape also doesn't try to fit two signals by stretching out the signals. Some other novel approaches in this method are the centroid calculations for these clusters are done in a manner that preserves the shapes of the series (Fig. 4), it is usually parameter and hyperparameter free – meaning that there is often not a need to preprocess signals before passing them to this model. On top of this, K-Shape is also very scalable. For shape-based distance computations the algorithms runs in O(m.log(n)) time complexity, where m is the size of the time series and n is the number of columns.

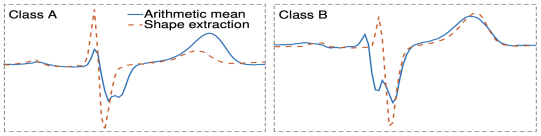

Figure 4: Examples of centroids for each class of the ECG-FiveDays dataset, based on the arithmetic mean property (solid lines) and our shape extraction method (dashed lines).

Fig. 4. K-Shape Centroids

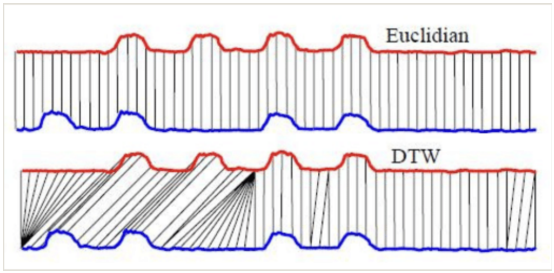

Fig. 5. Classical K-Means Algorithms

B. Data Preprocessing

We only consider the station complex id, hour of the day, and ridership features for this clustering. We group our

dataframe entries by (station complex id, hour of day) and resolve it using the mean of the ridership values.

We then transform this dataframe such that each station complex id contains 24 feature columns which contain the mean ridership value for every hour of the day.

As discussed above, we do not need to scale these values since we'll be using K-Shape.

Our final data frame now has an entry for each of the MTA subway stations, and each row now contains a time series with 24 entries. We can now feed this data into our KShape model.

One drawback of K-Shape is that it is not guaranteed that classical methods for determining the number of clusters such as the elbow method and the silhouette score produce meaningful outputs. We therefore used external knowledge of typical ridership pattern clusters to determine this number. We ended up picking 3 final clusters. The results and analysis for this will be in the following section.

V. Results and Analysis

A. Ridership Prediction

For our experiment, we ran our model through three subway stations in the city – 49 St (N,R,W), Bowery (J,Z), Fresh Pond Rd (M). We picked these stations because they differ in their peaks and ranges for their average ridership (Fig. 6).

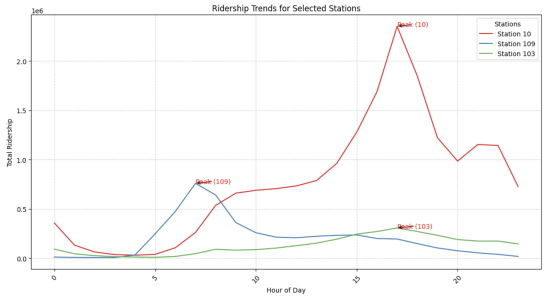

Fig. 6. Different Ridership Patterns.

Random Forest Regressor

Our base RandomForest model with 200 estimators, and max depth of 3 has the following scores for the testing data:

Parameter	Testing Data	Training Data
R2 Score	0.5632	0.5793
MAE Score	32.5646	29.4411
MSE Score	8036.7989	3960.2089

TABLE I
Random Forest Regressor Model Scores

These scores, while not optimal, do give a general sense that tree-based models are somewhat able to meaningfully map the feature set into ridership predictions. The model is also not overfit yet, implying that it is possible to optimise this even further.

XGBoost Regressor

Our final XGBoost model performs better, with the following hyperparameters -

- learning rate=0.2334
- max depth=7
- min child weight=4
- n estimators=200

Parameter	Testing Data	Training Data
R2 Score	0.8808	0.9731
MAE Score	14.1073	7.0519
MSE Score	2191.5553	252.4005

TABLE II
XGBoost Regressor Model Scores

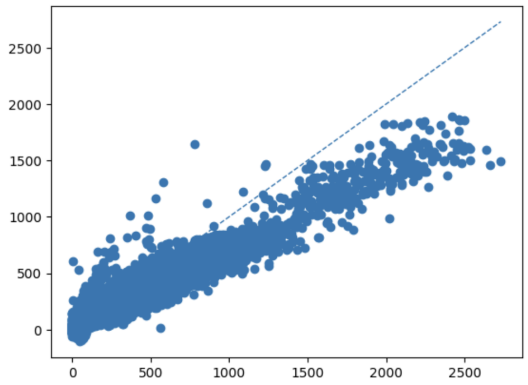

Fig. 7. XGBoost - Ground Truth vs Predicted Values

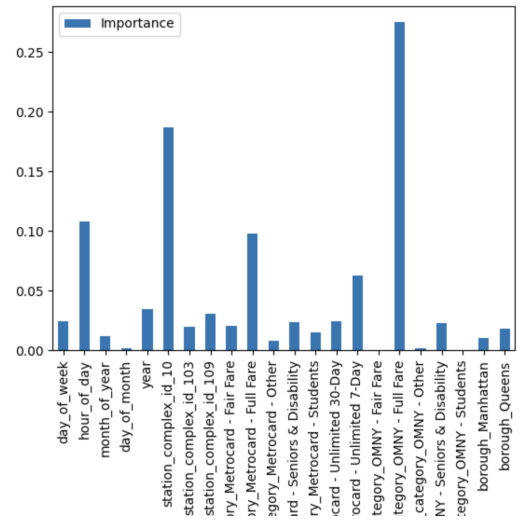

Fig. 8. XGBoost - Feature Importance

Based on Fig. 8, we can state -

- Features pertaining to the station complex id help us properly predict ridership, for example the base-line for station 10 is higher than that of station 103.
- Similarly, at every station, the number of riders paying full fare is higher than any other fare class category.

- Within our timestamp based features, we can see that the hour of the day is the most important feature, followed by the year and day of the week. Other features such as month of the year and the day of the month have very little importance.
- This maps well with our intuition about the dataset – ridership patterns are weekly and hourly, and we can see that our model takes into account year-on-year growth.

Finally, for our XGBoost Ridership Prediction Model, we can conclude the following -

- XGBoostRegressor performs much better than our baseline RandomForest Regressor.
- With the XGBoost model, the training scores are near-perfect, meaning that XGBoost is able to find a reasonable mapping between the feature set and the output. Even with the testing data, we can see that the model performs reasonably well.
- While the numeric difference between the training and testing scores is not insignificant, we cannot claim that the model is overfitting.
- We can claim this based on several reasons, one of them being the ground truth vs predicted values graph, where we can see that the model predicts close to the line of truth for the entire range of outputs.
- Based on the ground truth vs predicted values graph (Fig. 7), we can see that our model tends to underestimate the final ridership values. This is likely because we did not have enough data for the model to more accurately capture the year-on-year growth in subway ridership. Note that the usual split for training vs testing set sizes is 80:20 (for capturing year-on-year growths for time-series forecasting the training set is also usually much higher), whereas we were only able to use a 66:33 split due to lack of prior data and the outlier data points due to the COVID-19 pandemic.
- One way to improve on this ridership output would be to train it on prior data, another would be to introduce population and demographic data of New York to this dataset.

B. Station Clustering on Ridership Patterns

As mentioned before, we used the K-Shape model to cluster stations based on their ridership patterns into 3 clusters. These clusters can be seen in Fig. 9

Further, we can plot these stations on the map of New York and the color them according to the clusters that the model output. We can see this map in Fig. 10

Cluster 1 is represented by red dots, Cluster 2 is represented by blue dots, and Cluster 3 is represented by black dots.

From these outputs, we can conclude the following -

- K-Shape provides us with a logical clustering mechanism for our use case, without having to preprocess our raw time-series data. This can be seen because the clusters have significantly different shapes, and based on our knowledge of New York City's neighbourhoods, the clusters make logical sense.
- Cluster 1 groups together stations that usually have a sharp peak in the morning, and a smaller peak in the evening. Cluster 2 groups together stations that seem to have much less sharper peaks throughout the day. Cluster 3 groups together stations that have a sharper peak towards the end of the day.
- When we see these stations mapped with their clusters on the map of the city, we can see that the further out you go from the center of the city, the more likely it is that the station belongs to the first cluster. This can be explained by the fact that most people have offices that are closer to Manhattan.

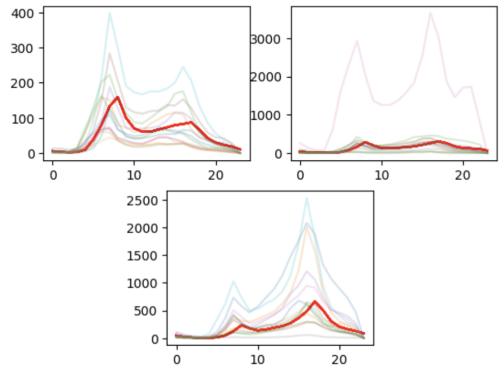

Fig. 9. Final Clusters based on ridership patterns

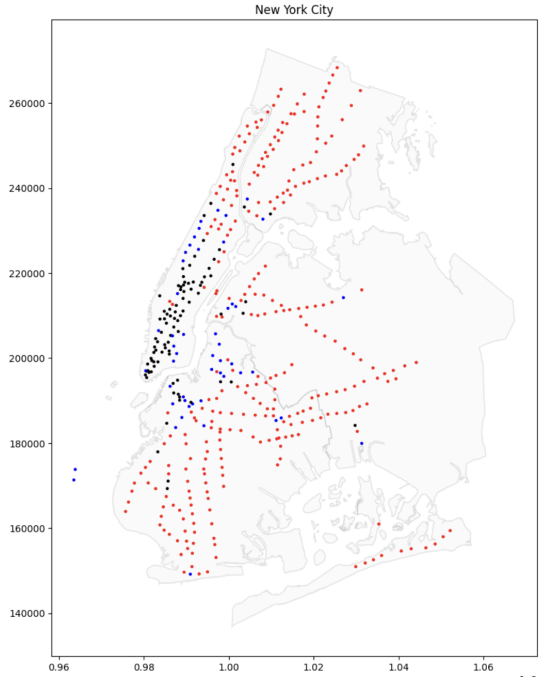

Fig. 10. MTA Stations Mapped with Clusters

- Local commercial hubs such as Downtown Brooklyn are likely to belong to the second cluster. This might be because these stations are less likely to be the source or final destination for most people during the day. These locations also see steady traffic all day.
- Cluster 3, which mostly belongs to popular places in Manhattan, sees a sharper peak at the end of the day, which might be because of its attractiveness for post-office hours activities such as dining out.
- We can claim that clustering based on ridership patterns are meaningful, and that the K-Shape model accurately clusters these stations because despite not having provided the model any geographic locations, our output clusters can be mapped almost directly to their distance from the city center in Manhattan.

VI. Conclusion

This study highlighted how well sophisticated machine learning methods can analyze subway ridership patterns in New York City. Our key findings include:

- By successfully capturing intricate, non-linear correlations in the data, such as weekly cycles, hourly trends, and year over year growth, the XGBoost outperformed the RandomForest baseline.
- The K-Shape algorithm successfully clustered subway stations into three distinct groups based on ridership patterns, identifying trends that matched the topography of New York City, which includes office centers, and popular leisure areas.
- Our approach demonstrates the potential of machine learning for transit planning and management. Meaningful station clustering combined with precise ridership forecasts provides beneficial data for resource allocation and schedule optimization.

While our results are promising, further improvements could be achieved by integrating additional data sources, such as demographic information, to enhance prediction accuracy and clustering nuance. In conclusion, this study illustrated how advanced machine learning techniques can address urban transit challenges, empowering transit authorities to make informed decisions that improve service quality and enhance urban mobility

References

[1] C. Ding, D. Wang, X. Ma, and H. Li, "Predicting Short-Term Subway Ridership and Prioritizing Its Influential Factors Using Gradient Boosting Decision Trees," Sustainability, vol. 8, no. 11, p. 1100, November 2016.

[2] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on tabular data?" arXiv preprint arXiv:2207.08815, 2022. [Online]. Available: https://arxiv.org/abs/2207.08815

[3] J. Paparrizos and L. Gravano, "k-Shape: Efficient and Accurate Clustering of Time Series," in Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15), pp. 1855-1870, May 2015.

[4] Y. Park, Y. Choi, K. Kim, and J. K. Yoo, "Machine learning approach for study on subway passenger flow," Scientific Reports, vol. 12, no. 1, p. 2754, February 2022.