Geospatial Demand Prediction with Explainable AI: Bike sharing Systems Use Case

Ahmed ElSabbagh

September 2024

Abstract

Bike Sharing Systems (BSS) have become very widely used in many cities as an alternative method of transportation, requiring high maintenance by operating companies to rebalance bikes in an optimal manner for easier access and availability by users. Accurate prediction of bike demand at the area level is a crucial first step in the rebalancing process. In this work, feature engineering and generation methods using Graph Neural Networks (GNNs) have been used to capture spatial and temporal dependencies and network activities, which translates into the demand pattern across the entire system. The performance was furthermore refined using insights obtained from SHapley Additive exPlanations (SHAP) used to analyze the feature importance for area-level predictions. Experiments conducted using Baywheels' BSS system data in San Francisco between 2022 and early 2024 have demonstrated a significant improvement in performance across several metrics, and refinement with SHAP-based trimming applied afterwards has demonstrated flexibility in creating partitioned XGBoost models, which collectively reduced the error rate by an overall 16% compared to the initial baseline model.

أنظمة مشاركة الدراجات Bike Sharing Systems (BSS) أصبحت واسعة الانتشار في العديد من المدن كوسيلة مواصلات بديلة، وهي في حاجة دائمة الي إعالة و صيانة مستمرين من شركات الادارة و ذلك حتى تعاد توزيع الدراجات بطريقة أفضل من أجل توفر خدماتٍ أسهل للمستخدمين. إن التنبؤ الدقيق بطلب الدراجات على مستوى المنطقة يعد خطوة أولى حاسمة في عملية إعادة التوزيع. استخدم في هذا العمل طرق لهندسة و توليد الصفات feature engineering and generation باستخدام الشبكات العصبية البيانية Graph Neural network (GNN) لالتقاط التبعيات المكانية والزمانية والأنشطة الشبكة التي تترجم نمط للطلب عبر النظام بأكمله. و علاوة على ذلك، تم تحسين الأداء باستخدام الرؤى التي تم الحصول عليها من خوارزمية SHapley Additive exPlanations (SHAP) لشرح النموذج، و ذلك عن طريق تحليل أهمية الصفات للتنبؤات على مستوى المنطقة. و لقد أظهرت تجارب اجريت على بيانات نظام Baywheels المستخدم في مدينة سان فرانسيسكو ببيانات بين عام ٢٠٢٢ و الشهور الاملى في عام ٢٠٢٤ تحسناً بارزاً في عدة مقاييس للأداء، وكذلك أظرت التحسينات باستخدام تقنية تشذيب الصفات باستخدام SHAP بعد ذلك مرونة في خلق نماذج XGBoost مقسمة، مما ساهم بشكل جماعي في تقليل نسبة الخطأ اجماليا بنسبة 16% مقارنة بنموذج خط الأساس الأولي.

Acknowledgment

In the completion of this thesis and the years of work that preceded it, I would like to thank my parents and siblings whose support has carried me through the journey. I would also like to thank my colleagues and managers for their patience at work throughout my study and the professional experience that inspired it. And finally, I would like to express my deepest appreciation to my professors during my academic journey, including Prof. Mohamed Safi and Prof. Essam Eldean Elfakharany, and all other professors and mentors who have contributed their insights directly or indirectly.

Contents

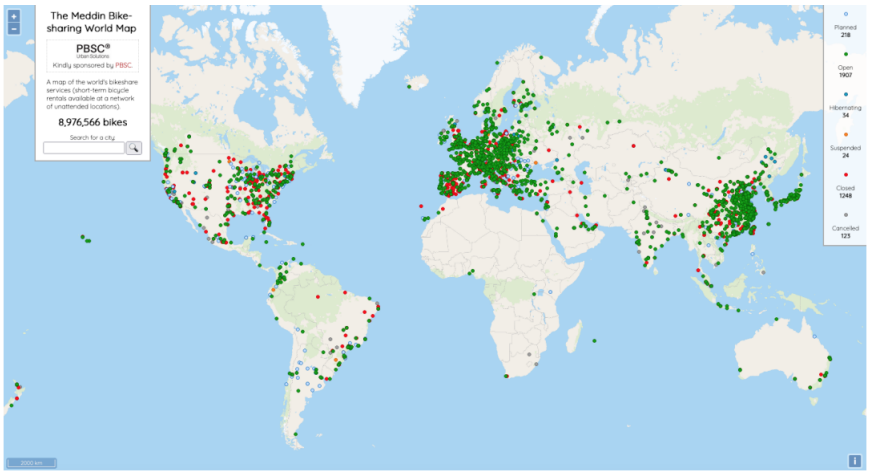


Figure 1: Bike Usage Report World Wide 2022. The largest concentration can be found in Europe and East Asia, with a good number of them present in the United States.

1 Introduction

Geospatial data analysis is the interdisciplinary science concerned with researching methods for handling, creating, and exploring spatio-temporal phenomena related to the Earth[1], including location data (GPS) and its movements. Geospatial analytics is a growing market, with a market size of more than 20 billion USD in 2022 and a predicted expansion to more than 45 billion USD by 2028 [2].

One of the many uses of geospatial analysis is the prediction and mitigation of supply and demand in an area location. This can be used for several applications, such as city traffic predictions, as used by Google Maps, and dynamic ride-sharing pricing, such as Uber[3].

This paper focuses on the application of geospatial demand prediction for Bike Sharing Systems (BSS). BSSs have become a widely used method of transportation for many people around the world, mainly as a cheap, fast, non-polluting option capable of circumventing traffic congestion, especially on last-mile connections, with a lower operational overhead cost compared to other transit systems[4]. BSSs are very ubiquitous in many countries[5] (Figure 1) and are often subsidized by governments to encourage their use, either for their health benefits or to reduce the dependence on personal vehicles[4][6].

The predictions will be interpreted using both explainability techniques designed to explain the direct factors that affect the predictions[7], as well as more traditional post-model explanatory methods to statistically explain the results.

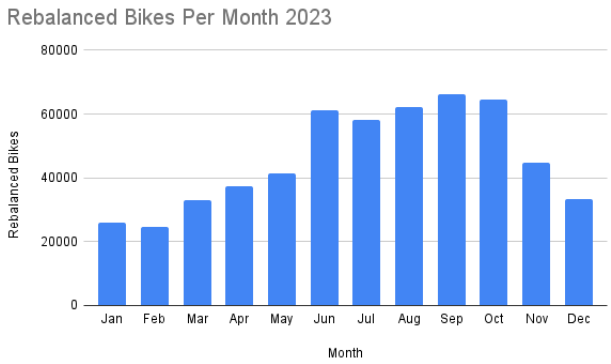


Figure 2: Numbers of monthly rebalanced bikes in CitiBike BSS in New York 2023

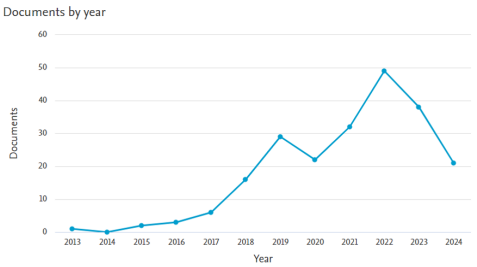


Figure 3: Papers discussing bike sharing demand prediction yearly over 2003-2024 period (219 total papers)

1.1 Motivation

Most modern BSSs use automated self-serve bikes with docking stations. The users of the system are very unlikely to return the bikes to their original stations, instead leaving them at the closest station to their destination[4]. This creates a bike imbalance issue in which stations with high bike demand at a certain time may not have available bikes due to their usage at an earlier time of the day[8]. BSS management companies manage rebalancing runs with vans or trucks, which can be a very costly and polluting operation, with cities like New York moving more than 66,000 bikes in September 2023 alone[9](Figure 2).

Many countries around the world use some form of BSS in several cities with nearly 9 million bikes, most of which are concentrated in Europe, Asia, and several large cities in the United States[5](Figure 1), which has generated a lot of research interest with more than 219 articles on demand prediction alone since 2013 (Figures 3 and 4).

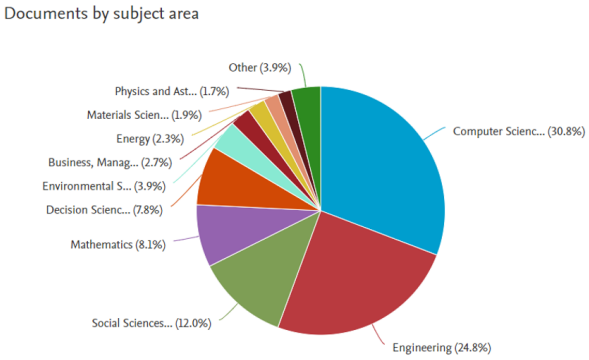


Figure 4: Papers discussing bike sharing demand prediction per subject Areas (219 total papers)

1.2 Problem Statement

The relocation process could be costly and time consuming, especially if there is no clear indication on which docking station has a higher demand[8]. Therefore, predicting demand becomes a very important first step before deciding on rebalancing techniques that would be used in a BSSs. Demand prediction additionally reduces the overhead for rebalancing operations by finding which stations do not require any rebalancing at a given time of day.

Explainability techniques further improves the decision making processes by understanding the reasons behind those predictions and adding more trust to the model as black-box models become more simple and easy to understand. Other measures could therefore be taken to improve the overall operations in addition to rebalancing. However, there seems to be very little literature on the explainability regarding BSSs.

1.3 Objectives

This study aims to create an explainable regression model to estimate demand, defined as the number of bikes requested at each station. Focusing primarily on proper feature processing, generation, and extraction techniques for a timeseries model for area-level predictions with the help of clustering techniques. In addition, applying explainability algorithms on prediction models to improve understanding of prediction results and add grounds for proposing future changes and additions to the model through the trimming of unnecessary features on station level.

2 Background

2.1 Early Bike Sharing Systems

Bike rental businesses need to be differentiated from BSSs, as a BSS is a component that aims to bridge a gap in public transportation networks, while bike rental is meant for leisurely activities[4]

The first generation of BSSs, implemented in Amsterdam in 1965, La Rochelle in 1974, and Cambridge in 1993, did not have security or payment methods, which resulted in system failure as bikes were stolen and damaged[4]. Since then, the security features and payment methods established in the next generations not only allowed a good source of funding with convenient payment methods[6][10], but also allowed easier data collection used for data analysis for business purposes such as predictive analysis, as well as real-time user assistance such as routing[6].

The third generation IT-based system includes Lyft bike services available in several US cities[10], Ttareungyi in Seoul[8], BIXI in Montreal[11]. All of which are available at designated stations compared to fourth-generation dockless BSSs[4][6].

2.2 Demand Prediction Models

Demand prediction is a time series problem that could be solved with either statistical methods or machine learning regression models.

Statistical methods such as Autoregressive Moving Average (ARMA) and Autoregressive Integrated Moving Average (ARIMA)[12] are univariate methods sometimes used as the baseline to compare with, but have fallen out of use in favor of multivariate regression methods such as Random Forest and XGboost[13][14].

Neural Networks, such as Long-Short Term Memory (LSTM) and Convolutional Neural Network (CNN) have become widely used for models[11], more recently, Graph Neural Networks (GNNs) have become more used either for feature extraction to feed in other models[14] or as models themselves with Spatio-Temporal GNN[15], which seems to be very reasonable as GPS data and the relationship between stations can be represented and analyzed with the graph structure and fed to GNNs as input.

2.3 Explainability

Explainability techniques includes built-in methods such as feature importance found in decision trees. However, these methods do not work with all models, which required the usage of purpose-designed largely model-agnostic methods(Figure 6).

Some of the more popular techniques include LIME (Local Interpretable Model-agnostic Explanations)[16], a technique that uses a simpler surrogate model with synthesized data perturbed around the actual data. And SHAP

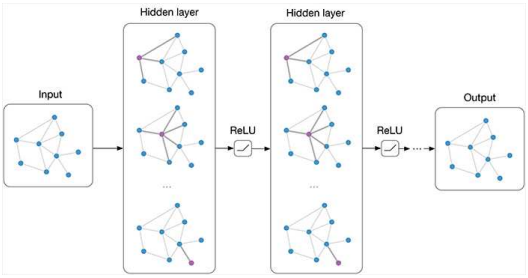


Figure 5: A representation of a Graph Neural Network (GNN)

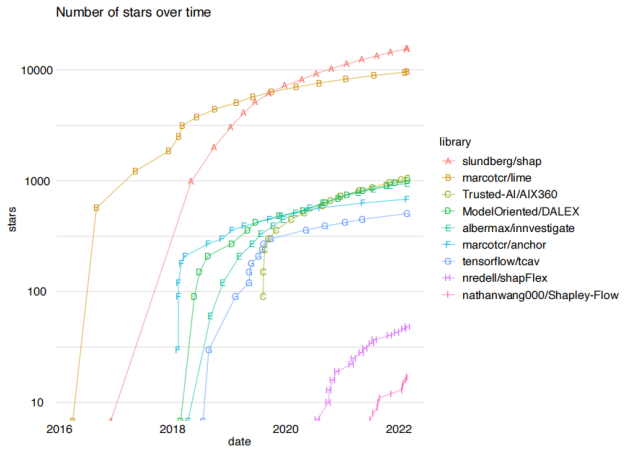


Figure 6: Number of stars on GitHub for the most popular repositories for explainability methods over time as defined by number of stars[7], SHAP implementation by user "slundberg" currently being the most popular method

(SHapely Additive exPlanations)[17], which is based on cooperative game theory to determine the features that were the most effective in a certain prediction.

These two methods are the most commonly used, with a variety of extensions that could cover use cases not covered by the main library, such as GraphLIME being used for GNNs[7]. Although other methods for GNNs exist, such as GNNExplainer[18].

There are many points of comparison between SHAP and LIME and their extensions, but LIME is generally more flexible, being able to evaluate results without relying on the model used at all, since the surrogate model is model-agnostic, its explanations, however, are unstable, as similar points may have big differences between themselves[19]. SHAP's theory, on the other hand, is more solid and the explanations more contrastive with good global interpretations, connecting Shapely values and LIME. However, more computing resources are required to provide the explanations[19]. In addition to technical difficulties in

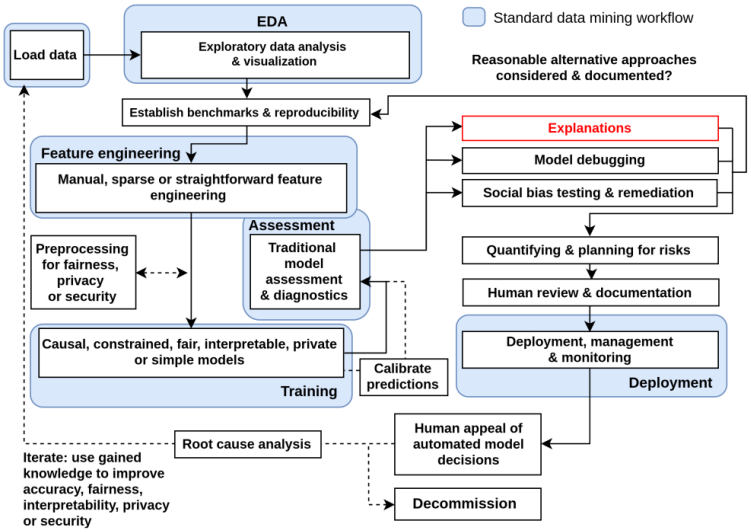


Figure 7: A diagram of a holistic ML workflow in which explanations are used along with interpretable models[20], explaining the feedback and debugging loop applied using explainability techniques

choosing appropriate kernels and defining the complexity of the problem, any explainability technique can be vulnerable to manipulation to introduce human biases[19][20]. Concerns about potential security risks and data privacy vulnerability with explainable AI[20], therefore, while their usage in understanding the ML model can infuse trust and allow and debugging nefarious models in more life-critical applications[20], which are not the main concern of this study.

3 Literature Review

BSS demand predictions have been a topic of interest for many researchers since 2013 and are growing, with 219 articles to date (Figure 3). Research interest dropped slightly during the 2020 COVID-19 pandemic, but has seen a resurgence in trend since 2021, when the pandemic restrictions began to relax, renewing interest in BSSs usage, and thus interest in related research.

Paper	Objectives	Methods and Dataset	Results	Limitations
Torres 2024	[21] Forecasting bike-sharing usage using machine learning regression techniques, with comparison in terms of accuracy and applicability	**Model:** Random Forest, Gradient Boosting, Neural Network **Clustering:** Kmeans **Dataset:** NYC CitiBike dataset Data from 2018 as training data, with period from 15-29 Jan 2019 as test. Takes: Station ID, Date time data, Weather data	Results explained in MAPE: 60% at best 40% at low demand stations 70% during peak periods. Best model is Neural Network Clustering improved accuracy by 10%	**Testing Limitation:** Should have made the comparison in terms of RMSE. Could have applied K-fold cross validation **Explainability Limitation:** The paper showed good explainations in terms of graphs, but not explainability techniques like SHAP or LIME
Ngeni 2024	[22] This study proposes a random forest ML model for estimating docking station demand in bike-sharing stations. With added explainability to assist in resource allocation for operators.	Random Forest Model SHAP, feature importance, correlation **Dataset:** Smart Location Database (SLD)	Explainability showed how environmental and demographic, as well as greenhouse gas emissions, have considerable effect on the prediction, whether positive or negative. Worst MSE: 10.21 Best MSE: 0.53 Overall MSE: 0.74	**Model Limitation:** The study mostly focused on the effects of the features on prediction, not the usability on a particular system **Dataset Limitation:** The dataset was very wide and unfocused.
Mehdizadeh 2022	[11] Short-term forecasting of shared bike demand, using deep learning approach on community level with comparison over COVID-19 pandemic demand	Different CNN and LSTM models. CNN for feature extraction. Louvain Community Clustering. Prediction Horizon: 15 minutes **Dataset:** BIXI (Montreal)	**Best model:** CNN-LSTM RMSE: 5.02 MAE:3.28	**Practical Limitation:** 15 minute horizon is too small to be practical for any decision making. **Clustering Limitation:** Tried only one clustering method and number of clusters that removes too many stations. **Station Concern:** Number of stations is not stable (546 to 671 in 2021), and some stations are dropped or changed. Not all stations have the same capacity, capacity needs to be included. **Future Work:** Test same architecture on car-share.
Liang 2022	[14] Create bike sharing demand prediction system based on adversarial approach between different mods of transportation and their respective GNN models. Station Level.	Temporal Gated CNN, GCN, Domain Discriminator. XGboost, LSTM, MGCN, GWNET Multi-relational graph construction GNNExplainer Intra-modal graphs for station-less ride hailing Prediction Horizon: 4-hour intervals. **Dataset:** CitiBikes (NYC)	XGboost outperformed LSTM. The proposed model outperformed all other models at 11.334 RMSE compared to next best model MM-GWNET at 12.461 Multi-modal got the best results	4 Hour interval is too long for practical decision making. Using ride-hailing makes little sense, taxis go directly to the destination, BSS help with last mile transportation. A better system would have been busses instead. Ride-hailing is a good indication for traffic however. Future work includes using station-less BSS data, should be explainable on node level, and support for service expansion
Cho 2021	[8] Analysing spatial and temporal patterns of forecasted demands and comparing efficiency of repositioning strategies.	Random Forest (Grid search with 100 to 700 trees, 200 increase). Prediction Horizon: 1 hour. **Dataset:** Ttareungyi BSS, Seoul	**Pickup RMSE:** 2.85 **Return RMSE:** 2.66 4 days considered outliers	The paper mostly discusses repositioning strategies aided by demand prediciton rather than demand results itself.
Seo 2020	[13] Station Level Bike sharing demand prediction with Random Forest	Random Forest Heuristic clustering depending on shortage rate and excess rate into overstocked, unstable, stable, and short-of-stock stations Prediction Horizon: 1 Hour **Dataset:** PBS database in Seoul,	RMSE: 1.242 bicycles/hour Weather and lag resulted in 20%	Did not test lag on weather data. Did not test removing terribly performing stations. The dataset covers a 5 months period, which may not be sufficient. Clustering technique does not appear to be replicable with different datasets.

Guo 2019	[15] Predicting demand with Spatiotemporal GNN and use the result for re-balancing bikes using truck based systems. City-Wide demand prediction	ST-GNN Graph Convolutional Network (GCN) Gated Recurrent Units (GRU) Prediction Horizon: 1 Hour **Dataset:** CitiBikes (NYC)	RMSE 3.86 MAE 2.58 MAPE 7.9%	Future Work: Effect from weather and social events Linear programming techniques (e.g. Dimensionality reduction). Limitation: Demand definition could lose some context. Bike returns and rents should treated as separate issues.
Zhou 2018	[23] Demand prediction based on Markov chain model.	Markov chain. Prediction Horizon: Daily. **Dataset:** Zhongshan City System	Lowest Average Absolute Error: 6.57.	Concludes that regression models can't serve station-level demand forecasting. Unique assessment index seems difficult due to unequal traffic flow between stations. Testing results are not easily comparable with other papers.

3.1 Literature Review Discussion

3.1.1 Evaluation Metrics

The two most common evaluation metrics are Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). MAE simply takes the mean of the errors, where errors are the number of extra or missing bikes in our prediction. RMSE on the other hand takes the root of the squared error, where squaring adds a penalty to very high predictions, while rooting makes the error more interpretable.

$$MAE = \sum_{i=1}^{n} \frac{|x_i - y_i|}{n}$$

$$RMSE = \sum_{i=1}^{n} \sqrt{\frac{(x_i - y_i)^2}{n}}$$

3.1.2 Regression Models

Regular tree-based and gradient-boosting regression models are very commonly used for demand prediction, many outperforming both univariate and multivariate regression models as seen in Ashqar2017[24]. XGboost has only recently been used with Liang2022[14], outperforming LSTM as a regression model, showing that it is a capable alternative.

Deep learning models have been used since 2019[15] and have begun to see more use with Mehdizadeh2022[11] and Liang2022[14]. However, most of the research done focused on improving the dataset, with the prediction model often being a secondary concern. Mehdizadeh's predictions for example focused more on the improvement made by using weather and temporal data, making a far more noticeable improvement.

Markov chain was also used[23], but the results were not comparable to those in the other papers.

3.1.3 Feature Engineering

Feature engineering begins with determining the prediction horizon based on which the demand is calculated. However, the horizon is not standardized, ranging from an entire day[23] to 4 hours[14] to 15 minutes[11], but it seems that 1 hour is the most common horizon.

The number of stations then becomes important, as indicated in by Maleki2023[25], there is a constant demographics change, such as stores closing and other opening, changes in business locations, and so on that impact the viability of a machine learning model over a period of time, as the trained model no longer matches the existing data, this may cause concept drift[26] in the model. Additionally, not all papers necessarily cover an entire city, such as Liang2022[14] only covering Manhattan, since it is the busiest with the most accessible transportation, most stations outside an area could be removed from considerations.

Another manner in which stations can be removed is specifying the prediction level, station-level predictions are often used[14][13], demand over clusters can, however, be used instead[11], eliminating concept drift errors by instead specifying more wider areas where bikes are used.

However, the most successful method of improving predictions is to lag and window features based on the target, as well as adding weather data over a period of time[11][13].

A newer field in feature extraction includes Graph Neural Networks (GNN), which are used to extract Spatio-Temporal features as well as modeling[14], which has outperformed LSTM and boosting models. GNNs and their variations could be used for extracting the relationships between various stations in form of GNN embedding to be fed as features for other normal models.

3.1.4 Explainability

Despite the wide usage of models with built-in explainability, this method is rarely used, as most research papers focus primarily on improving the performance rather than explaining the reasons behind the predictions.

Liang2022[14] created an extension for GNNExplainer[18] for multi-relational graphs to explain demand between stations in relation to ride hailing and the subway system. A bike sharing system was used in Molnar's "Interpretable Machine Learning"[19] as an example to various explainability techniques, including LIME[16] and SHAP[17].

However, these techniques are not widely used in the explored literature, it is not clear which features are more likely to work globally and locally, and which do not. It can be useful if they are furthermore included in the literature, especially in mission critical machine learning[20] or models with much smaller prediction horizons.

Torres2024[21] did not apply explainability techniques, the paper however showed interpretations of the results in term of low demand and high demand stations, making a comparison between the model results for each over time, and showed comparisons for the model results per feature. Such explanatory

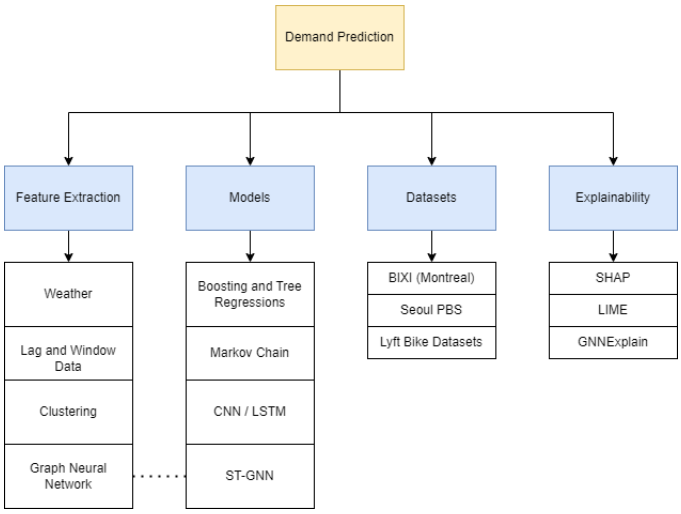


Figure 8: Literature review summary showing covered feature extration techniques, models, datasets and explainability techniques.

methods are useful; however, they do not explain each output individual results.

3.1.5 Datasets

Common datasets include several BSSs used in different cities, such as BIXI in Montreal[11], Ttareungyi in Seoul[13][8], and CitiBike in New York[14].

CitiBike is itself part of Lyft's bike sharing network, which includes several other cities in the United States which can be used in a similar format with the same processing techniques[10]. Therefore, any solution applied to one of them can be easily replicated in the other.

Liang2022[14] is unique in using data for Manhattan's ride hailing and metro services for using a multimodal prediction model, using CitiBike for the BSS data.

3.2 Literature Review Conclusion

There are several issues related to feature processing that need to be answered when creating a geospatial demand prediction model. Starting with the prediction horizon, between several minutes, hours, or days. This is not standardized among the many papers, as the prediction horizon changes the baseline on which algorithms are compared.

A choice also needs to be made for whether to use station-level prediction - which would result in more data with infrequent trips, or applying clustering to the stations, which would decrease the granularity of the predictions to cluster-level predictions. Additionally, the clustering method and the number of resulting clusters would need to be specified, as Mehdizadeh's method removed

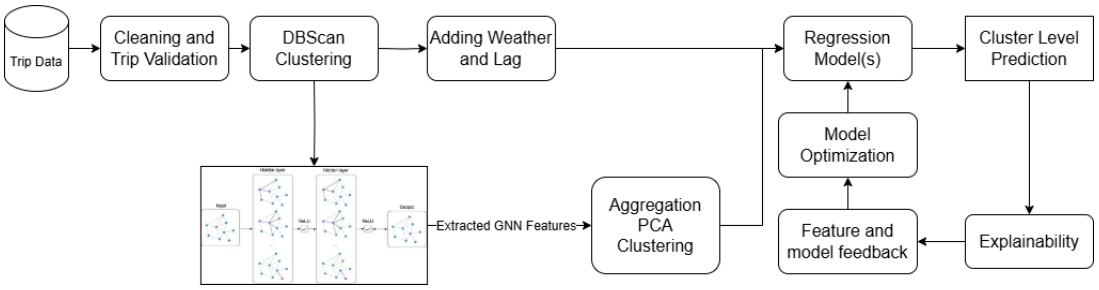


Figure 9: Proposed Model

too many possibly relevant stations even with the advantage of eliminating concept drift.

The application of explainability could be widened to help determine the features that could help in the prediction, making for easier ahead of time decision making even outside the prediction horizon.

4 Proposed Solution

4.1 Research Gap

There is a severe shortage in using explainable models. There is not enough research on the usability of features inserted or its applicability to a timeseries model.

While graph embedding features extracted with GNN have been shown to be a good method in finding relationships between stations and determining potential traffic flow, thus determining demand and returns in each station. As shown in Guo2019[15], the use of dimensionality reduction techniques, such as PCA, has not been explored sufficiently.

Reducing dimensions from graph embedding might reduce the size and training time of the model with minimum reduction to the RMSE.

Clustering techniques have applied in some occasions, but cluster sizes, numbers, and clustering techniques were not explored, neither is the application of DBScan. Clustering does reduce the granularity of the prediction, but reduces the dangers of concept drift.

4.2 Proposed Model

The proposed model focuses on perfecting preprocessing techniques, validating trips in general, and reducing prediction overhead by using clustering techniques such as DBScan.

Weather and lag features would be included with features extracted with a graph neural network, where a comparison between using dimensionality reduction methods would be made, with predictions made on cluster level with

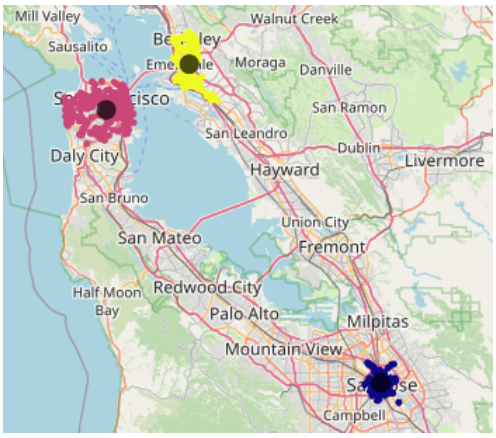


Figure 10: Baywheels station distribution in San Francisco Bay Area, yellow is Berkeley and Oakland, blue is San Jose, red is San Francisco City

regular regression models and deep learning models.

The initial results will be studied through explainable techniques and further improvement would thereafter be decided for the final model.

4.3 Dataset

The dataset used here will be Lyft's Baywheels[10], with data collected from January 2022 to March 2024. The initial number of rows is 5815902 (5.8 million rows), Relevant features being start/end time, start/end stations, and duration. These features will be filtered and converted into time series format, with added weather data from Meteostat API[27].

5 Experiments and Analysis

This section discusses the details of the preprocessing techniques used to improve the data quality prior to training the model, as well as feature extraction and creation methods used for various experiments.

5.1 Data Cleaning

The first step in cleaning the data is standardizing the stations. Some stations have the same exact ID but differ slightly in their GPS locations; standardizing the station location by taking the center resolves any possible ambiguity.

By contrast, some of the data include GPS locations but do not have an ID, these stations can be approximated to the closest standard station, a minimum was set at 500 meters distance, which can be further reduced for precision.

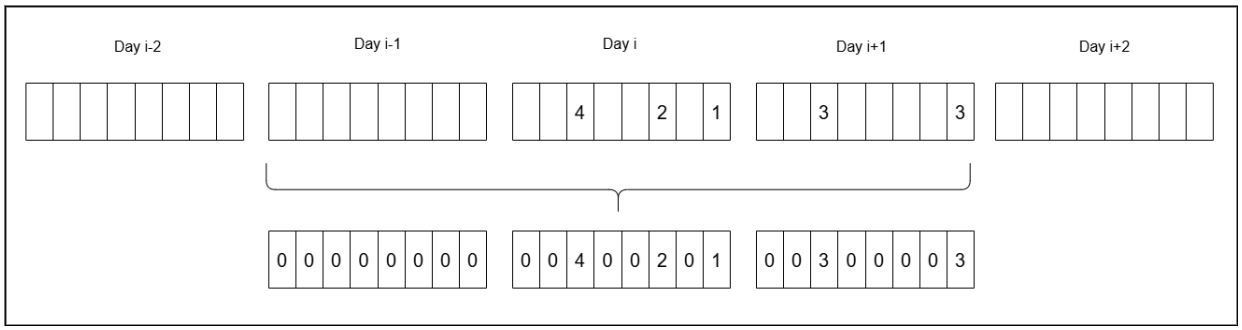


Figure 11: Filling Activity Gap: Since activities were spotted on Day i, Day i+1 and i-1 will be filled with 0 for all the empty hours of the day

There are some trips with very limited durations; on closer inspection, it seems those trips lead back to the same station or another one nearby. It is reasonable to think that a user would use the bike for a round trip for a short errand, but these short trips should not all be counted as real trips. Therefore, a trip duration cutoff needed to be established, 4 minutes for trips that return to the same station, and 2 minutes for trips going to different stations.

Since the majority of the trips take place in San Francisco city, the stations in San Jose, Oakland, and Berkeley will be excluded. Furthermore, with the clear separation of distance and interconnectivity between each of the cities (Figure 10), it makes little sense to train a shared model between them.

The final total number of cleaned trips is 4838234 trips (4.8 million).

5.2 Clustering and Aggregations

To ensure that the stations are not repetitive and in order to handle changes in station location and demographic changes, the stations will be combined through clusters. The clustering method used is DBScan, with an Epsilon (radius) value of 0.0041, which is equivalent to approximately 450 meters, the minimum sampling was reduced to 1, with the goal being that single stations can be given a cluster ID.

The clusters are then given their centroids as the new location coordinates that will be from this point instead of the raw station GPS data (Figure 12).

Demand is defined as **the number of records with a starting cluster**, while return is defined as **the number of records with an ending cluster**. Aggregates are created for each of these hours, forming our target variables. A simple model will be created to predict the returns, which will then be used as input for the main demand model. In addition, the demand between each cluster will form an adjacency matrix that is used for the graphs used for the GNN model.

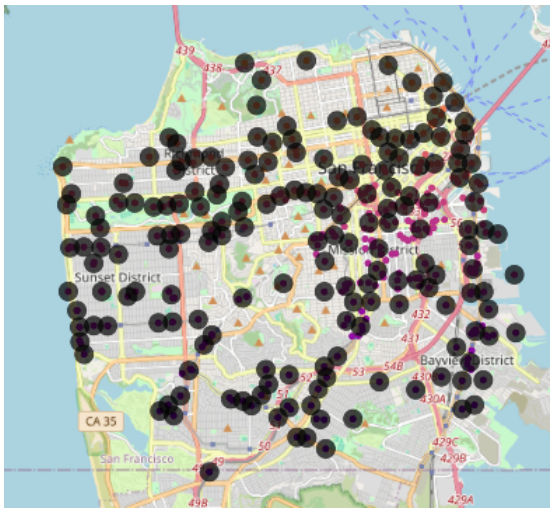


Figure 12: San Francisco Cluster Centroids: Isolated stations are considered their own cluster

However, it should be noted that there are days in some station clusters that do not have any activities, or if they do, there are certain hours of inactivity that cannot be predicted. To ensure that these hours are taken into account, the 3 days window of day, day-1, and day + 1, are all filled with records with 0 demand per station (Figure 11). This window was chosen to cover the 24 hour lag feature for the demand (and return) without adding too many records for training. The tests results will be divided between between overall results, non-zeros demand, and zero demand rows, to reflect the real performance of the model.

5.3 Exploratory Data Analysis

5.3.1 Station Distribution

There are 376 stations in the city of San Francisco, which, when clustered, form 183 station clusters, 117 of which contain only one station with the largest cluster containing 15 stations (Figure 13). The most active station clusters with the highest demand are mostly business districts in San Francisco, the more residential a district or neighborhood is, the less available stations are, the lower the demand (Figure 14).

5.3.2 Date-time Distribution

Most of the regular demand takes place during the working weekdays with the weekend (Sunday) receiving low demand compared to the other days, Monday following it as the lowest as the work week is still at its beginning. Unexpectedly,

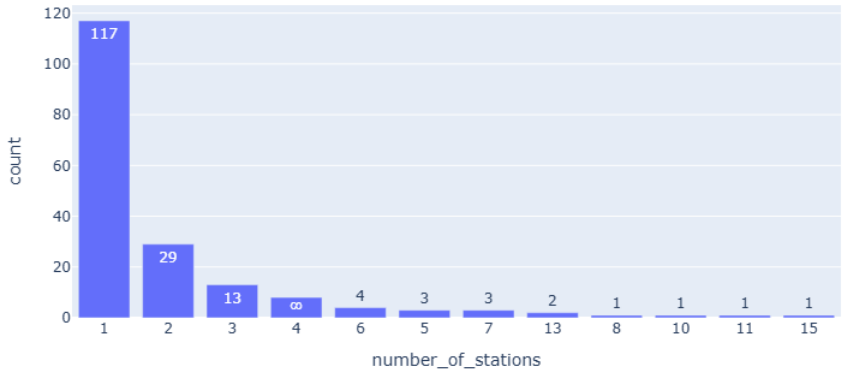


Figure 13: Distribution of number of stations per cluster. Most clusters have a single station (117), the largest numbers of stations in a single cluster is 15

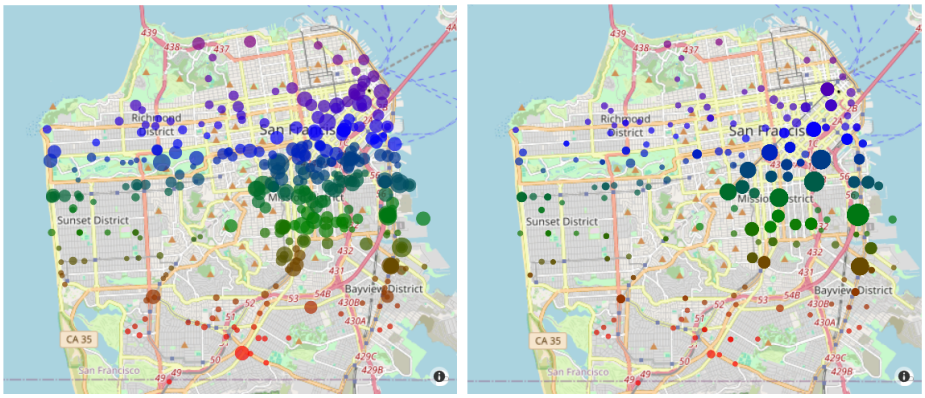


Figure 14: Distribution of demand per station (left) and station cluster (right), both scaled. The further station clusters are from the business districts, the less likely they are included in a cluster

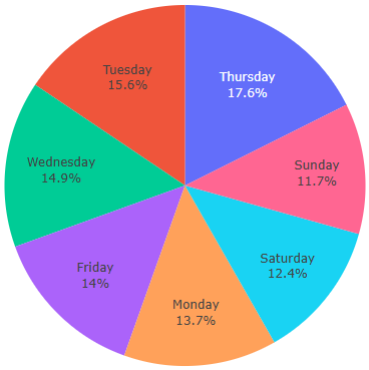


Figure 15: Demand percentage for weekdays, the weekends, Sunday and Saturday, have a lower percentage compared to the other days

Saturday is not the second lowest day in the overall data despite being part of the weekend (Figure 15).

A general trend for the rest of the year indicates that the months between April and October are the busiest, followed by a downward trend as the holiday season begins in the United States, dropping to the minimum in December, after which activity and demand regain momentum in January (Figure 16). This is further observed in hourly demand, as demand in Winter falls sharply compared to Summer and Fall on hourly average; interestingly, however, is that the usual hourly pattern remains mostly the same regardless of the weather. However, this pattern is less visible on weekdays, where the regular weekdays (Monday to Friday) all peak twice during rush hours, while weekends (Saturday and Sunday) have only one peak in the middle of the day (Figure 17).

5.3.3 Weather Correlation

Looking at the correlation matrix (Figure 18), it would seem that the weather features have very little correlation with demand; however, this may be due to the generally stable climate in San Francisco. Moreover, by looking at the scatter plots (Figure 19), it is apparent that extreme conditions cause a decline in demand, particularly high precipitation and wind speed, whereas temperature takes a much more normal-like distribution, where the demand peaks between 12-23 degrees Celsius and tapering off for either extremely cold or extremely temperatures with a few occasional outliers. Humidity is the least intuitive because it is understood that humidity indicates discomfort, but besides several outliers, most demand takes place between 50% and 80%. From the correlation matrix (Figure 18), the correlation between precipitation (which indicates rain) and relative humidity is not strong (0.15), indicating that despite possible discomfort, the weather road conditions may not be severely impacted in a manner that would discourage bike riding.

While other features show a vague indication, the Coco weather codes are

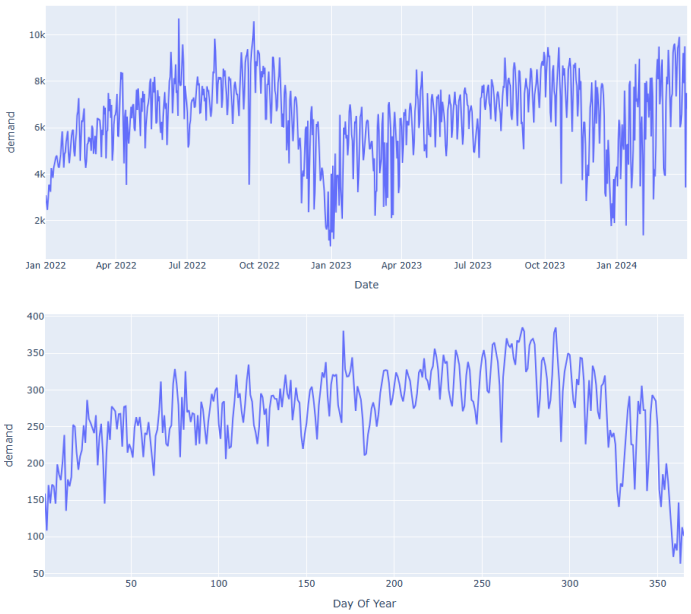


Figure 16: Top: Overall demand by date. Bottom: Demand by day of year. Note the decreasing demand at the end of each year and slow increasing trend from the beginning

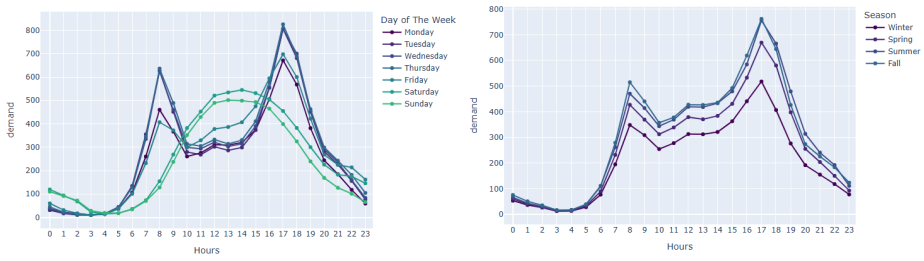


Figure 17: Average hourly demand based on weekdays (left) and seasonal (right). Note the same hourly pattern regardless of season, and the different pattern between weekdays and weekends

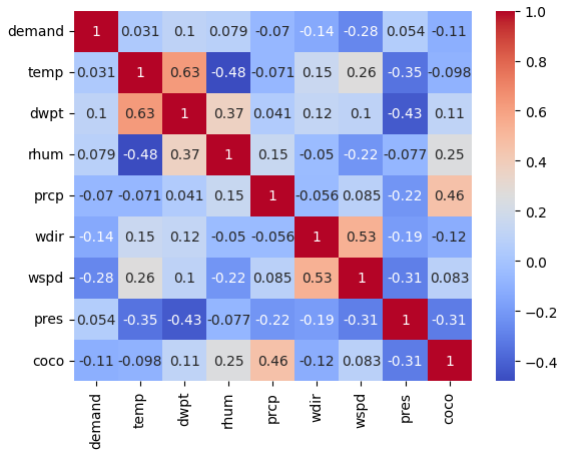
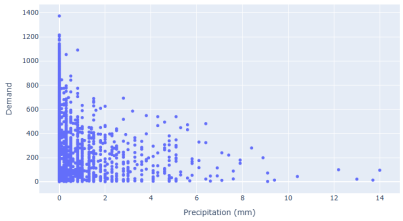
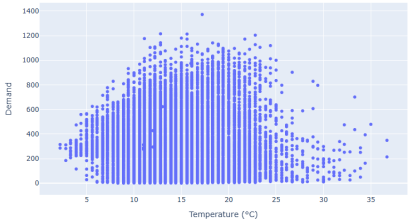


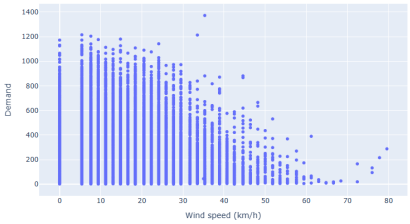
Figure 18: Weather features correlation with demand, showing the weak correlation between demand and weather features, the highest absolute negative correlation (and absolute highest) is wind speed (wspd), the highest positive correlation is dew point (dwpt)



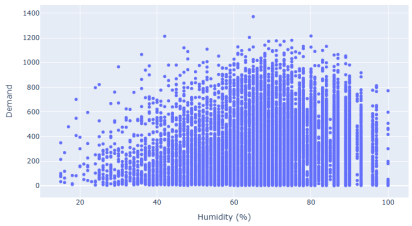
(a) Precipitation



(b) Temperature



(c) Wind speed



(d) Relative Humidity

Figure 19: Weather features scatter plot with demand

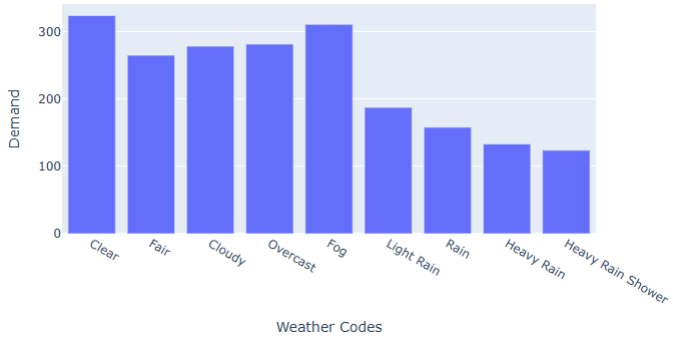


Figure 20: Average demand by weather codes, from left to right, the weather condition becomes increasingly worse. Foggy conditions is the second highest for demand, any amount of rain results in increasingly lower demand

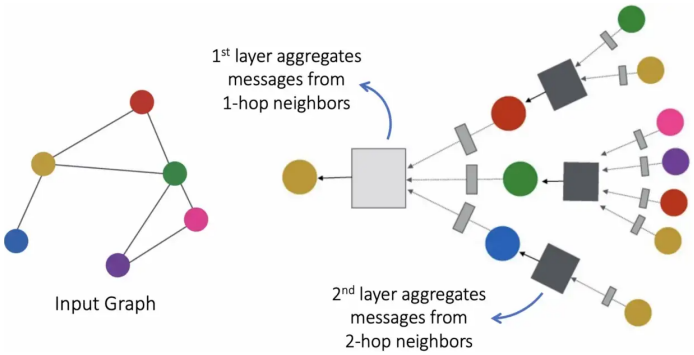


Figure 21: GNN Message Passing[28]

more direct (Figure 20). The average demand during calm weather conditions (clear, fair, cloudy, overcast) is high, with a surprising increasing during foggy conditions; there are several possibilities, but it is likely that fog coincides with the morning rush hour. The average demand drops significantly during rainy conditions with increasing intensity.

5.4 Graph Neural Network (GNN)

The nature of this problem can be defined in terms of graphs, where the edges represent the number of trips going from station A to station B, and in this sense, GNNs are naturally common in their application as observed in the literature review. The basic concept is that the message passing between the GNN layers will inform the prediction of a demand at a certain node using the state of the rest of the network (Figure 21).

There are several methods for using the GNN for demand prediction, but

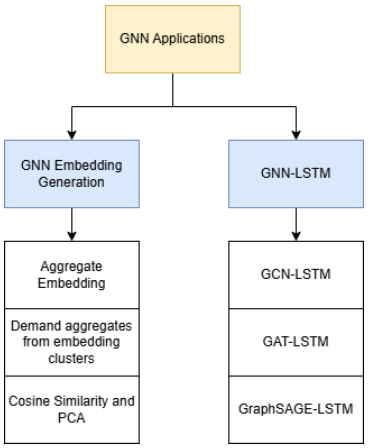


Figure 22: Methods of applying GNN, whether for generating features for other models, or as a standalone model

they can be divided into two: 1) The GNN-LSTM model for predicting demand. 2) Graph embedding generator used to generate features for traditional models, features include embedding aggregates themselves, clustering stations, and aggregating demand values, or using cosine similarity between stations and applying dimensionality reduction (PCA) (Figure 22).

All of these features are added from the previous hour as lag features. In addition, the graphs themselves include the demand degree feature, which shows the active edges and the overall activities of the network.

All the data in the year 2022 is being used as training data for the GNN model used for feature generation, this restricts the training data for the models to 2023-2024. The GNN-LSTM on the other hand needed to be reduced due to hardware limitations, being limited to data from July 2022 onward. In all cases, the final models are tested using data up to March 2024 (Figure 23).

5.5 Model Features

The features can be divided into engineered features generated from the dataset itself, features generated from related models, and features imported through Python packages and APIs (Table 3).

5.5.1 Date-time and station clusters

Stations clusters are the IDs given to the clustered stations as mentioned earlier from DBScan clustering, they include several stations between them, and in total comprise of 183 station clusters. The date-time features are simply extracted from the station start and end time. Both of these series of features are important identification of the record, showing the activity of a certain station at a certain time.

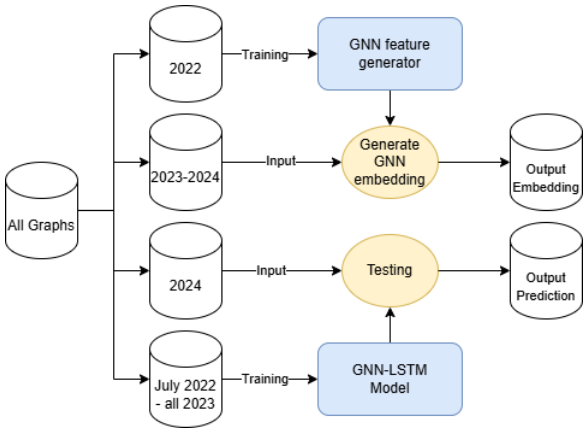


Figure 23: Graph subsets for GNN. 2022 data trains feature generator, July 2022-2023 data trains GNN-LSTM prediction model

It should be noted that "day of week" is different from "day", the latter referring to the day within the calendar year, the former is expected to be more immediately significant as it expresses weekends and working days.

5.5.2 Holiday and weather data

The main purpose of a BSS is a means of transportation; therefore, it is very expected that people do not need to use them on holidays, since most of the transportation is required for work days, which is a very similar behavior to weekends. The python holiday package shows the day of the week by country.

Meteostat[27] shows the historical weather data on several time frames in terms of the closest weather tower to San Francisco, but the data provided for this use case are hourly, the only columns that were not used from Meteostat's hourly data is sunshine total, snow, and wind gust, the three of which being mostly unapplicable and mostly null data.

5.5.3 GNN features

These features are generated detailed statistics from the GNN models and graphs in general, and are meant to inform the final model of the activity of the network or relevant parts of the network to the node being predicted.

The graph embedding comprises 50 dimensions per station per hour, which are then aggregated into 50 features by mean or variance and used as input.

Additionally, by taking the embedding and clustering it with KMeans clustering (where K=4) for each hour, we obtain an hourly similarity overview of the demand between stations (Figure 25). We can then calculate the mean and total demand for each hour and use it as a feature.

Those 50 dimensions can be used to calculate the cosine similarity between each station, resulting in 182 features per station per hour. To make these

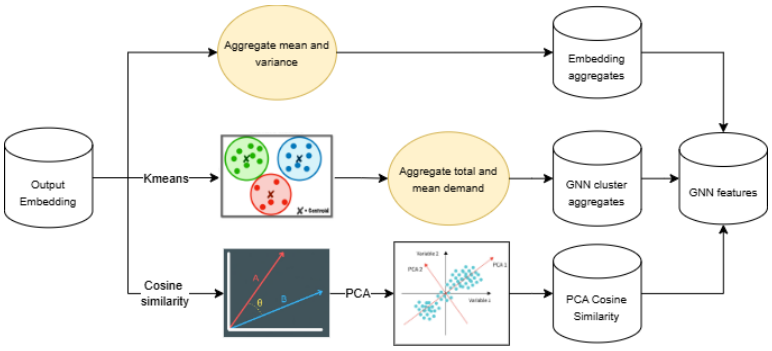


Figure 24: GNN features summarized. Three types of GNN features are produced, aggregated (mean and variance) embedding, GNN cluster aggregates, and dimensionally reduced cosine similarity (with PCA)

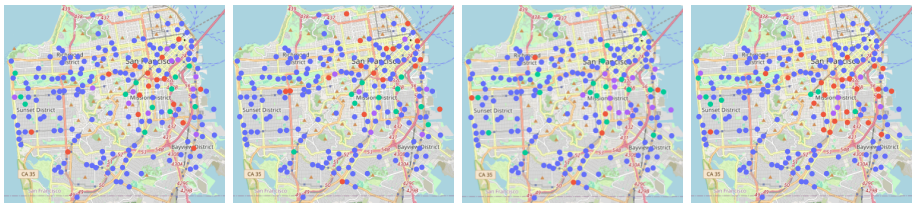


Figure 25: 4 hours of changes in clustering according to graph embedding similarity. Notice stations being in different clusters every hour in terms of demand similarity informed by the GNN embedding

features more manageable, those features are reduced with PCA dimensionality reduction, several number of features were tried, but the best result was obtained by reducing the cosine similarity features to one feature.

$$CosineSimilarity = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sum_{i=1}^{n} B_i^2}$$

5.5.4 Lag features

The GNN features are technically lag features, showing the activities of the network in the previous hour. But any feature that is viable can be made into a lag feature. Demand and weather data from previous hours are calculated and provided as lag features, mainly for 1 hour, 2 hours, and 24 hours. Network wide demand and network activity (expressed through node activities such as demand degrees) are also lagged to 1 hour.

	Metric	Score
Overall	MSE	3.0241
	RMSE	1.7389
	MAE	0.8839
	MAPE	0.3613
Non-Zero	MSE	7.0882
	RMSE	2.6623
	MAE	1.7022
	MAPE	0.3631
Zeros	MSE	0.4235
	RMSE	0.6507
	MAE	0.3601
	MAPE	0.3601

Table 2: Returns model results output

5.5.5 Returns

Predicting bike returns is a difficult problem that deserves more attention since it solves a different aspect of the logistics problem with BSS, which is the stations that can have excess bikes to be rebalanced from. It makes sense that demand at a certain station would increase providing that more bikes were available; this correlation has been previously explored to define the target before[15], which, as previously said, is not very useful, since the lost context could be more useful if they are treated as separate problems.

To that end, a returns prediction model was trained. The output of either models can be used as an input for the other. For this scope, the returns model is used as input for the demand model.

The returns model is an XGboost model which uses date-time data, weather data, holidays, and returns lag, using data from 2022 as input, with tests being done on 2023-2024 data, and the results for that are then used for the demand model.

Table 2 features the performance results of the model, showing an overall RMSE of 1.7389, where non-zero hours had an RMSE of 2.6623.

5.6 Environment Setup

All data preparations, pre-processing, fitting, and testing, were performed using a Google Colab environment[29]. Neural network models were first tested for viability on CPU before being fully trained on T4 GPU, while non-NN models were trained on CPU only.

The system had two 13GB RAM Intel(R) Xeon(R) running at 2.20GHz, and 15GB RAM on Tesla T4 GPU.

Feature	**Sources**	**Summary**
Station cluster	Given ID from DB-Scan clustering	183 station clusters
Weather Data	Meteostat API	Temp (temperature in °C)
		Dwpt (Dew point in °C)
		Rhum (relative humidity in %)
		Prcp (Precipitation in mm)
		Wdir (Wind Direction in degrees)
		Wspd (Average wind speed in km/h)
		Pres (Sea level air pressure in hPa)
		Coco (Weather condition code)
Holiday	Holidays pip package	Whether the day specified is a holiday in the US.
Date-time features	Generated from trip start date (Demand) and end date (Returns)	Month
		Day
		Hour
		Quarter
		Day of Week
Lag features	Generated from prepared data: Demand and Weather Data	Lag features from 1 hour, 2 hours and 24 hours before.
		Total demand (1 hour lag)
		Demand degrees (1 hour lag), how many edges are active between the nodes in the station-cluster
GNN Features	Generated from GNN model feature engineering	Mean GNN cluster demand
		Total GNN cluster demand
		Mean embedding
		Variance embedding
		Cosine Similarity (reduced with PCA)
Returns	Returns model	Predicted returns generated from the simplified returns model

Table 3: Features summary

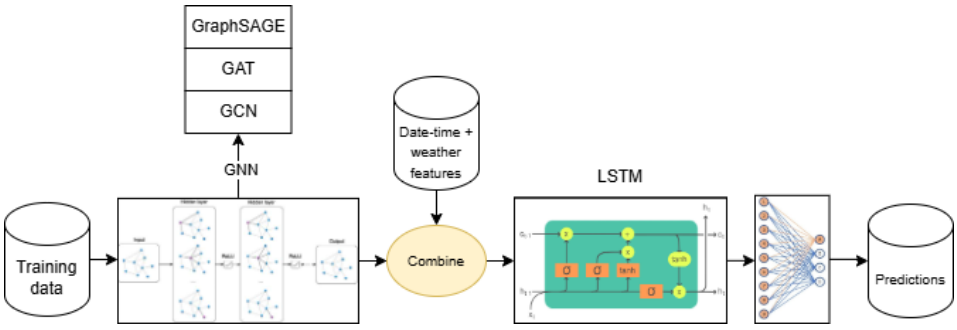


Figure 26: GNN prediction pipeline. GNN acts as encoder for embedding, it is then combined with other features and fed to LSTM and a fully connected layer for predictions

Providing a bigger testing environment with more RAM and GPU power, a larger dataset could have been used for fitting and testing; this hardware limitation can, however, be overcome.

All tests were carried out in 10 different iterations, and the final results presented are the mean of all iterations.

5.7 Models

5.7.1 GNN Prediction models

The three main GNN models used for comparison are GCN (Graph Convolution Network)[30], GAT (Graph ATtention)[31], and GraphSAGE (Graph SAmple and aggreGatE)[32].

Previous benchmarks on link prediction have shown that GCN is simpler and more direct than GraphSAGE and GAT, as well as using lower computational requirements[33]. GAT uses an attention mechanism which identify the more relevant neighbors, which requires more computational resources in general[31], and GraphSAGE is designed to handle very large feature-rich evolving graphs where the model needs to handle previously unseen nodes[32]. Considering the smaller size and simplicity of the graph inputs used for this paper, GCN is the better option, GraphSAGE however could be considered as an alternative to solve the data and concept drift issues for new stations and station clusters, even if GCN is better performance wise.

The embedding from the GNNs is then combined with weather data and date time feature, they are then used with an LSTM network and a fully connected layer to produce demand prediction per hour per station clusters (Figure 26).

As shown in the results (Table 4, Figure 27, the overall performance of GAT in terms of RMSE is better than both GCN and GraphSage, however, when looking at the zero demand test results, it is clear that boy GraphSage and GCN perform better on both RMSEs and MAE, with GraphSage having a slight advantage on MAE. For non-zero demand test results, GAT is by far the

	Metric	GCN	GAT	GraphSAGE
Overall	MSE	5.2666	5.2214	5.4079
	RMSE	2.294	2.2851	2.325
	MAE	0.766	0.7710	0.7674
	MAPE	0.3052	0.3186	0.2997
Non-zero	MSE	22.8687	22.4557	23.8592
	RMSE	4.7821	4.7387	4.8846
	MAE	2.64302	2.5901	2.6601
	MAPE	0.4484	0.4349	0.4322
Zeros	MSE	0.5890	0.6415	0.50468
	RMSE	0.7674	0.8009	0.7104
	MAE	0.2672	0.2876	0.2644
	MAPE	0.2672	0.2876	0.2644

Table 4: GNN Prediction Model Results. Note GAT outperforming the others for non-zero predictions, but GraphSage being the best for Zeros predictions

best performer. However, all models did have a low MAPE and MAE compared to the regression models, as will be observed in the following section.

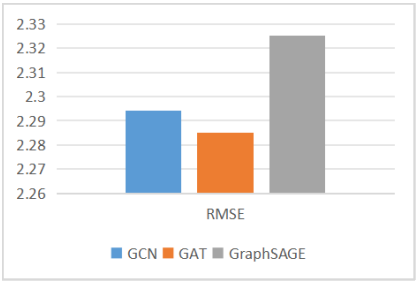
5.7.2 Regressors

In addition to experiments with GNNs, experiments with regressor algorithms have been conducted using GNN features indicated in figure 24, not all features, however, are useful and some may lead to overfitting, therefore each of these features, as well as different combinations of the other features listed in table 3.

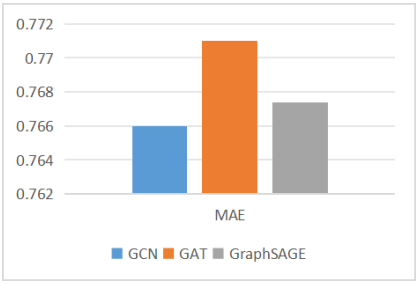
The regressors used are two boosting models, XGBoost[34], and lightGBM[35], in addition to random forest. Random forest has been a very reliable algorithm used in many previous papers, XGBoost was less used, but was reported to outperform LSTM in a previous paper[14].

The early experiments with simple features consisted only of date-time features as a baseline, in addition to increasingly more complex features. Weather and lag features were used in a comparison (Table 5, Figure 28 a and b), in the early models using only weather and datetime features, Random Forest outperformed both boosting models; however, when more complex features were introduced, namely lag features, Random Forest began to struggle training time wise as well as consumed resources, while also performing worse than the boosting models. Using word embedding alone has improved the Random Forest results, but they were still slightly higher than XGBoost.

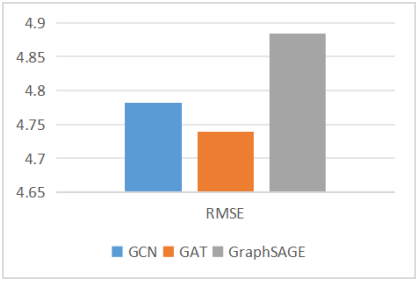
After adding more complex features from GNN (Table 6, Figure28 c and d), it becomes clear that not only does Random Forest performance become increasingly worse, but XGBoost becomes the absolute best model for all subsequent experiments. LightGBM is faster than XGBoost according to a comparative study[36], however, the practical difference between the two with this dataset was insignificant. Moreover, at this stage of the tests, the features added to



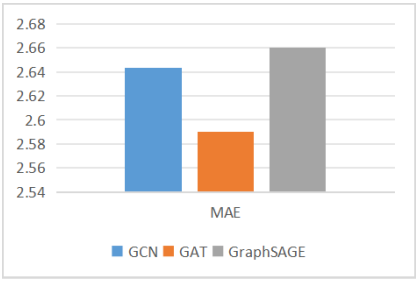
(a) RMSE (Overall)



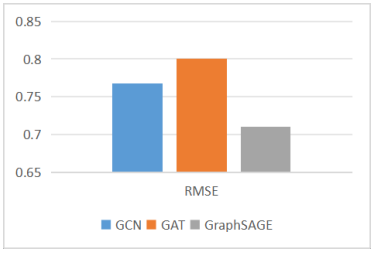
(b) MAE (Overall)



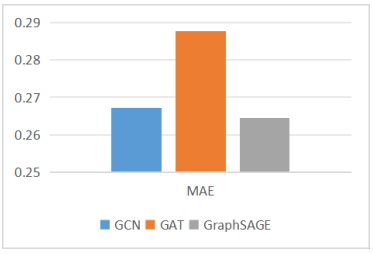
(c) RMSE (Non-Zeros)



(d) MAE (Non-Zeros)



(e) RMSE (Zeros)



(f) MAE (Zeros)

Figure 27: Results from Table 4 visualized

	Metric	XGBoost	LightGBM	Random Forest
Weather	MSE	4.0615	4.6536	3.7531
	RMSE	2.0153	2.1572	1.9373
	MAE	1.0671	1.184	1.0211
	MAPE	0.435	0.5325	0.4136
Weather + Lag	MSE	3.6805	3.8179	4.1405
	RMSE	1.9185	1.9539	2.0348
	MAE	1.0252	1.0565	1.0904
	MAPE	0.4157	0.4309	0.459
Embedding	MSE	4.1124	4.6311	3.7223
	RMSE	2.0279	2.152	1.9293
	MAE	1.0718	1.1832	1.0256
	MAPE	0.4365	0.5333	0.4146

Table 5: Early model results, date-time features and the listed features included.

Random Forest have become too complex, the training time needed increased to several hours with no clear improvement in the results compared to XGBoost.

Therefore, XGBoost is the algorithm to be used for any further experiments. Except for one experiment which was centered on using the embedding features as is without aggregation, random forest not only outperformed the boosting algorithms again, but also outperformed the GNN algorithms in the previous section. It is clear that GNN is better at reducing MAE, while all regressors are better at reducing MSE and RMSE.

After creating the bike returns prediction model, the returns feature was also added to the model, more GNN features were then generated by applying KMeans clustering using the GNN embedding (Figure 25) and calculating the mean and total lag demand from stations in this screenshot in time. The GNN lag cluster demand is meant to provide the model with global state of predictions in a similar manner to lag demand, but on a more local level showing similar stations, which will henceforth be referred to as "cluster details". This resulted in even lower error rates (Table 7, Figure 28e); it was therefore, decided that the final models with the full assessment will use weather and lag features, returns model result, and embedding clustering details.

In addition to the mean and variance of the GNN embedding, there were two more features that needed to be tested, the first is the dimensionally reduction of the mean and variance embedding using PCA, the other is applying Cosine Similarity on the GNN embedding and then using PCA on it instead (CS-PCA). Looking at the final results, the CS-PCA based model had the best overall and non-zero RMSE (Table 8, Figure 29), PCA based model, however, showed a slightly better performance over all metrics with zero value predictions, the differences however are very minimal.

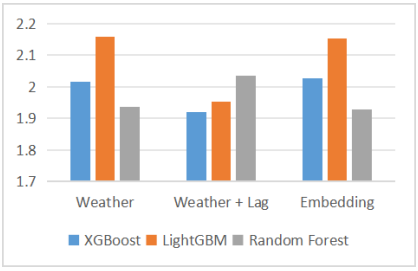
However, it should be noted that regardless of the best model in terms of MAE, the GNN models (Table 4) with MAE = 0.7710 for GAT compared to the

	Metric	XGBoost	LightGBM	Random Forest
Mean Embedding	MSE	3.6470	3.8314	4.1419
	RMSE	1.9097	1.9573	2.0352
	MAE	1.0248	1.0584	1.0913
	MAPE	0.4165	0.4315	0.4569
Variance Embedding	MSE	3.6794	3.8116	4.1765
	RMSE	1.9181	1.9523	2.0437
	MAE	1.0264	1.0575	1.0894
	MAPE	0.4178	0.4322	0.4549
Mean and Variance	MSE	3.6863	3.8236	4.1487
	RMSE	1.9199	1.9554	2.0368
	MAE	1.0304	1.0579	1.0913
	MAPE	0.4197	0.4321	0.4565

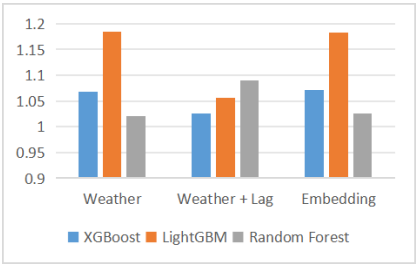
Table 6: GNN aggregates features results, mean embedding, variance embedding, and both combined

	Features	MSE	RMSE	MAE	MAPE
Regular Model	Mean Embedding	3.6470	1.9097	1.0248	0.4165
	Variance Embedding	3.6794	1.9181	1.0264	0.4178
Returns	Mean Embedding	3.5342	1.8799	0.9973	0.3947
	Variance Embedding	3.5494	1.8840	1.0002	0.3975
Clustering	Mean Embedding	3.5229	1.8769	0.9957	0.3947
	Variance Embedding	3.5192	1.8759	0.9947	0.3943

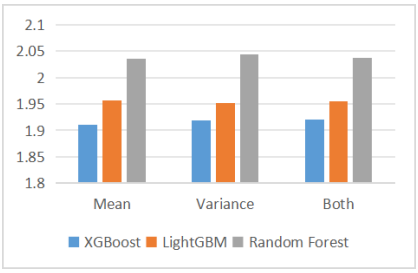
Table 7: GNN aggregates features results before and after adding returns features and clustering details



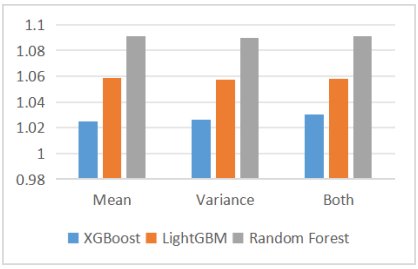
(a) RMSE Early Regressors



(b) MAE Early Regressors



(c) RMSE with GNN embedding aggregates



(d) MAE with GNN embedding aggregates

(e) RMSE with returns and clustering details

Figure 28: Results from Tables 5 (a, b), 6 (c, d), 7 (e) visualized. Notice the initial superior performance for Random Forest and the consistently middling performance of LightGBM, the performance improves as more different features are added, with XGBoost emerging as the best algorithm

	Metric	Mean	Variance	PCA	CS-PCA
Overall	MSE	3.5229	3.5192	3.5222	**3.4781**
	RMSE	1.8769	1.8759	1.8767	**1.8649**
	MAE	0.9957	0.9947	**0.9917**	0.9931
	MAPE	0.3947	0.3943	**0.3921**	0.3951
Non-Zeros	MSE	7.1851	7.1787	7.1956	**7.0816**
	RMSE	2.6805	2.6793	2.6824	**2.6611**
	MAE	1.6903	1.6886	1.686	**1.6827**
	MAPE	0.3543	0.3539	**0.3529**	0.3533
Zeros	MSE	0.5287	0.5272	**0.5187**	0.5317
	RMSE	0.7271	0.7260	**0.7202**	0.729
	MAE	0.4276	0.4273	**0.4241**	0.4291
	MAPE	0.4276	0.4273	**0.4241**	0.4291

Table 8: GNN embedding features experiments. Using mean and variance of the the GNN embedding, PCA of the embedding, and Cosine Similarity PCA the embedding

CS-PCA model with MAE = 0.975. However, these MAE results were biased towards the zeros predictions, whereas non-zeros for GAT show MAE = 2.5901 compared to the CS-PCA model showing non-zeros MAE = 1.869.

In other words, GNN is better at identifying idle station clusters, whereas the regressor models work much better for identifying active stations with lower instances of high error rate, as indicated by the lower RMSE.

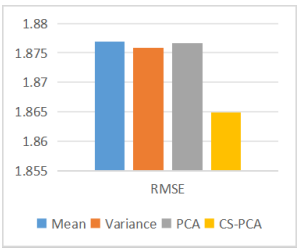
Overall, considering the requirements of identifying non-zero values, it is concluded therefore that regressors with GNN based features are the better option for making bike demand predictions, and considering that RMSE is an important parameter for detecting high instances of error rate and the very minor differences in results for the other metrics, it is also concluded that CS-PCA is the best model for the remainder of the experiments with explainability.

5.8 Explainability

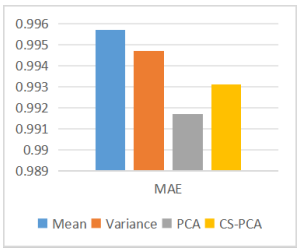
There are two methods that were explored to explain the results of the model; they are the feature importance of XGBoost models and SHAP values. Feature importance provides explanations of the overall results of the model on the technical level during training, while SHAP provides a more direct view of the individual results of the tests, making it more viable for operational use compared to feature importance, which can only be used to explain the training itself.

5.8.1 Feature Importance

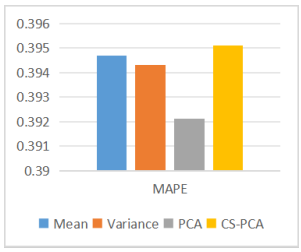
The feature importance built-in to XGBoost can be calculated using three methods, gain, cover, and weight[37]. Gain refers to the average information gain



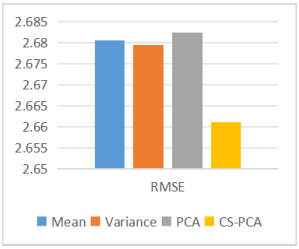
(a) RMSE (Overall)



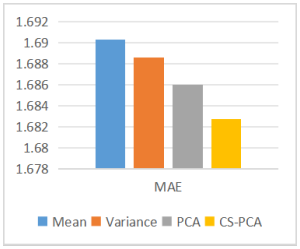
(b) MAE (Overall)



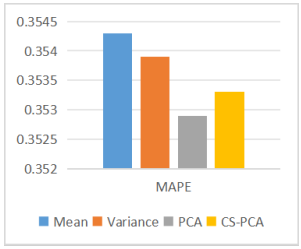
(c) MAPE (Overall)



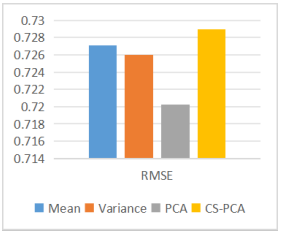
(d) RMSE (Non-Zeros)



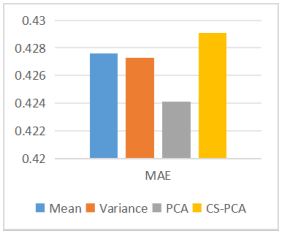
(e) MAE (Non-Zeros)



(f) MAPE (Non-Zeros)



(g) RMSE (Zeros)



(h) MAE (Zeros)

Figure 29: Results from Table 8 visualized, showing the performance of using of GNN features in the model.

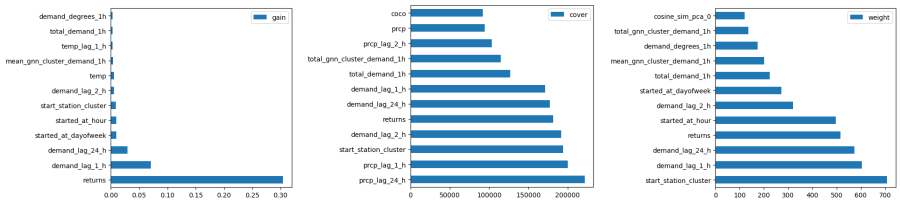


Figure 30: Feature importance on the best model showing the top 12 features. From left to right, gain, cover, and weight.

taken from splits across all the trees, cover refers to coverage, which refers to the amount of observations influenced by a split from a feature, while weight is the number of times a feature is used to split a model.

Gain is generally seen as the most relevant metric for feature importance, followed by coverage in showing the impact of the model on the data used, whereas weight may not be informative but it can be used for debugging the model for overfitting.

Looking at the features with the highest gain (Figure 30), it is clear that the return feature has the highest impact by a large margin. During the debugging process, it was found that of the top 10 model features, only 2 are not in the top 10 for both gain and weight, the exact ranking not being relevant. For the other 26 features, the gain values are extremely close that the exact weight ranking does not show a clear indication of overfitting. One of two conclusions can be made, either that the return feature is causing a leakage and should therefore be removed due to a dominating impact or that since gain and weight rankings are very similar, then the model is not overfitting and the return feature should be kept. Based on the knowledge that the return feature is itself an output of a model, it is hard to conclude that it is indeed causing a data leak.

5.8.2 SHAP values

The feature importance metrics use statistics gained from training the model, which gives a good overall indication regarding the training accuracy, but it does not explain much in regards to testing accuracy. Feature importance is also calculated over the entire model, which cannot therefore make it difficult to determine if the model works for the entire dataset or if possible partitions can be made to train several models.

The solution for these issues is SHAP (SHapely Additive exPlanations)[17]. The SHAP explainer uses the test data instead of the training data, and the Shapely values calculated through SHAP can then be calculated either individually, globally, or individually with a global explanation chart.

Figure 31a shows the effects of each individual prediction based on SHAP values, where for each feature, the feature value is represented as a point, and the SHAP values indicate whether the effect of the feature was positive or negative (high demand, low demand, no effect), this is already a very good indication by

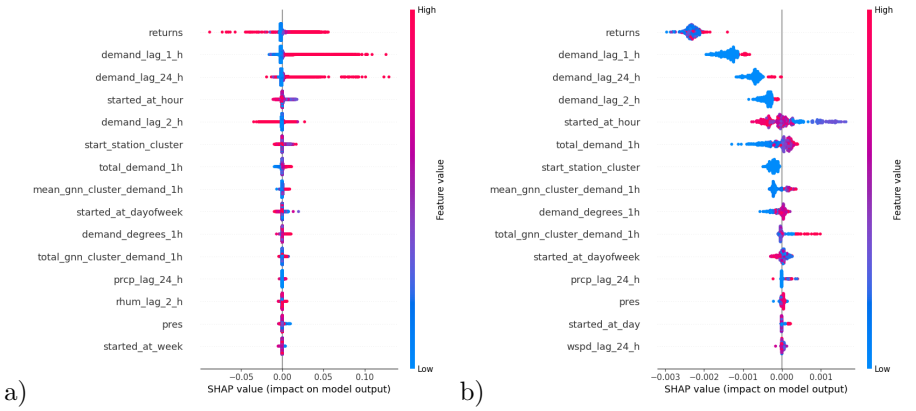


Figure 31: SHAP bee swarm chart values ranked (best 15). Showing each individual effect of a value on a prediction, ranked based on average Shapely values. a) Overall test cases. b) a random station cluster.

itself, with regard to testing data, but more importantly, it can be isolated by station, as Figure 31 b shows the same chart but for a single station cluster. The overall SHAP in Figure 31 a seems contradictory, as the starting hour is unclear, since some stations are simply more active than others at different times of day. However, when looking at Figure 31 b, it is clear that this station is more active at early hours, where low values of the starting hour (early hours of the day, bluer) more often lead to a positive effect on demand (higher demand). It can also be inferred that this station cluster is more affected by the overall network, since GNN features such as demand degrees and mean gnn cluster demand have a much clearer divide. This explanation can be further expanded on monthly, weekly, and even daily averages, and can even include individual predictions.

5.8.3 Feature Trimming

Feature importance is one of the methods used for feature selection, where the features with the least importance are removed and the model re-trained. As stated previously, this change applies to the entirety of the dataset in the network without providing any distinctions.

However, for SHAP on the other hand, the different stations can be divided. The RMSE metric can be calculated separately for each station cluster; using those RMSEs, the station clusters were divided into "worst performing" and "regular performing" clusters (Figure 33). The SHAP values for each of these categories were then calculated (Figure 32). For the best performing clusters, up to 5 features were removed, and their records were re-trained with a new model. For the worst performing clusters, up to 19 features were considered to be excluded from their new model, the performance did improve when all of them were removed, but other additional real-world considerations of the features and testing showed that 14 of those 19 features show the most optimal

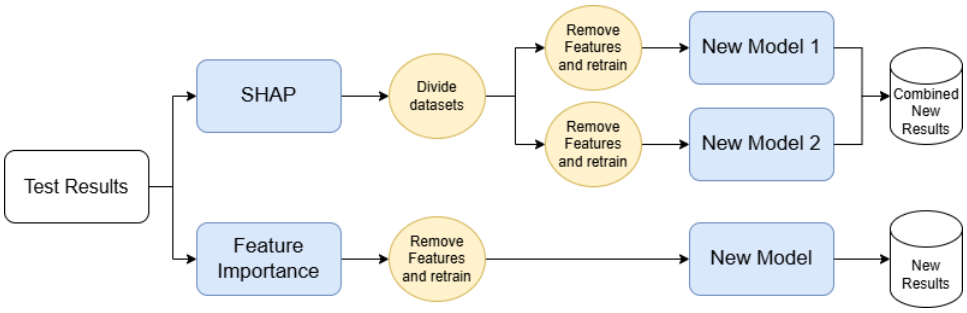


Figure 32: SHAP vs Feature Importance feedback loop. Feature Importance only trims the dataset from features over the entire dataset, SHAP trims features from two different subsets of data depending on their own SHAP scores, then combine the results for testing purposes.

improvement when removed.

The partitioned results of the test data for each of the new partitioned models were combined to produce the final results, showing an improvement compared to the original single model (Figure 9); by comparison, the features removed from feature importance did not show any improvement and have, in fact, become worse than the original.

The features that were removed were some additional date-time and weather features. The month, day of month, quarter and week were not very relevant as opposed to simply the day of the week and the hour, as well as whether the day was a holiday. The wind speed feature (wspd) was removed for the worst and regular stations, but the worst clusters also removed the wind direction (wdir), temperature (temp), and dew point (dwpt), as well as the lags for wspd and temp, which were not removed for the regular clusters model. Conversely, the worst clusters model performed better with day-of-month feature and was kept as opposed to the regular clusters.

With the exception of relative humidity (rhum), feature importance did include some of the same removed features from SHAP, but their removal improved the performance. This could indicate that the worst performing clusters affected the rest of the clusters during the training, SHAP however was better at identifying which clusters are affected by which feature and allow a better decision making for model re-training. Although it is theoretically possible to create a model for each station, the lack of exact patterns for a less busy station requires that station clusters be included together to help establish patterns for the model, whether the stations are busy or not.

It should be noted that through the progression of the models, the GNN-LSTM models, especially GraphSAGE, consistently outperformed all models in terms of predicting zero demand, SHAP trimmed models outperformed them in terms of RMSE (0.689 vs 0.7104), but not MAE and MAPE (0.41 vs 0.2644). It can, however, be concluded that this final model is overall the most optimized and best performing model on most metrics and data partitions.

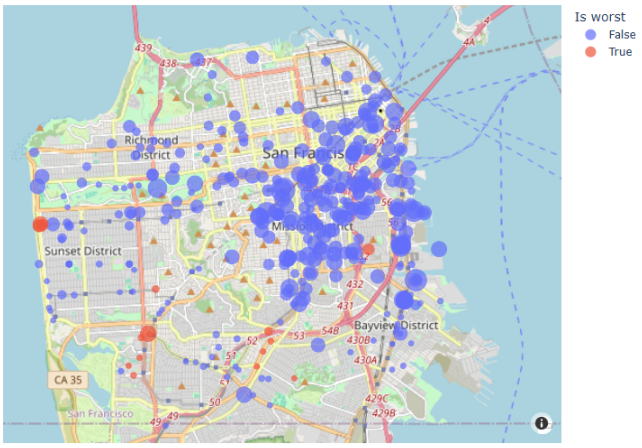
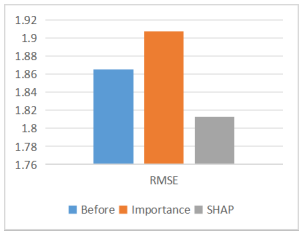


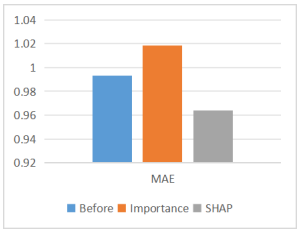
Figure 33: Map showing the station clusters. Worst performing clusters in red, regular performing station in blue. Most except one cluster are outside the range of the business district.

	Metric	Before Trimming	Feature Importance	SHAP
Overall	MSE	3.478	3.637	3.288
	RMSE	1.8649	1.9071	1.813
	MAE	0.9931	1.0183	0.964
	MAPE	0.395	0.4133	0.3806
Non-zero	MSE	7.081	7.382	6.714
	RMSE	2.6611	2.717	2.591
	MAE	1.6828	1.7	1.641
	MAPE	0.353	0.355	0.3447
Zeros	MSE	0.5318	0.575	0.487
	RMSE	0.7292	0.758	0.689
	MAE	0.4292	0.461	0.41
	MAPE	0.4292	0.461	0.41

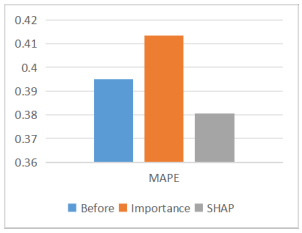
Table 9: SHAP vs feature importance based multi-modal feature trimming results



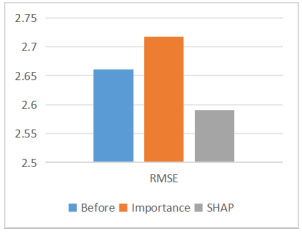
(a) RMSE (Overall)



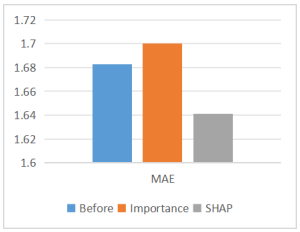
(b) MAE (Overall)



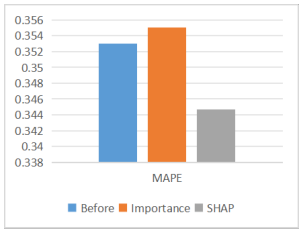
(c) MAPE (Overall)



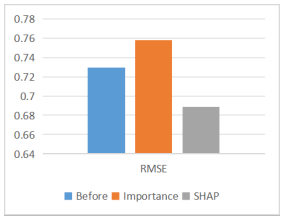
(d) RMSE (Non-Zeros)



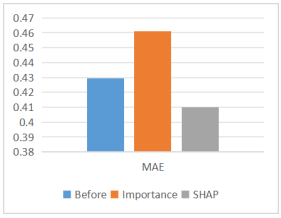
(e) MAE (Non-Zeros)



(f) MAPE (Non-Zeros)



(g) RMSE (Zeros)



(h) MAE (Zeros)

Figure 34: Results from Table 9 showing the improvement of SHAP compared to the original and as opposed to using feature importance

6 Conclusions and Future Work

This work has explored using a GNN decoder for an LSTM model and as a feature generator and extractor for other regression models, furthermore refining the final model through data partitioning and feature removal using the SHAP explainability technique. A case study using Lyft's Baywheels BSS in San Francisco was conducted to explore its feasibility. The introduction of GNN embedding-based clustering features, as well as the application PCA dimensionality reduction to the cosine similarity of the embedding, has resulted in a substantial improvement over the initial baseline models. However, further analysis, aided by indications from SHAP explainability, has resulted in further performance optimization by partitioning stations with a certain relevance and allowing the creation of better optimized models with a smaller number of features. The addition of GNN cluster features and SHAP-based trimming has resulted in a collective improvement of 16% compared to the initial baseline model (RMSE = 1.813 compared to 2.1572).

The progression made through this work showed the limitations in using Random Forest with increasingly complex features, leading to the final conclusion that XGBoost is the superior regression model for demand prediction. GNN-LSTM models performed less optimally compared to the regression model in most metrics, especially after training of the SHAP-trimmed models which outperformed the GNN-LSTM for zero-demand predictions.

For future work, more research into using GraphSAGE for a dynamic constantly evolving network, this neural network could be used to solve the data and concept drift issue due to its ability to handle new nodes in the network. Furthermore, other clustering methods could be applied using graph embedding in for GNN cluster features, this work was limited to KMeans where K = 4, different cluster sizes or other clustering algorithms, such as DBScan or hierarchical clustering, could provide more comprehensive details about the state of the network. Finally, further tests could be performed with more data partitions for SHAP-based feature trimming.

References

[1] Ralf Bill, Jörg Blankenbach, Martin Breunig, Jan-Henrik Haunert, Christian Heipke, Stefan Herle, Hans-Gerd Maas, Helmut Mayer, Liqui Meng, Franz Rottensteiner, Jochen Schiewe, Monika Sester, Uwe Sörgel, and Martin Werner. Geospatial information research: State of the art, case studies and future perspectives. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 90:349–389, 8 2022.

[2] Business Trends Spotlight. Geospatial analytics market: Future demand and top key players analysis — 2031, 12 2023.

[3] Aditi Sinha. Supply-demand gaps in real-time using geospatial data, 1 2020.

[4] Susan Shaheen, Stacey Guzman, and Hua Zhang. Bikesharing in europe, the americas, and asia. *Transportation Research Record*, pages 159–167, 1 2010.

[5] The meddin bike-sharing world map report 2022 edition, 2022.

[6] Andreas Nikiforiadis, Katerina Chrysostomou, and Georgia Aifadopoulou. Exploring travelers' characteristics affecting their intention to shift to bike-sharing systems due to a sophisticated mobile app. *Algorithms*, 12:264, 12 2019.

[7] Andreas Holzinger, Anna Saranti, Christoph Molnar, Przemyslaw Biecek, and Wojciech Samek. Explainable ai methods - a brief overview, 2022.

[8] Jung-Hoon Cho, Young-Hyun Seo, and Dong-Kyu Kim. Efficiency comparison of public bike-sharing repositioning strategies based on predicted demand patterns. *Transportation Research Record: Journal of the Transportation Research Board*, 2675:104–118, 11 2021.

[9] Lyft. Citi bike monthly operating reports, 1 2024.

[10] Lyft. Lyft bikes.

[11] Aliasghar Mehdizadeh Dastjerdi and Catherine Morency. Bike-sharing demand prediction at community level under covid-19 using deep learning. *Sensors*, 22:1060, 1 2022.

[12] Andreas Kaltenbrunner, Rodrigo Meza, Jens Grivolla, Joan Codina, and Rafael Banchs. Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system. *Pervasive and Mobile Computing*, 6:455–466, 8 2010.

[13] Young-Hyun Seo, Sangwon Yoon, Dong-Kyu Kim, Seung-Young Kho, and Jaemin Hwang. Predicting demand for a bike-sharing system with station activity based on random forest. *Proceedings of the Institution of Civil Engineers - Municipal Engineer*, 174:97–107, 6 2021.

[14] Yuebing Liang, Guan Huang, and Zhan Zhao. Cross-mode knowledge adaptation for bike sharing demand prediction using domain-adversarial graph neural networks. 11 2022.

[15] Ruiying Guo, Zhihan Jiang, Jingchun Huang, Jianrong Tao, Cheng Wang, Jonathan Li, and Longbiao Chen. Bikenet: Accurate bike demand prediction using graph neural networks for station rebalancing. pages 686–693. Institute of Electrical and Electronics Engineers Inc., 8 2019.

[16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. volume 13-17-August-2016, pages 1135–1144. Association for Computing Machinery, 8 2016.

[17] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. 5 2017.

[18] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. 3 2019.

[19] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. Interpretable machine learning – a brief history, state-of-the-art and challenges. 10 2020.

[20] Patrick Hall, Navdeep Gill, and Nicholas Schmidt. Proposed guidelines for the responsible use of explainable machine learning. 6 2019.

[21] Jaume Torres, Enrique Jiménez-Meroño, and Francesc Soriguera. Forecasting the usage of bike-sharing systems through machine learning techniques to foster sustainable urban mobility. *Sustainability (Switzerland)*, 16, 8 2024.

[22] Frank Ngeni, Boniphace Kutela, Tumlumbe Juliana Chengula, Cuthbert Ruseruka, Hannah Musau, Norris Novat, Debbie Aisiana Indah, and Sarah Kasomi. Prediction of bike-sharing station demand using explainable artificial intelligence. *Machine Learning with Applications*, 17:100582, 9 2024.

[23] Yajun Zhou, Lilei Wang, Rong Zhong, and Yulong Tan. A markov chain based demand prediction model for stations in bike sharing systems. *Mathematical Problems in Engineering*, 2018, 2018.

[24] Huthaifa I. Ashqar, Mohammed Elhenawy, Mohammed H. Almannaa, Ahmed Ghanem, Hesham A. Rakha, and Leanna House. Modeling bike availability in a bike-sharing system using machine learning. pages 374–378. IEEE, 6 2017.

[25] Abolfazl Maleki, Erfan Nejati, Amir Aghsami, and Fariborz Jolai. Developing a supervised learning-based simulation method as a decision support tool for rebalancing problems in bike-sharing systems. *Expert Systems with Applications*, 233, 12 2023.

[26] Evidently AI Team. What is concept drift in ml, and how to detect and address it.

[27] Meteostat. Meteostat: The weather's record keeper.

[28] Paula Linh Kramer. T035 · gnn-based molecular property prediction, 2022.

[29] Google. Google colab.

[30] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 9 2016.

[31] Petar Veličkovi´veličkovi´c, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lì, and Yoshua Bengio. Graph attention networks, 2 2018.

[32] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. 6 2017.

[33] Xing Wang and Alexander Vinel. Benchmarking graph neural networks on link prediction. 2 2021.

[34] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. volume 13-17-August-2016, pages 785–794. Association for Computing Machinery, 8 2016.

[35] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree, 2017.

[36] Nenny Anggraini, Syopiansyah Jaya Putra, Luh Kesuma Wardhani, Farid Dhiya Ul Arif, Nashrul Hakiem, and Imam Marzuki Shofi. A comparative analysis of random forest, xgboost, and lightgbm algorithms for emotion classification in reddit comments. *JURNAL TEKNIK INFORMATIKA*, 17:88–97, 5 2024.

[37] Amjad Abu-Rmileh. The multiple faces of feature importance in xgboost, 2 2019.