Ride Smartly: Hourly Bike-sharing Rental Demand Prediction

Yukai Wang, Yuwei Zhuo, Lizhen Qiu

***Abstract*—This project investigates hourly bike-sharing demand prediction through a data-driven modeling framework that integrates temporal, environmental, and historical demand features. We evaluate a range of supervised learning models, including linear regression, multilayer perceptron (MLP), random forest, and XGBoost, and show that nonlinear models consistently achieve superior predictive performance, with the best performance model MLP achieving a test R^2 value exceeding 0.96.**

To further improve accuracy, we refine the feature space by introducing lagged and rolling statistics to better capture short-term temporal dependencies, while removing redundant variables to enhance model stability. Building on these improvements, we propose a hybrid modeling framework that combines a baseline MLP with a residual LSTM module. This design decomposes the prediction task into a global demand estimation and a structured temporal correction, allowing the sequential model to focus on residual dynamics.

Experimental results demonstrate that the hybrid approach achieves consistent performance gains over strong baselines, particularly in capturing local peaks and short-term fluctuations. These findings highlight the importance of combining feature engineering with hybrid architectures for time-dependent demand prediction tasks.

***Index Terms*—bike-sharing, prediction, linear regression, MLP, random forest, XGBoost.**

I. Introduction

THIS project aims to predict hourly urban bike-sharing rental demand in order to support more efficient and proactive system management. A key challenge in this problem is that bike demand exhibits strong temporal dependencies, as well as interactions with exogenous factors such as weather conditions, holidays, and seasonal effects. To address this, we develop a high-precision predictive modeling framework at an hourly resolution, incorporating both time-series features and contextual variables. By systematically comparing and refining multiple supervised learning models, including linear models and nonlinear approaches, we aim to construct a robust predictive tool that enables system operators to anticipate demand fluctuations and dynamically allocate bikes and docking capacity. However, we we must acknowledge its geographic and temporal limitations: being restricted to the city of Washington D.C. over a two-year period may affect the model's generalizability to modern or geographically diverse contexts. Therefore, a more comprehensive model that can be applicable in general contexts is the future step.

The key motivation for this study is provided by the work of Pan et al.[1]. The article demonstrates the critical role of time-series dependencies in bike-sharing demand forecasting; their application of Long Short-Term Memory (LSTM) networks effectively validates that temporal patterns are central to accurate demand prediction. While their findings are insightful, their comparative analysis remains relatively narrow in algorithm selection. To address this limitation and further strengthen the modeling framework, our study applies a wider range of prediction methods to bike-sharing demand. The method design is informed by advances in general temporal sequence modeling, like the work on medical events by Farhan et al.[2]. Although applied to a different domain, this work highlights the value of representing structured, context-dependent temporal events in a machine-learnable form to capture latent patterns. In our project, this principle translates to using lagged demand, rolling statistics, and temporal indicators as structured representations of historical system states, enabling models to learn underlying demand dynamics.

II. Dataset Introduction

The *Bike Rental Data Set* we use comes from UCI Machine Learning Repository and is downloaded from Kaggle. Victor Augado's team reorganized the original datasets: bike rental information created by Capital Bikeshare, weather information created by freemeteo, and holiday schedule posted by District of Columbia Department of Human Resources. The combined dataset records multiple features of the bike rental situation in DC from 2011 to 2012.

It is a widely used dataset. however, since it's a combination of existing datasets, it was concerned that whether all the information in these dataset matches with each other, especially when the holiday information is not accessible.

The dataset is presented in .csv file, and explanation of the features are clearly stated in README file. In this study, we conducted a thorough preprocessing and quality check on the bicycle rental dataset. his included handling missing values introduced during feature construction, verifying data consistency, and aligning all variables at the hourly level. After verification, the dataset structure was complete, and no missing observations or null values were found. This was attributed to the excellent record mechanism of the original collection system.

Since the data only recorded bike rental information in Washington D.C. from 2011 to 2012, there are several inherent weaknesses regarding the model's generalizability and temporal relevance. First, the geographic specificity means the patterns identified are calibrated to the mid-Atlantic climate

and the specific urban infrastructure of Washington D.C.; therefore, the model may not accurately predict rental behavior in cities with different topographies or extreme weather variations. Second, the limited temporal window fails to account for long-term shifts in urban mobility. Over the past decade, the bike-sharing landscape has changed hugely because of the proliferation of electric bikes, changes in commuting habits following the COVID-19 pandemic, and so on. Consequently, a model trained on decade-old data may underestimate current demand or fail to capture the modern 'new normal' of micro-mobility.

III. Data Analysis

A. Data Processing

Since time-series features are critical for this task, we constructed additional variables such as lagged demand (e.g., demand from the previous hour) and short-term rolling averages to capture recent trends. These features help incorporate temporal dependencies that cannot be captured by static variables alone.

We then categorized the variables into numerical and categorical features. For categorical variables such as season, weather situation, and weekday, we applied one-hot encoding to make them suitable for machine learning models that require numerical inputs. For numerical variables, including both original features (e.g., temperature and humidity) and engineered features (e.g., lag and rolling statistics), we applied standardization when required, particularly for models sensitive to feature scaling such as linear regression and neural networks. At the same time, we preserved the original feature forms for tree-based models, which do not require normalization.

In terms of feature selection, we focused on identifying variables that are theoretically and empirically relevant to bike demand. The features we use is shown in Fig.1. Temporal features (hour, weekday, and seasonal indicators), environmental variables (weather-related factors), and historical demand features (lag and rolling values) were all included, as they capture different dimensions of the demand-generating process. We also ensured that redundant or less informative variables were minimized to avoid unnecessary model complexity.

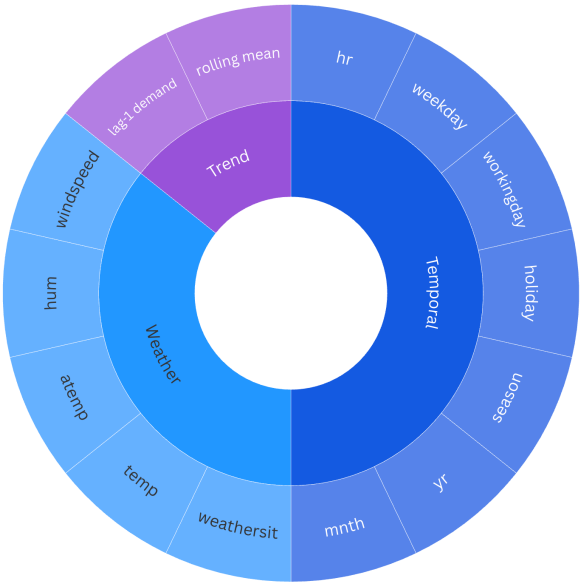
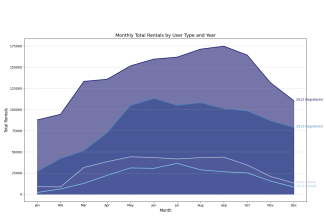
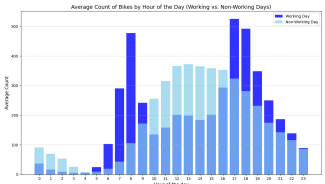


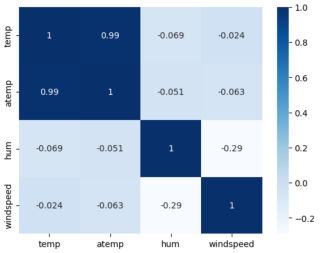
Fig. 1: Features selection.



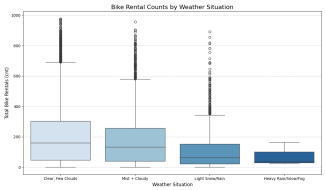
(a) Changes across years and months.



(b) Comparison of daily rental between workingday and non-workingday.



(c) The correlation between Weather features.



(d) Sensitivity of the weather situation of the day.

Fig. 2: Data Overview and Sensitivity Analysis.

B. Data Overview

Before implementing model analysis, we looked at different aspects of the data so that we have a better understanding of what the trend should be.

First, from the monthly rental trends (Fig.2a), we observe a clear seasonal pattern: bike usage increases steadily from winter to summer, peaks around mid-year, and then declines toward the end of the year. This indicates strong seasonality in demand, suggesting that temporal features such as month or season are important predictors.

Second, the hourly rental patterns further emphasize the importance of temporal dynamics. On working days, there are two pronounced peaks corresponding to commuting hours (morning and evening), while on non-working days, the demand is more evenly distributed throughout the daytime. This difference suggests that user behavior varies significantly depending on whether it is a working day, reinforcing the need to capture both temporal and categorical patterns in the model.

Third, the correlation heatmap reveals that *temp* and *atemp* are almost perfectly correlated (correlation $\approx$ 0.99), and *weathersit* and *hum* show moderate relationships with rental counts.

Finally, the boxplot of rentals across different weather situations shows a clear decreasing trend: rental counts are highest under clear weather conditions and drop significantly as weather worsens. This highlights the strong negative impact of adverse weather on bike usage.

Overall, these observations indicate that bike rental demand

is driven by a combination of seasonal trends, weather conditions, and daily behavioral patterns, which motivates the use of both feature engineering and time-dependent models.

IV. Model Analysis

On the modeling side, we first implemented multiple supervised learning approaches. These include baseline linear regression models, nonlinear models such as multilayer perceptron (MLP), and ensemble methods including random forest and gradient boosting. Each model was implemented in a consistent pipeline to ensure fair comparison across methods. We use python to implement this project and the code is developed in VS Code. For collaborative development and version control, we utilized GitHub, which allowed us to manage code integration and track changes across different stages of the project. The final implementation was executed in the same environment as the development phase.

We have also conducted preliminary experiments to evaluate model performance. The dataset was split into training and testing sets while preserving temporal ordering to avoid data leakage. We evaluated each model using standard performance metrics, including mean squared error (MSE) and coefficient of determination (R^2) for test-set.

A. Base Model description

Linear Regression

As the most classic regression models, we employed three linear regression variants as the first step: Simple Linear Regression (OLS), Ridge Regression, and Lasso Regression. Simple LR minimizes the ordinary least squares loss function:

$$\min_{\beta} \|y - X\beta\|_2^2.$$

Ridge regression adds an L2 regularization penalty to mitigate multicollinearity:

$$\min_{\beta} \|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2.$$

Lasso regression uses an L1 penalty to perform automatic feature selection:

$$\min_{\beta} \|y - X\beta\|_2^2 + \lambda\|\beta\|_1^1.$$

Standardization (zero mean, unit variance) was applied to all numerical features here before training. The model was trained using a closed-form least squares solver, and no overfitting was observed when comparing training and test set performance.

The R^2 for these three models is approximately 0.8, with Simple LR achieving the highest test R^2 of 0.86. This indicates that our data is relatively simple and unlikely to be overfit.

Multilayer Perceptron (MLP)

The MLP regressor is a feedforward neural network architecture designed for regression tasks. It utilizes fully connected hidden layers characterized by weighted sums and nonlinear activation functions to approximate the mapping from input features to bike-sharing demand. The model optimizes the loss function via backpropagation. To ensure stable gradient convergence and prevent features with larger magnitudes from dominating weight updates, all input data underwent standardization prior to training. In this study, the MLP employs the ReLU activation function and the Adam solver, with training conducted over a maximum of 500 iterations. Upon evaluation, the model yielded a training R^2 of 0.98 and a test R^2 of 0.96. The proximity of these values indicates that no significant overfitting occurred, and a successful balance was achieved in the bias-variance trade-off. Since the MLP does not assume linearity or a specific data distribution, its low-bias nature allows it to capture complex nonlinear and interactive patterns, providing highly accurate demand estimates ($R^2 \approx 0.96$).

Random Forest

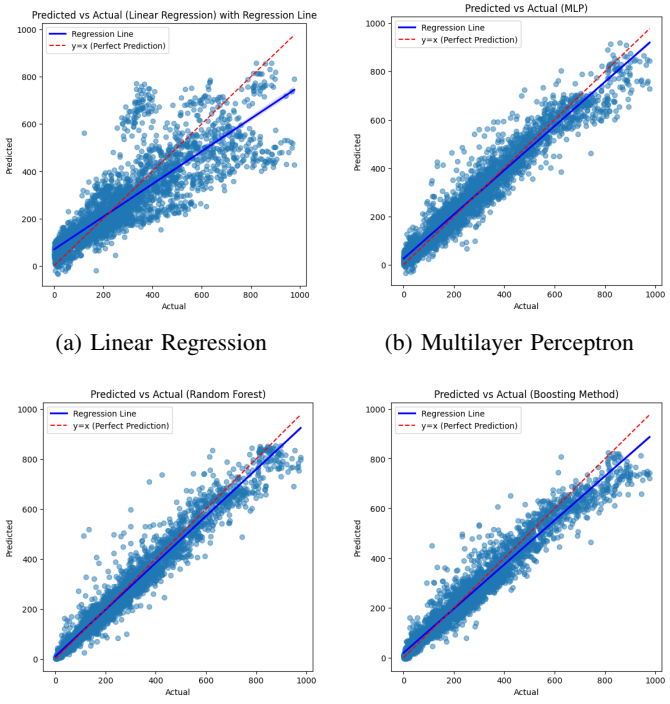
Random Forest is an ensemble learning method that constructs a set of independent decision trees during training and outputs the average prediction across all trees. Since Random Forest is based on tree-partitioning algorithms which are invariant to the scale of input features, no data scaling was implemented, allowing the model to maintain the original interpretability of the features. By allowing trees to grow until all leaf are pure, the model is able to capture intricate patterns. Although this configuration has a tendency to overfit the training data, the ensemble mechanism mitigates this by maintaining overall predictive stability. This non-parametric model does not assume linearity, normality, or independence of features, effectively balancing flexibility and stability better than a single decision tree. Random Forest demonstrated high accuracy on both the training and test sets, with a test $R^2 \approx 0.958$.

XGBoost

XGBoost (Extreme Gradient Boosting) is a gradient-boosted decision tree ensemble that builds trees sequentially to minimize the residual errors of preceding models. Similar to Random Forest, XGBoost is scale-invariant. Thus, no data scaling was implemented here too. The model optimizes a regularized supervised loss function to balance prediction accuracy and model complexity. We utilized a configuration with 300 estimators, a 0.05 learning rate, a maximum tree depth of 4, and a 0.8 subsample ratio.

Like Random Forest, XGBoost is a tree-based ensemble model, but it leverages sequential iterations and residual fitting to better capture nonlinear relationships. and its performance is highly dependent on meticulous parameter tuning. For this case, it shows the $R^2 \approx 0.953$.

The visualization of above base model analysis can be found in Table.I and Fig.3. These findings demonstrate that models capable of capturing nonlinear patterns—including multilayer perceptrons (MLP) and tree-based methods—consistently outperform simple linear regression. This result implies that the relationship between input features and bike-sharing demand is not strictly linear. Furthermore, incorporating time-series features such as lagged values and rolling averages substantially enhances prediction accuracy for all models tested.



(a) Linear Regression (b) Multilayer Perceptron

(c) Random Forest (d) XGBoost

Fig. 3: The comparison between the actual count and the predicted count acorss different predictive models.

Model	Testing MSE	Testing R^2
Linear Regression	6652.443	0.8628
Multilayer Perceptron	1921.8	0.9600
Random Forest	2045.287	0.9580
XGBoost	2291.7	0.9527

TABLE I: The mean square error and R^2 of different prediction models.

B. Advanced Model Implementation

1) Features Reselection: We first explored whether feature engineering could further improve model performance. To guide this process, we examined the importance (or significance level) of each feature across different models and visualized the results using a word cloud, as shown in Fig. 4.

From the figure, it is clear that although *hr* remains the dominant predictor, the engineered temporal features (such as lag and rolling statistics) also exhibit high importance. This suggests that capturing temporal dynamics is critical for modeling bike rental demand, and motivates us to further design features that better encode temporal patterns.

In addition, we can deduce some redundant features. Further analysis revealed that these two variables *temp* and *atemp* are highly correlated (Fig.2c shows that their correlation is 0.99), leading to redundancy in the feature space. To reduce multicollinearity and improve model stability, we removed *temp*, which proves to result in an improvement in model performance.

2) Hybrid Framework: Fig. 5 illustrates our advanced hybrid forecasting framework. Instead of using a single model to learn the entire bike-demand signal from scratch, we

Fig. 4: The significant level of the features.

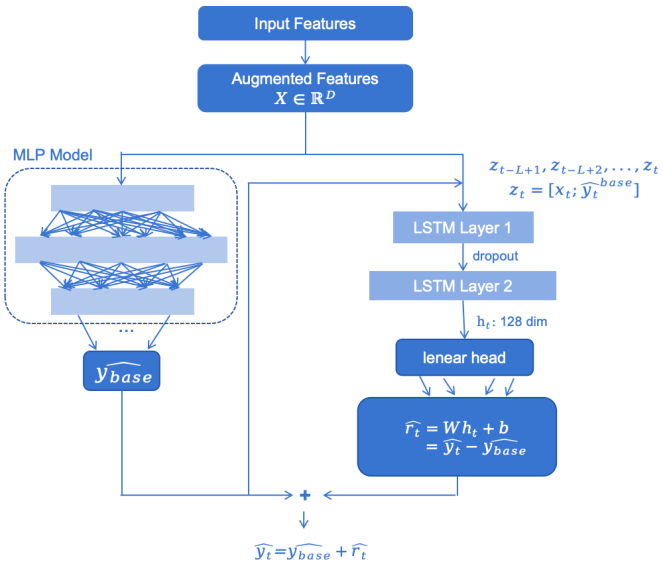


Fig. 5: Hybrid forecasting architecture combining a direct MLP baseline with a residual LSTM refinement module.

decompose the prediction into two parts: a direct baseline prediction and a sequential residual correction. Let $\mathbf{x}_t$ denote the engineered feature vector at hour t. The baseline multilayer perceptron (MLP) first predicts the main demand level as

$$\hat{y}_t^{\text{base}} = f_{\text{MLP}}(\mathbf{x}_t). \tag{1}$$

We then define the residual target as

$$r_t = y_t - \hat{y}_t^{\text{base}}, \tag{2}$$

where y_t is the true bike demand. To preserve both the original feature information and the baseline estimate, we construct an augmented input vector

$$\mathbf{z}_t = \left[\mathbf{x}_t; \hat{y}_t^{\text{base}}\right]. \tag{3}$$

Given a sequence length of $L = 72$, the residual branch receives a temporal window

$$\mathbf{Z}_t = (\mathbf{z}_{t-L+1}, \mathbf{z}_{t-L+2}, \ldots, \mathbf{z}_t), \tag{4}$$

which is processed by a two-layer LSTM. Using the last hidden representation $\mathbf{h}_t$, the residual head estimates

$$\hat{r}_t = \mathbf{W}\mathbf{h}_t + b, \tag{5}$$

and the final hybrid prediction is

$$\hat{y}_t = \hat{y}_t^{\text{base}} + \hat{r}_t. \tag{6}$$

This design is beneficial for two reasons. First, the MLP is well suited for learning the nonlinear mapping from weather, calendar, and tabular demand-history features to the overall demand level. Second, the LSTM only needs to model the structured temporal error left by the baseline model, rather than relearning the full signal. As a result, the residual formulation makes the sequential task easier and allows the model to focus on correcting local peaks, troughs, and short-term temporal dependence.

To further strengthen the feature representation, we added four new history-based variables: `Lag-24`, `Lag-168`, `Rolling-24-mean`, and `Rolling-24-std`. They are defined as

$$\mathrm{Lag24}_t = y_{t-24}, \qquad \mathrm{Lag168}_t = y_{t-168}, \tag{7}$$

$$\mathrm{RollMean24}_t = \frac{1}{24}\sum_{i=1}^{24} y_{t-i}, \tag{8}$$

$$\mathrm{RollStd24}_t = \mathrm{Std}(y_{t-24}, y_{t-23}, \ldots, y_{t-1}). \tag{9}$$

These new variables complement the original short-term lag features and provide explicit information about daily and weekly seasonality. In particular, `Lag-24` captures the repeating hourly pattern from the previous day, while `Lag-168` reflects weekly recurrence. The 24-hour rolling mean summarizes the recent local demand level, and the 24-hour rolling standard deviation describes short-term volatility. Together, these added features make the baseline predictor more informative and reduce the amount of temporal structure that must be recovered by the residual LSTM.

The training procedure follows a chronological forecasting pipeline. We first sort the observations by time and split the dataset into training and held-out segments without shuffling. Categorical features are one-hot encoded and numerical features are standardized using statistics fitted on the training set only, which avoids information leakage. Following our implementation, the baseline MLP is trained on centered targets,

$$\tilde{y}_t = y_t - \bar{y}_{\mathrm{train}}, \tag{10}$$

and the mean is added back during inference. After obtaining the baseline predictions, we compute residual targets and build augmented sequences for the LSTM. In our implementation, the residual model uses sequence length 72, hidden dimension 128, two LSTM layers, and dropout 0.1.

For the residual branch, the optimization objective is the mean squared error (MSE),

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{N}\sum_{t=1}^{N}(r_t - \hat{r}_t)^2 . \tag{11}$$

We use MSE because it penalizes large residual errors more heavily, which is desirable in demand forecasting where peak-hour mistakes are especially costly. The residual LSTM is trained with Adam and weight decay, and a ReduceLROnPlateau scheduler is applied so that the learning rate decreases when the held-out error stops improving. From the training curves, the training MSE decreases steadily while the held-out MSE remains in a relatively narrow range, suggesting that the baseline model already explains most of the global demand pattern and the residual branch mainly provides finer corrections.

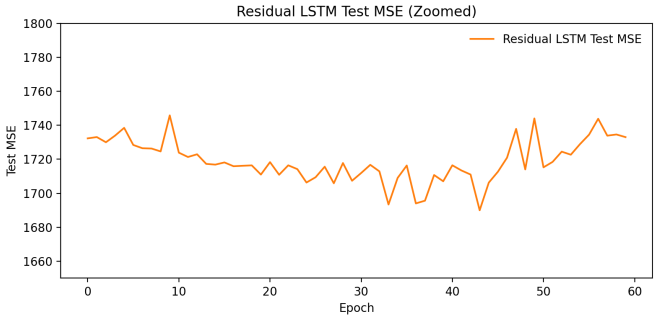


Fig. 6: Test Loss Curve

The test curve Fig. 6 further shows a clear training pattern: from Epochs 1 to 44, the test MSE generally decreases, indicating that the residual LSTM gradually learns useful temporal corrections on top of the baseline predictor. After around Epoch 44, however, the test MSE starts to fluctuate upward, which suggests the signal of overfitting. Therefore, the best generalization performance is achieved near Epoch 44, and later training mainly improves the fit to the training data rather than the generalization ability.

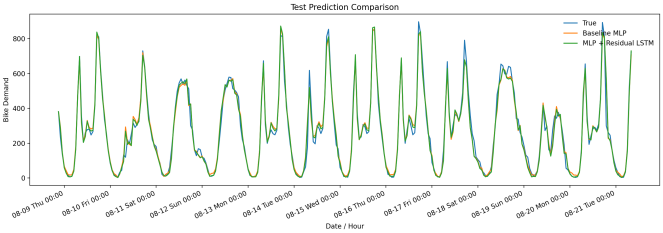


Fig. 7: Comparison of true demand, baseline MLP prediction, and hybrid prediction on the test set.

Fig. 7 shows the final prediction results on the test set. The baseline MLP already captures the overall daily demand cycle well, which indicates that the engineered tabular features are highly informative. The hybrid model further improves the alignment between prediction and ground truth, especially around local peaks, valleys, and rapid transitions. Quantitatively, the hybrid framework reduces the test RMSE from 41.72 to 41.11 and lowers the MAE from 27.68 to 26.60, while the R^2 score improves to 0.9650. Although the gain is moderate, it is consistent and meaningful because the baseline model is already strong. The prediction curves show that the residual LSTM successfully learns structured errors left by the direct predictor and produces a forecast that is generally closer to the true demand trajectory. Some mismatch still remains at the sharpest spikes, indicating that extremely abrupt surges are still difficult to model perfectly, but overall the hybrid architecture yields a smoother and more accurate forecast.

V. Conclusion

Overall, the forecasting performance of our current model is already very strong. The baseline model captures most of the major demand patterns in the data, and the improved hybrid model only provides a modest additional gain. In particular, the residual LSTM mainly improves prediction around local valleys and peaks, while the overall prediction trend remains very similar to the baseline. We also explored several other extensions, including predicting `casual` and `registered` users separately and using XGBoost for residual modeling, but none of them outperformed the baseline framework. This suggests that the current model and feature set have already captured most of the predictable structure in this benchmark dataset.

From the prediction curves, we also observed several useful patterns. Among the engineered features, short-term historical information appears to be especially important, and `lag_1` seems to have a strong influence on the forecast. This indicates that the most recent demand level carries substantial predictive information for the next hour. Such a finding is practically meaningful, since it suggests that real-time or rolling prediction systems may benefit significantly from incorporating the latest observed demand as soon as it becomes available.

Our study is still constrained by the scope of a single-city dataset collected in Washington, D.C. from January 1, 2011 to December 31, 2012. This naturally limits direct generalization to other cities, and the age of the dataset may also reduce its temporal relevance. Still, prior work has shown that transferring bike-sharing demand models across cities is a meaningful practical problem [3], and transfer learning provides a natural framework for reusing learned representations when the source and target domains differ. Therefore, a promising next step is to freeze part of the current network and fine-tune the remaining layers on a smaller amount of local data from a new city, which would improve adaptability while reducing additional training cost.

VI. Team Participation

Project Task	Responsible Member
Data search & collection	Everyone
Data cleaning	Lizhen Qiu
Feature selection	Yuwei Zhuo
Base model implementation	Yukai Wang & Yuwei Zhuo
Base model evaluation	Lizhen Qiu
Advanced model construction	Everyone
Final model evaluation	Yukai Wang

References

[1] Farhan, Wael, et al, *A predictive model for medical events based on contextual embedding of temporal sequences.* JMIR medical informatics 4.4 (2016): e39.

[2] Pan Y., Zheng R.C., Zhang J., Yao X, *Predicting bike sharing demand using recurrent neural networks*. Procedia Computer Science, 147 (2019) 562-566.

[3] S. Guidon, D. J. Reck, and K. W. Axhausen, "Expanding a(n) (electric) bicycle-sharing system to a new city: Prediction of demand with spatial regression and random forests," *Journal of Transport Geography*, vol. 84, p. 102692, 2020, doi: 10.1016/j.jtrangeo.2020.102692.