COMP3901 Research Report

Predicting Station-level Hourly Bike-Sharing Demand Using XGBoost and LSTM

Photo Credit: Jeff Greenberg, Getty Images

Michael Gysel z5251938

Table of Contents

1. Introduction 3

2. Literature Review 4

3. Methods 6

3.1 XGBoost 6

3.2 Long Short-Term Memory (LSTM) 7

4. Data Sets 9

5. Data Pre-processing Methods 9

6. Data Analysis 10

6.1 Evaluation Metrics 10

6.2 XGBoost Model Development 11

6.3 LSTM Model Development 14

7. Results and Comparison 18

8. Conclusions, Limitations, and Future Research Directions 21

Source Code 23

References 23

1. Introduction

Over the next 30 years, the world's population is expected to increase by 2 billion people [1]. As global populations continue to urbanize, this will mean more congestion, noise, pollution, and greenhouse gas emissions, largely due to increased motorized vehicle usage [2, 3]. Bike-sharing systems (BSSs) pose one possible solution to these growing problems by reducing motorized vehicle usage.

BSSs are a shared transport service that allow for short-term bicycle rental available at unattended urban locations [4]. BSSs were first developed in the 1960s, but have more recently exploded in usage due to the commonplace use of technology to track both bikes and users [3]. Currently, over 1,000 cities globally utilise BSSs [5]. There are two main types of BSSs: docked and dockless. In docked BSSs, bikes are tethered to stations where bikes are picked up and dropped off; in dockless BSSs, bikes can be picked up and dropped off anywhere within a particular geographic area. This study will focus on docked BSSs.

BSSs have been shown to significantly reduce vehicle miles traveled and the resulting congestion, noise, pollution, and greenhouse gas emissions [6]. Furthermore, BSSs even benefit public health due to the increased physical activity of its users [7]. While most BSSs reduce traffic congestion, poorly balanced bike fleets have actually led to an increase in traffic congestion in some use cases. This poor fleet balancing reduces BSS use due to mismatched supply and demand and requires staff to manually relocate bikes using trucks [8]. Thus, accurate bike-sharing demand is critical for the sustainable use of BSSs.

Recently, machine learning models have seen wide use in solving spatio-temporal transportation problems. Specifically, the use of Gradient Boosting Machine (GBM) and Long Short-Term Memory (LSTM) models have shown great promise in predicting hourly station-level bike demand in BSSs [9, 10]. This report proposes the use of the GBM model XGBoost and Long Short-Term Memory to predict hourly station-level bike demand using the New York City Citi Bike bike-sharing demand data and National Oceanic and Atmospheric Administration (NOAA) meteorological data.

The rest of this report is organized as follows. Section 2 discusses related work on predicting bike-sharing demand. Section 3 provides a background on the XGBoost and LSTM models used to predict bike-sharing demand in this report. Section 4 discusses the

Citi Bike and NOAA data sets and Section 5 discusses the data pre-processing methods used. In Section 6, a data analysis is presented to predict station-level hourly bike-sharing demand using XGBoost and LSTM and Section 7 compares the results. Finally, Section 8 presents a conclusion, addresses limitations of this research, and proposes future research directions.

2. Literature Review

Over the past decade, the use of machine learning models to predict bike-sharing demand has been an area of significant research interest. These studies have used a wide variety of machine learning models, including Linear Regression (LR), Random Forest (RF), Gradient Boosting Machine (GBM), Gated Recurrent Units (GRU), Long Short-Term Memory (LSTM), and Geometric Convolutional Neural Networks (GCNN). Most of these studies used spatiotemporal, meteorological, and time-lagged factors, such as bike demand an hour beforehand; moreover, some also utilised factors such as taxi demand and attributes of station neighbors. Most of these studies focused on station-level demand, though some focused on neighborhood or even citywide demand.

As shown in Table 1, Ashqar et. al. modeled the station-level demand at San Francisco Bay Area Bike Share stations using RF, Least-Squares Boosting (LSBoost) and Partial Least-Squares Regression (PLSR). The results showed that RF and LSBoost had the lowest prediction error and that station neighbors and time-lagged factors were significant predictors [11]. Choi and Han used bike sharing data from Daejeon, Republic of Korea to compare the station-level demand forecasting models for RF, XGBoost, LSTM, and GRU. Using only time-lagged factors to predict future demand, this study concluded that the deep learning models, LSTM and GRU, outperformed the machine learning models, RF and XGBoost [9]. Yang et al. utilised graph-based attributes between stations, applying them as predictors in XGBoost, MLP, and LSTM models, in order to predict station level hourly demand in New York City and Chicago, United States. When utilising these graph-based attributes, the LSTM model outperformed the others; when excluding the graph-based attributes, XGBoost outperformed the others [10]. Lin et al. proposed four GCNN models to predict station-level hourly demand using the Citi Bike data set from New York City, United States. Varying adjacency matrices used in each of the four GCNN models, this study uncovered hidden heterogeneous pairwise correlations between stations [12].

Machine learning models have not only been applied to station-level analyses, but also to neighborhood and city-wide demand predictions. Singhvi et al. utilised taxi usage

demand along with spatio-temporal and meteorological factors in LR models in order to predict morning rush-hour bike demand using the New York City Citi Bike system data set. This study concluded that analysing neighborhood-level demand as opposed to station-level demand can improve the accuracy of predictions [13]. Sathishkumar et al. used meteorological, temporal, and time-lagged factors in order to predict city-level hourly demand in Seoul, Republic of Korea. This study concluded the GBM models outperformed the LR models [14].

Table 1: Use of Machine Learning to Predict Bike-Sharing Demand

Authors	Article	Demand Level	Variables	Machine Learning Models Used
Ashqar et al.	Modeling Bike Availability in a Bike-Sharing System Using Machine Learning	Station	- Meteorological - Temporal	- Random Forest - Least-Squares Boosting - Partial Least Squares Regression
Choi and Han	The Empirical Evaluation of Models Predicting Bike Sharing Demand	Station	- Meteorological - Temporal	- Random Forest - Gradient Boosting Machine (XGBoost) - Long Short-Term Memory - Gated Recurrent Units
Yang et al.	Using graph structural information about flows to enhance short-term demand prediction in bike-sharing systems	Groups of Stations	- Meteorological - Temporal	- Gradient Boosting Machine (XGBoost) - Multilayer Perceptron - Long Short-Term Memory
Lin et al.	Predicting station-level bike-sharing demands using graph convolutional neural network	Station	- Spatial - Temporal	- Geometric Convolutional Neural Network - Gradient Boosting Machine (XGBoost) - Long Short-Term Memory
Singhvi et al.	Predicting Bike Usage for New York City's Bike Sharing System	Neighborhood	- Meteorological - Spatial - Taxi Usage	- Linear regression
Sathishkumar et al.	Using data mining techniques for bike sharing demand prediction in metropolitan city	City	- Meteorological - Temporal	- Linear regression - Gradient Boosting Machine - Support Vector Machine - Boosted Trees - Extreme Gradient Boosting Trees

This report contributes to the literature by applying XGBoost and LSTM to the problem of predicting station-level hourly bike demand using New York City Citi Bike bike sharing and meteorological data.

3. Methods

In this section, we will briefly describe the two machine learning models used in this report: XGBoost and LSTM.

3.1. XGBoost

XGBoost is an implementation of Gradient Boosting Machine (GBM) that uses an ensemble of decision trees in order to predict a target. GBM models can be described in three main components: a loss function, weak learner, and additive model. The loss function is chosen based on the prediction being made; in the case of regression problems such as station-level hourly bike-sharing demand, Root Mean Square Error (RMSE) is commonly used. Secondly, GBM requires the use of weak learners; in the case of XGBoost, the weak learners are Decision Trees. Lastly, these weak learners are combined in an additive model, such that each subsequent weak learner corrects the errors of the previous ensemble. In XGBoost, a Decision Tree is initially created that reduces the overall loss function. Next, subsequent Decision Trees are added to the model, with each subsequent Decision Tree further reducing the overall loss function. This process can be seen in Figure 1 [15].

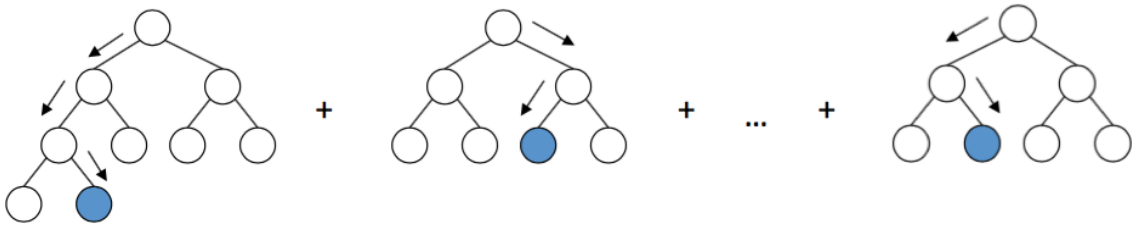

Figure 1: Model of Gradient Boosting Machine [15]

3.2. Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a form of Recurrent Neural Network (RNN) that is particularly useful when processing sequences of values. As shown in Figure 2, RNNs allow past outputs to be used as inputs, passing information from one step of the network to the subsequent steps of the network. As a result, RNNs can handle variable-length inputs, track long-term dependencies, and maintain information about the sequence's order [16]; thus, they are particularly useful in predicting time-series bike demand.

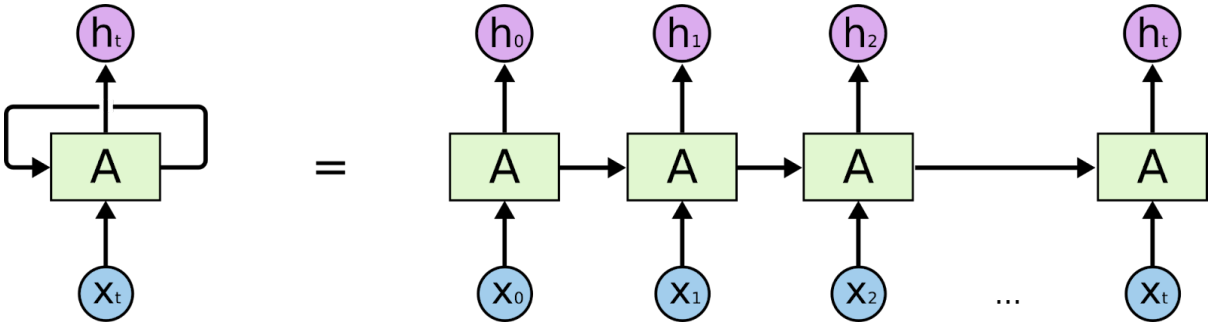

Figure 2: Model of Recurrent Neural Network [17]

Neural networks train through a process known as gradient descent, where network weights are iteratively chosen to minimize the total loss of the network, a process that can be visualised in Figure 3 [18]. In order to determine whether increasing or decreasing a particular weight decreases error, the derivative of the loss with respect to the weight is calculated. Thus, if the derivative is positive, then increasing the weight would increase the error; if the derivative is negative, increasing the weight would decrease the error. In order to train the entire network, we must backpropagate the error from the output nodes to the input nodes, multiplying local gradients using the chain rule. Because RNNs process sequential data, they backpropagate at each timestep and across all timesteps, a process known as backpropagation through time [16].

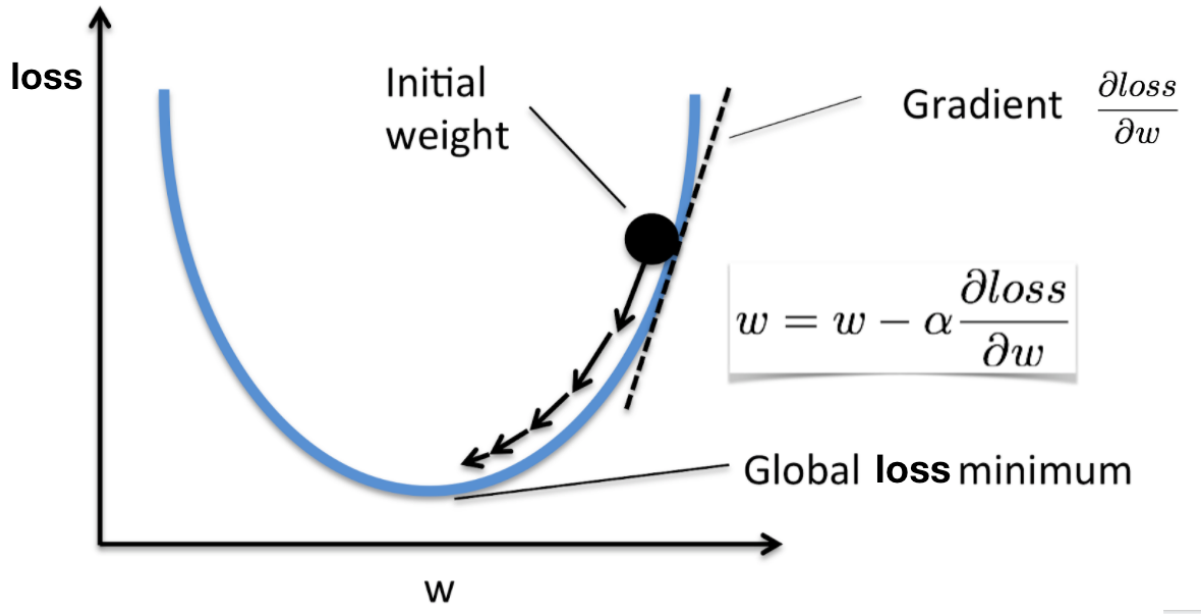

Figure 3: Gradient Descent [18]

Traditional RNNs suffer from the exploding and vanishing gradient problems. If we multiply very large gradients together during backpropagation, the gradient explodes in size. This exploding gradient causes large updates in network weights during training, resulting in an unstable network. If we multiply very small gradients together during backpropagation, the gradient vanishes in size. This vanishing gradient causes small updates in network weights during training, resulting in a network that has difficulty training [16].

LSTM solves the exploding and vanishing gradient problems through the use of gates, which better control the flow of information throughout the network, allowing each cell to maintain a state separate from what is outputted to the network. As can be seen in Figure 4, LSTMs use four types of gates: an update gate which selectively updates the cell state, a relevance gate which stores relevant information from the current input to the cell, a forget gate which drops previous information, and an output gate which outputs information from the cell [17, 19]. As a result, LSTMs are able to backpropagate through time with an uninterrupted gradient flow, solving the exploding and vanishing gradient problems [16].

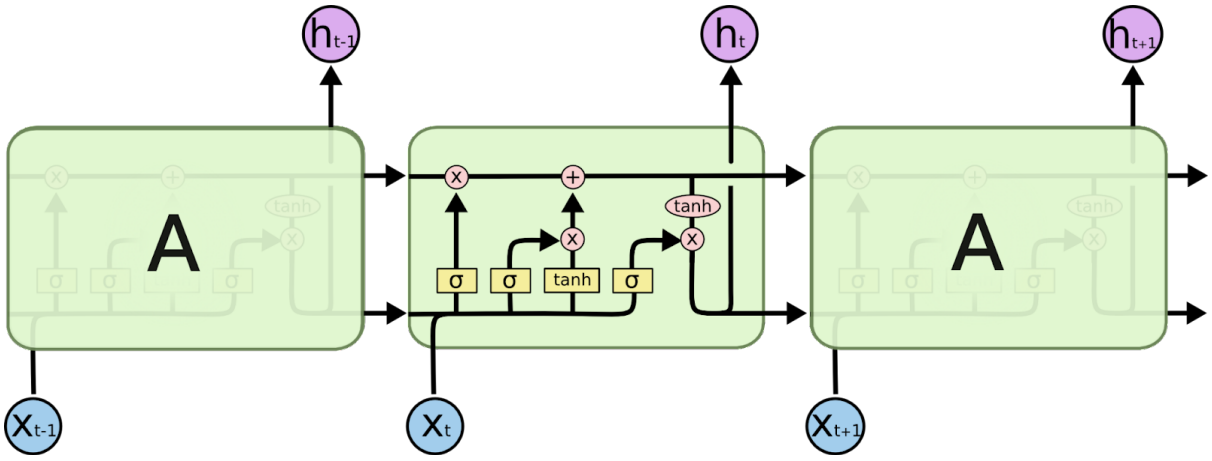

Figure 4: Model of LSTM Cell Gates [17]

4. Data Sets

This study used two data sets: New York City Citi Bike bike-sharing demand data and National Oceanic and Atmospheric Administration (NOAA) meteorological data, from May 1st, 2019 to July 31st, 2019. Citi Bike is a BSS located in New York City, United States, and the data set includes information on Citi Bike trips made at the 53 Citi Bike stations. More specifically, the Citi Bike data set includes station ID, station latitude and longitude, start time and date, stop time and date, start and end station names, station latitudes and longitudes, bike ID, user type, gender, and year of birth [20]. The NOAA meteorological data set includes meteorological data for a weather station located in Central Park in New York City, United States. More specifically, the NOAA data set includes daily minimum and maximum temperatures, precipitation, snowfall, snow depth, average wind speed, and the presence of specific weather types, such as fog, heavy fog, thunder, ice pellets, and numerous others [21].

5. Data Pre-processing Methods

For each Citi Bike bike station, hourly bike demand was aggregated, resulting in a data set that includes the total number of trips for each station at each hourly timestep between May 1st, 2019 and July 31st, 2019. Because the XGBoost model does not process sequences of data, each record was supplemented with time-lagged data, specifically the station-level bike demand 1 to 6 hours prior, 24 hours prior, 48 hours prior, and one week prior to the record's timestep. This data set was then combined with the

NOAA meteorological data, so that each record contained meteorological information on that corresponding day.

From this data set, several features were engineered. Date and time features were extracted from the hourly timesteps; specifically the day of the week, hour of the day, and month. Because each of these were cyclical features, the sine and cosine of each were calculated. Additionally, whether the timestep landed on a weekend or weekday was extracted. Lastly, it was observed that large peaks in bike demand occurred between 7 a.m. and 9 a.m. and between 4 p.m. and 7 p.m. As such, two features were added denoting if each record occurred in the a.m. peak period or p.m. peak period. The features resulting from the data pre-processing can be seen in Table 2.

Table 2: Pre-processed Data Set

Feature Type	Features
Hourly bike demand	num_trips, start_datetime, start_station_id, start_station_longitude, start_station_latitude
Time-lagged factors	num_trips_1hr, num_trips_2hr, num_trips_3hr, num_trips_4hr, num_trips_5hr, num_trips_6hr, num_trips_24hr, num_trips_48hr, num_trips_week
Temporal factors	day_of_week, day_of_month, month, hour, is_weekend, hour_sin, hour_cos, day_of_week_sin, day_of_week_cos, day_of_month_sin, day_of_month_cos, month_sin, month_cos, is_am_peak, is_pm_peak
Meteorological factors	avg_daily_wind_speed, precipitation, temp_max, temp_min, fog, heavy_fog, thunder, haze

The final pre-processed data set contained 108,120 records corresponding to the 53 bike stations and 2,040 hourly intervals between May 8th, 2019 and July 31st, 2019.

6. Data Analysis

6.1. Evaluation Metrics

The XGBoost and LSTM models were built and trained using Root Mean Square Error (RMSE) as the evaluation metric. RMSE was chosen as the training evaluation metric because it estimates the average standard deviation of an observed value compared to our model's prediction [22]. Thus, it allows us to estimate the difference we can expect

between our prediction and the actual bike demand, in terms of the number of bikes per hourly interval per station.

The XGBoost and LSTM models were then compared using RMSE, Mean Absolute Error (MAE), and R-Squared (R^2). Mean Absolute Percentage Error (MAPE) was not considered due to the high rate of records with a station-level hourly bike demand of 0, which would result in numerous undefined values when calculating MAPE.

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(x_i - \widehat{x_i}\right)^2}$$

where n is the number of data points and predictions, x_i is an observation and $\widehat{x_i}$ is a prediction.

$$MAE = \frac{1}{n}\Sigma_{i=1}^{n}\left|x_i - \widehat{x_i}\right|$$

where n is the number of data points and predictions, x_i is an observation and $\widehat{x_i}$ is a prediction.

$$R^2 = \frac{\Sigma_{i=1}^{n}\left(\hat{x}_i - \bar{x}\right)^2}{\Sigma_{i=1}^{n}\left(x_i - \bar{x}\right)^2}$$

where n is the number of data points and predictions, x_i is an observation, $\widehat{x_i}$ is a prediction, and $\hat{x}$ is the average of all observations.

6.2. XGBoost Model Development

6.2.1. Input Data

XGBoost was used to predict station-level hourly bike demand. The pre-processed data set was split into training and testing data sets which contained the first 70% and last 30% of the records, respectively. Because the domain of bike demand prediction requires predicting bike demand in the future based on past data, the data set was not shuffled when creating the training and testing data sets. Per Figure 5 below, a sensitivity analysis was performed to determine the optimal train-test split. 70% was chosen as the train-test split because it most reduced RMSE of the XGBoost models and reduces the likelihood of overfitting relative to higher train-test splits. More specifically, using the 70% train-test split reduced the RMSE on the validation set to 1.285.

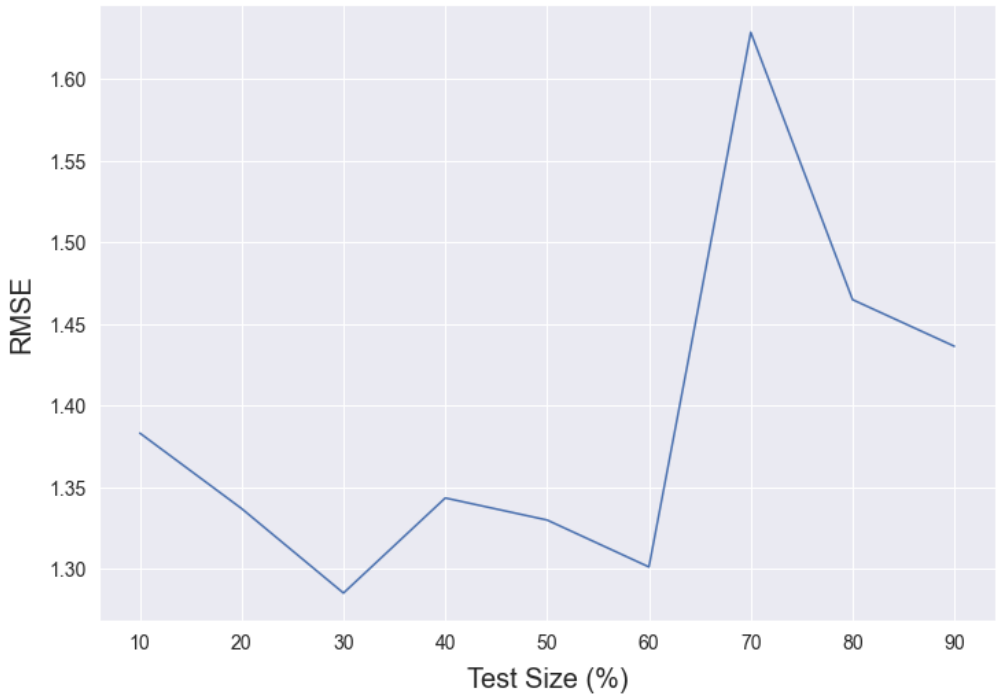

Figure 5: Test Size vs. RMSE

The training data set contains 75,684 records, corresponding to each hourly interval from May 8th, 2019 at 12:00 a.m. to July 6th, 2019 at 11:00 a.m. for the 53 bike stations. The test data set contains 32,346 records, corresponding to each hourly interval from July 6th, 2019 at 12:00 p.m. to July 31st, 2019 at 11:00 p.m.

6.2.2. Model Training

In machine learning, hyperparameters are parameters that are set prior to the learning process and that directly impact how well the model trains [23]. During training, a variety of hyperparameters are set and their performance is evaluated using k-fold cross validation. In k-fold cross validation, the training data set is split into k groups. One group is used as a test data set and the remaining groups used as a training data set. The model is then fit to the training data set and evaluated on the test data set; this process is repeated k times, using each of the k groups as a test data set one time [24]. Per Figure 6 below, a sensitivity analysis was performed to determine the optimal value of k. During the hyperparameter selection process a k value of 4 was chosen due to computational limitations; however, a k value of 14 was chosen when training the final model in order to most reduce the RMSE. The k-value of 4 resulted in an RMSE of 1.234 on the validation data set and the k-value of 14 resulted in an RMSE of 1.193 on the validation data set.

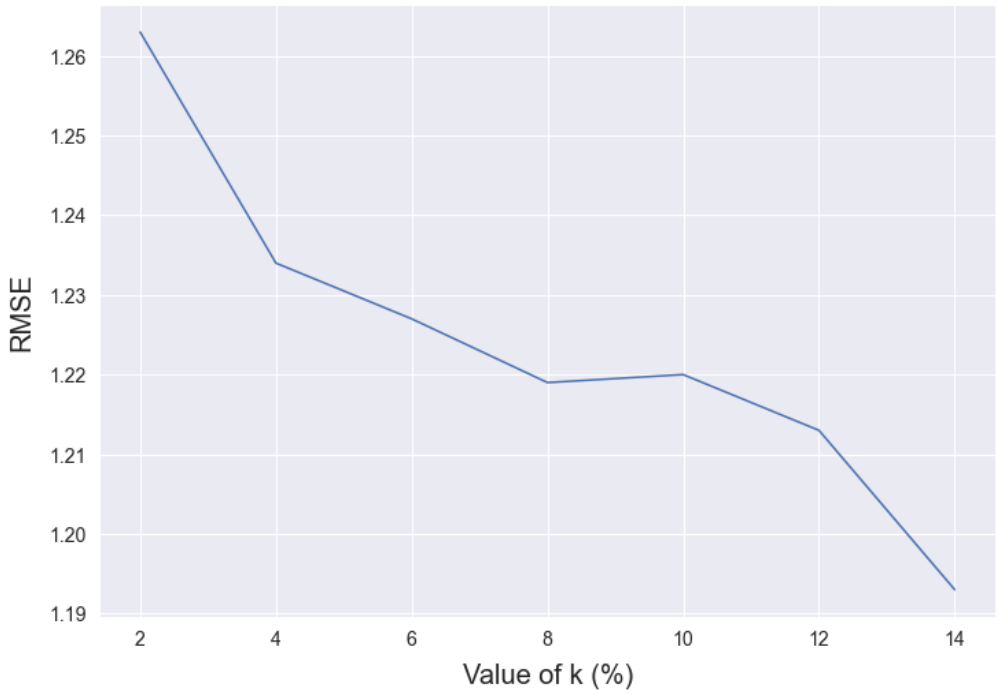

Figure 6: Value of k vs. RMSE

Per Table 3, the following hyperparameters were tuned for the XGBoost model: the maximum depth of each tree (max_depth), the minimum sum of instance weight needed in each child (min_child_weight), the number of gradient boosted trees (n_estimators), the step size shrinkage used in each boosting step (learning_rate), the subsample ratio of training instances (subsample), and the subsample ratio of columns when constructing each tree (colsample_bytree) [25]. Initially, max_depth and min_child_weight were trained together because they both add constraints to the structure of each decision tree and can be used to prevent overfitting. Next, n_estimators and learning_rate were trained together because they interact during the training process; moreover, the smaller the learning_rate, the fewer corrections are made to each tree added to the model, thereby resulting in more trees added to the model. Lastly, subsample and colsample_bytree were trained in order to limit the records and columns used in the model, thereby preventing overfitting. As shown in Table 3, the model that most reduced the RMSE on the validation data set was built using max_depth of 6, min_child_weight of 7, n_estimators of 500, learning_rate of 0.01, subsample of 0.8, and colsample_bytree of 0.5; this model resulted in a 1.170 RMSE on the validation data set.

Table 3: XGBoost Model Training			
Hyperparameter	**Description**	**Values Tested**	**(Value Chosen: RMSE on Validation Data Set**
max_depth	Maximum depth of each tree	{3, 6, 9}	(6: 1.297)
min_child_weight	Minimum sum of instance weight required in each child	{1, 3, 5, 7}	(7: 1.297)
n_estimators	Number of gradient boosted trees	{10, 25, 50, 100, 200, 400, 500}	(500: 1.190)
learning_rate	Step size shrinkage used in each boosting step	{0.0001, 0.001, 0.01, 0.1, 0.2, 0.3}	(0.01: 1.190)
subsample	Subsample ratio of training instances	{0.5, 0.6, 0.7, 0.8, 0.9, 1.0}	(0.8: 1.184)
colsample_bytree	Subsample ratio of columns when constructing each tree	{0.5, 0.6, 0.7, 0.8, 0.9, 1.0}	(0.5: 1.170)

6.3. LSTM Model Development

6.3.1. Model Structure

LSTM was also used to predict station-level hourly bike demand. One individual LSTM model was trained for the entire data set, built with an input layer of 1,484 neurons, corresponding to the 28 features at each of the 53 bike stations, and with an output layer of 53 neurons, corresponding to the 53 bike stations for which the model predicts hourly bike demand.

6.3.2. Input Data

Because LSTM models require 3-dimensional input data - corresponding to the number of samples by the number of timesteps by the number of features - the data was restructured. The number of samples and timesteps were dependent on the number of timesteps chosen during the training process; as is discussed in Section 6.3.3 below, 72 timesteps were selected during the model training process. The input data contained 1,484 features, corresponding to the 28 features for the 53 bike stations. Note that because LSTM is trained with time series data, the station-level demand 1 to 6 hours prior, 24 hours prior, 48 hours prior, and one week prior were removed from the data set.

Just as in the XGBoost training process, a sensitivity analysis was conducted to determine the optimal train-test split, which is shown in Figure 7. 70% was chosen as the train-test split because it significantly reduces RMSE while reducing the likelihood of

overfitting relative to higher train-test splits. More specifically, using the 70% train-test split reduced the RMSE on the validation set to 1.991.

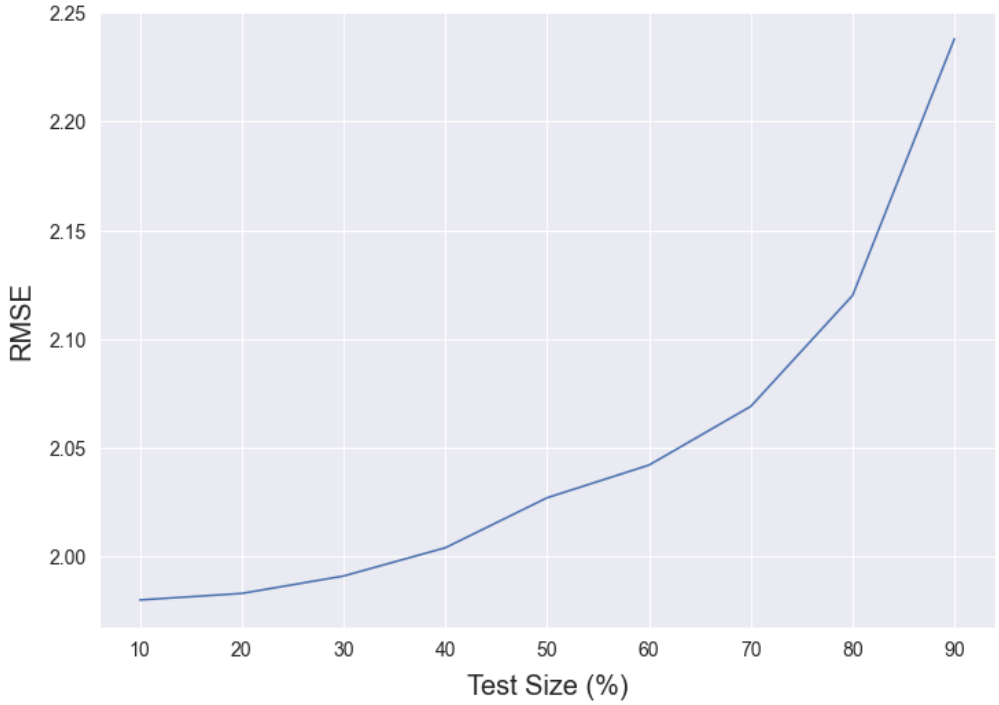

Figure 7: Test Size vs. RMSE

Thus, the resulting training data set is size 1355 by 72 by 1484, corresponding to the number of samples by the number of timesteps by the number of input variables between May 8th, 2019 at 12:00 a.m. and July 6th, 2019 at 11:00 a.m. The resulting test data set is size 539 by 72 by 1484, corresponding to the period between July 6th, 2019 at 12:00 p.m. and July 31st, 2019 at 11:00 p.m.

6.3.3. Model Training

The LSTM model was trained to determine the optimal value of k when using cross-fold validation, data scaler, number of timesteps in the input data, number of hidden layer nodes in the network, number of layers in the network, use of bidirectional or unidirectional LSTM layers, activation function, optimizer, and number of epochs.

Because the LSTM model was trained using cross-fold validation, a sensitivity analysis was performed to determine the optimal value of k, which can be seen in Figure 8. A k-value of 4 was chosen because it substantially reduces RMSE while also reducing

computation time. More specifically, using a k-value of 4 resulted in an RMSE of 1.987 on the validation data set.

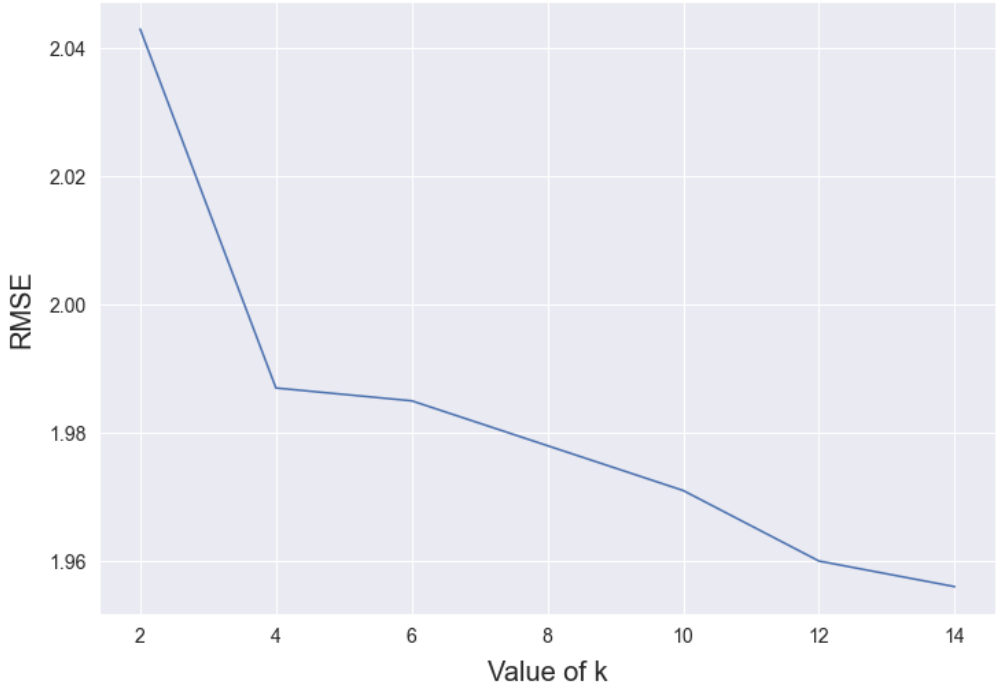

Figure 8: Value of k vs. RMSE

LSTM models often benefit from normalisation or standardisation because unscaled input variables may slow model convergence and result in an unstable training process; thus, all numeric columns were normalized or standardized using sklearn's RobustScaler, MinMaxScaler, StandardScaler, and PowerTransformer [26]. Normalisation refers to rescaling the data between a range of 0 to 1 and standardisation refers to rescaling the data to have a mean of 0 and a standard deviation of 1. RobustScaler removes the mean and scales the data according to the interquartile range, StandardScaler removes the mean and scales the data to unit variance, MinMaxScaler rescales the data such that all feature values are in the range 0 to 1, and PowerTransformer applies a power transformation in order to make the data more Gaussian [27]. As is shown in Table 4, the unscaled data resulted in the lowest RMSE of 2.390 on the validation data set. As a result, unscaled data was used during training.

Next, the number of timesteps was determined. Because 72 timesteps, corresponding to 72 hourly intervals, resulted in the lowest RMSE, 72 timesteps were chosen. As shown in Table 4, the 72 timestep LSTM model resulted in an RMSE of 1.997 on the validation data set.

Table 4: LSTM Model Training			
Parameter	**Description**	**(Values Tested: RMSE on Validation Data Set)**	**Value Chosen**
Data Scaler	Standardization or Normalization of data	{(RobustScaler: 3.019), (StandardScaler: 2.775), (MinMaxScaler: 2.777), (PowerTransformer: 5.005), (Unscaled: 2.390)}	Unscaled
Timesteps	Number of timesteps in each sample	{(1: 2.034), (3: 2.024), (6: 2.017), (10: 2.013), (24: 2.002), (48: 1.997), (72: 1.997), (168: 2.050), (336: 2.040)}	72
Hidden Nodes	Number of hidden nodes	{(53: 2.332), (250: 2.112), (500: 2.033), (850: 1.997), (1037: 1.990), (1272: 1.976), (1484: 1.972)}	1484
Layers	Number of LSTM layers	{(1: 1.979), (2: 1.975), (3: 2.089)}	1
Unidirectional vs Bidirectional Layer	Unidirectional or Bidirectional LSTM layers	{(Unidirectional: 1.975), (Bidirectional: 1.958)}	Bidirectional
Activation Function	Mathematical function that transforms inputs to outputs	{(sigmoid: 1.954), (tanh: 1.967)}	sigmoid
Optimizer	Controls how network weights are updated during training	{(adam: 1.955), (Adadelta: 2.564), (SGD: 2.217)}	adam
Number of Epochs	Number of times the model processes the input data	{(5: 2.128), (10: 2.129), (20: 2.129), (50: 2.129), (100: 2.131)}	0.5

While there are no specific rules for choosing the number of hidden nodes in a neural network, there are some empirically derived guidelines. First, the optimal number of hidden nodes is often between the number of input nodes, which is 1,484 nodes, and the number of output nodes, which is 53 nodes. Second, two-thirds of the sum of the number of input nodes and the number of output nodes, which is 1,024 nodes, often produces optimal results [28]. As shown in Table 4, a range of values between the 1,484 input nodes and 53 output nodes were chosen, resulting in an optimal number of hidden nodes of 1,484. Using 1,484 hidden nodes resulted in an RMSE of 1.972 on the validation data set.

Next, the number of layers in the neural network were tested. While there are also no specific rules for choosing the number of layers in an LSTM network, most problems can be best modeled in one to two hidden layers [29]. As such, three different models were

tested with one, two, and three LSTM layers. While the use of two LSTM layers reduces RMSE slightly more than the use of one LSTM layer, the single layer LSTM model was used to reduce computation time, resulting in an RMSE of 1.979 on the validation data set.

Next, the single LSTM layer was modeled as a unidirectional and bidirectional layer. In a unidirectional LSTM layer, each output layer receives information from past timesteps; in a bidirectional LSTM layer, each output layer receives information from both past and future timesteps [28]. Because the bidirectional LSTM layer produced the lower RMSE on the validation data set of 1.958, the bidirectional LSTM layer was selected.

After the LSTM model structure was determined, the following hyperparameters were tuned: activation function, optimizer, and the number of epochs. In deep learning, the activation function is a mathematical function that transforms the inputs fed from the previous layer to the outputs fed to the subsequent layer [16]. In regression problems, the sigmoid and hyperbolic tangent activation functions have been shown to be effective, and were thus both tested [10]. Next, the optimizer was trained, which controls how network weights are updated in order to optimize model predictions. Specifically, the optimizers Adam, Adadelta, and SGD were tested, which all use various stochastic gradient descent methods to update the network [30]. Lastly, the number of epochs, which refers to the number of times the model processes the training data set, was tested [31]. Per Table 4, using the sigmoid activation function, adam optimizer, and 5 epochs most reduced the RMSE, resulting in an RMSE of 1.917 on the validation data set.

7. Results and Comparison

The XGBoost and LSTM models were used to predict station-level hourly bike demand on the test data set, corresponding to the 53 Citi Bike bike stations between July 6, 2019 and July 31, 2019. As shown in Table 5, the XGBoost model resulted in an RMSE on the test data set of 1.31, MAE of 0.52, and R^2 of 0.75; the LSTM model resulted in an RMSE of 2.33, MAE of 1.06, and R^2 of 0.26. Thus, the XGBoost model outperforms the LSTM model in all metrics.

Table 5: XGBoost and LSTM Model Results

Model	RMSE	MAE	R-Squared
XGBoost	1.31	0.52	0.75
LSTM	2.33	1.06	0.26

Per Figures 9 and 10, the predicted and actual bike demand is shown for station 3194, which is the station with the median number of bike trips for the given data set. As can be seen, the XGBoost model better predicts peaks in bike demand than the LSTM model, which are of particular importance when rebalancing bike fleets.

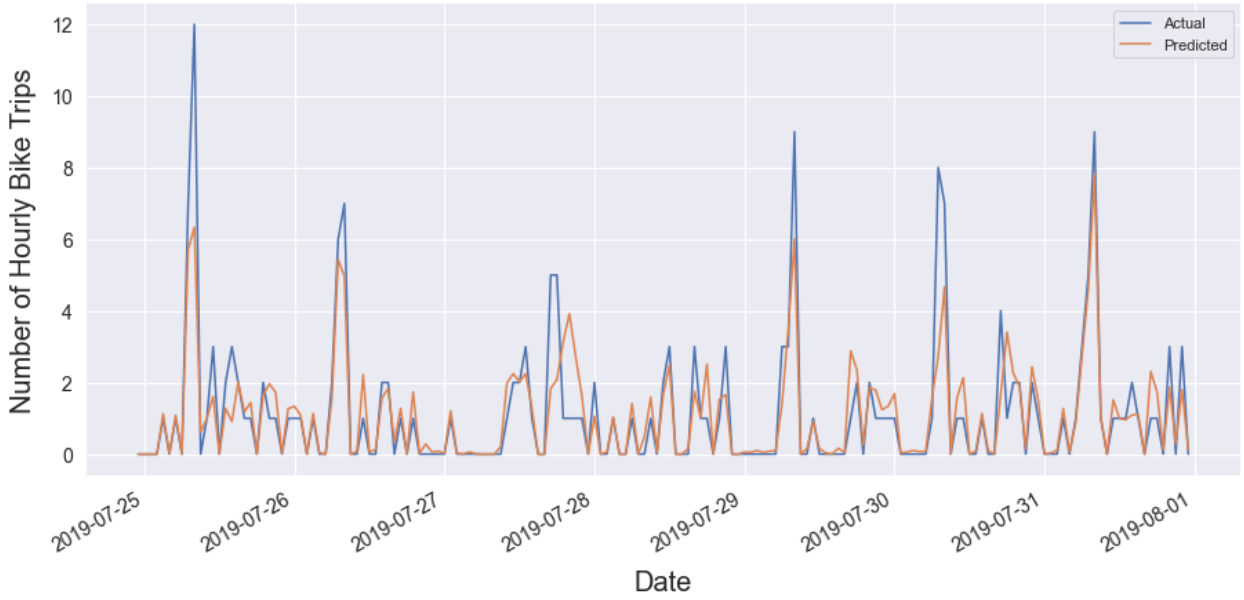

Figure 9: XGBoost Actual vs. Predicted Bike Demand for Station 3194

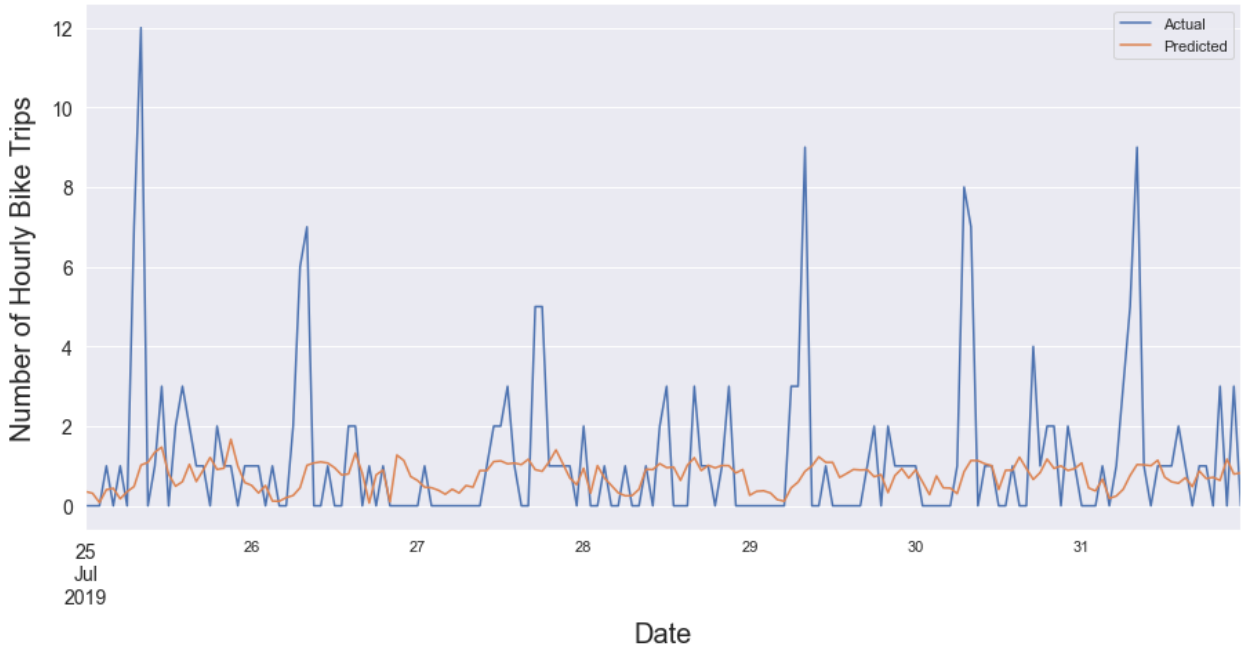

Figure 10: LSTM Actual vs. Predicted Bike Demand for Station 3194

Figures 11 and 12 show the distribution of absolute errors between the station-level hourly observations and predictions. As can be seen, the XGBoost model resulted in significantly fewer instances of large errors, which is of particular importance when rebalancing bike fleets.

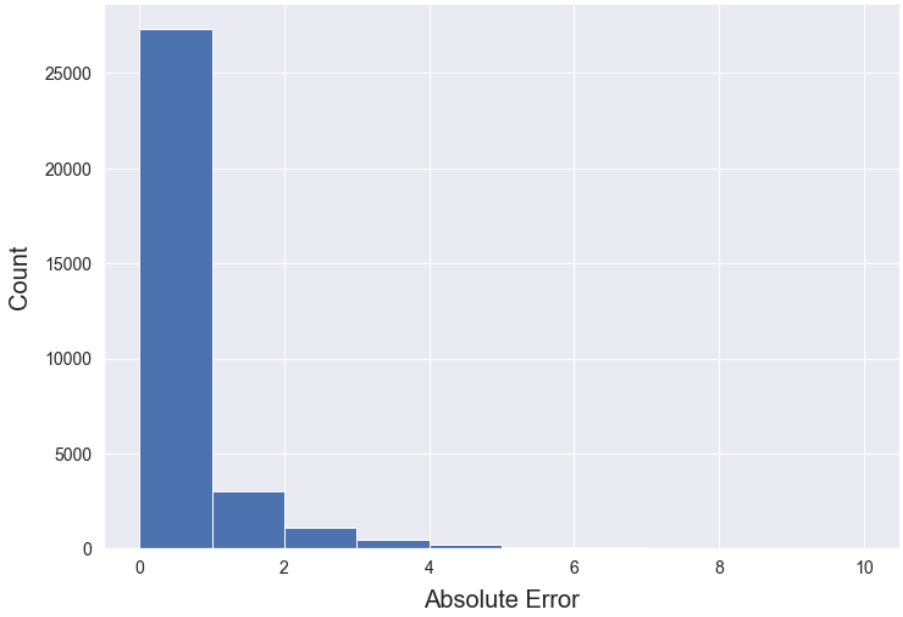

Figure 11: Histogram of XGBoost Absolute Error

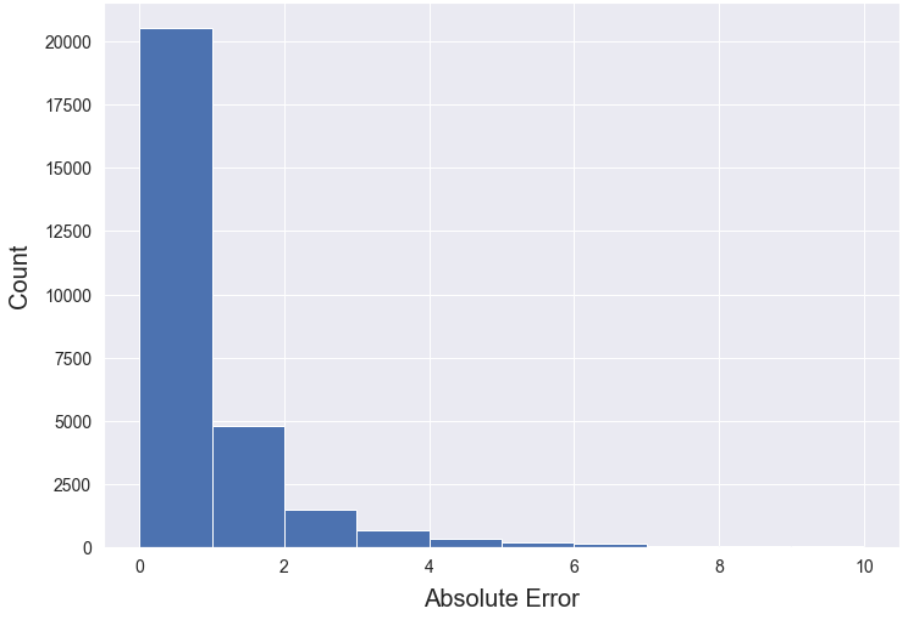

Figure 12: Histogram of LSTM Absolute Error

Table 6 below shows the distribution of absolute errors in more detail, specifically the number of predictions made by the XGBoost and LSTM models that were incorrect by more than 5 bikes, 10 bikes, and 20 bikes, out of the 32,436 station-level hourly predictions made. The XGBoost model made 361 predictions that were incorrect by more than 5 bikes, 69 predictions that were incorrect by more than 10 bikes, and 12 predictions that were incorrect by more than 20 bikes; the LSTM model made 717 predictions that were incorrect by more than 5 bikes, 195 predictions that were incorrect by more than 10 bikes, and 51 predictions that were incorrect by more than 20 bikes. Thus, the XGBoost model resulted in significantly fewer instances of large errors, which are again, of particular importance when rebalancing bike fleets.

Table 6: Distribution of Absolute Errors

Magnitude of Error	>5	>10	>20
XGBoost	361	69	12
LSTM	717	195	51

8. Conclusions, Limitations, and Future Research Directions

This report applied XGBoost and LSTM to the problem of station-level hourly bike demand. Both models utilized the New York City Citi Bike bike-sharing and National Oceanic and Atmospheric Administration (NOAA) meteorological data sets from May 1st, 2019 to July 31st, 2019. RMSE, MAE, and R^2 were used to compare the performance of the XGBoost and LSTM models. The XGBoost model outperforms the LSTM model in all three metrics. Furthermore, the XGBoost model also better predicts peaks in bike demand, thereby reducing instances of large errors between predicted demand and actual demand. Moreover, reducing these large errors is of particular importance when rebalancing bike fleets.

The XGBoost and LSTM models are limited in the input data used, the single timestep predictions made, and by not accounting for spatial dependencies between bike stations. The meteorological data was daily; thus, weather events that occurred on more discrete time scales could not be factored in. Furthermore, there appear to be abnormal peaks in bike demand, which could be the result of large, irregular events, which the input data does not account for. Secondly, the models only predict one hour into the

future and the LSTM model in particular requires significant computation time. As a result, the models may not be feasible when applying them to the problem of bike fleet rebalancing. Lastly, the models do not account for spatial dependencies between bike stations, though these spatial dependencies may be used to improve predictions.

In future research, each of these issues could be addressed to better improve bike demand predictions and thus bike fleet rebalancing. Additional data sources could be mined to better predict the unexpected peaks in bike demand and thus reduce large prediction errors, which are of particular importance in bike fleet rebalancing. Secondly, multi-step time forecasting models could be built with more focus placed on the time efficiency of each model. Lastly, Graph Convolutional Neural Networks (GCNN) could be built in order to model the spatial dependencies between bike stations and further enhance bike demand predictions. While the LSTM model captures temporal dependencies and the XGBoost model was supplemented with time-lagged factors to capture temporal dependencies, neither captured spatial dependencies between bike stations.

Source Code

The source code for the data cleaning, data exploration and feature engineering, XGBoost model development, and LSTM model development can be found at the following Github Repository:

https://github.com/mgysel/COMP3901---ML-Bike-Demand-Models

References

[1] United Nations, Department of Economic and Social Affairs, Population Division (2019), "World Population Prospects 2019: Highlights," 2019. [Online]. Available: https://www.un.org/development/desa/en/news/population/world-population-prospects-2019.html.

[2] United Nations, Department of Economic and Social Affairs, Population Division (2019). "World Urbanization Prospects: The 2018 Revision," 2018. [Online]. Available: https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html.

[3] P. Demaio, "Bike-sharing: History, Impacts, Models of Provision, and Future," Journal of Public Transportation, 12 (4): 41-56, 2009. [Online]. Available: DOI: http://doi.org/10.5038/2375-0901.12.4.3.

[4] United Nations, Department of Economic and Social Affairs, Population Division (2019), "Bike Sharing Systems," 2019. [Online]. Available: https://sustainabledevelopment.un.org/content/documents/4803Bike%20Sharing%20UN%20DESA.pdf.

[5] P. DeMaio, "The Bike-sharing Blog," 29-Jul-2020. [Online]. Available: http://bike-sharing.blogspot.com/.

[6] Zhang, Yongping and Mi, Zhifu, "Environmental Benefits of bike sharing: A big data-based analysis," Applied Energy., vol. 220, June 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261918304392.

[7] Alliance for Biking & Walking, “Bicycling and Walking in the United States: 2010 Benchmarking Report,” in Transportation Research Board Weekly, Washington, DC, 2010, [Online]. Available: https://trid.trb.org/view/914503.

[8] P. Aeschbach, X. Zhang, A. Georghiou and J. Lygeros, "Balancing bike sharing systems through customer cooperation - a case study on London's Barclays Cycle Hire," 2015 54th IEEE Conference on Decision and Control (CDC), 2015, pp. 4722-4727, doi: 10.1109/CDC.2015.7402955.

[9] H. Choi and M. Han, "The Empirical Evaluation of Models Predicting Bike Sharing Demand," 2020 International Conference on Information and Communication Technology Convergence (ICTC), 2020, pp. 1560-1562, doi: 10.1109/ICTC49870.2020.9289176.

[10] Y. Yang, A. Heppenstall, A. Turner, and A. Comber, “Using Graph Structural Information about Flows to Enhance Short-Term Demand Prediction,” Computers, Environment, and Urban Systems, 2020, vol. 83. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0198971520302544

[11] Ashqar et al, “Modeling Bike Availability in a Bike-Sharing System Using Machine Learning,” Workshops at the 29th AAAI Conference on Artificial Intelligence, 2015 [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.720.4385&rep=rep1&type=pdf

[12] L. Lin, W. Li, and S. Peeta, “Predicting Station-Level Bike-Sharing Demand Using Graph Convolutional Neural Network,” Cornell University Electrical Engineering and Systems Science, 2020. [Online]. Available: https://arxiv.org/abs/2004.08723

[13] D. Singhvi, S. Singhvi, P. Frazier, S. Henderson, E. Mahony, D. Shmoys, D. Woodard, “Predicting Bike Usage for New York City’s Bike Sharing System,” Workshops at the 29th AAAI Conference on Artificial Intelligence, 2015. [Online]. Available: https://www.aaai.org/ocs/index.php/WS/AAAIW15/paper/viewPaper/10115

[14] V. Sathishkumar, J. Park, Y. Cho, “Using Data Mining Techniques for Bike Sharing Demand Prediction in Metropolitan City,” Computer Communications, vol. 153 pp. 353-366, 2020. [Online]. Available: https://arxiv.org/abs/2004.08723

[15] A. Rogozhnikov, “Gradient Boosting Explained,” 2021. [Online]. Available: https://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html

[16] Massachusetts Institute of Technology, “MIT 6.S191 Introduction to Deep Learning,” 2021 [Online]. Available: http://introtodeeplearning.com/

[17] C. Olah, “Understanding LSTM Networks,” 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[18] S. Kim, “ML/DL for Everyone with PyTorch, Lecture 3: Gradient Descent,” 2021. [Online]. Available: https://docs.google.com/presentation/d/1CF-vEPzMSkVKnePO__AewGfzmui2aGMr8HhI9diD2t8/edit#slide=id.g27be483e1c_0_18

[19] Stanford University, "CS 230 - Recurrent Neural Networks Cheatsheet", Stanford.edu, 2021. [Online]. Available: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks.

[20] Citi Bike, “System Data,” 2021. [Online]. Available: https://www.citibikenyc.com/system-data

[21] National Oceanic and Atmospheric Administration, National Centers for Environmental Information, “Data Access,” 2021. [Online]. Available: https://www.ncdc.noaa.gov/data-access

[22] J. Moody, “What does RMSE really mean?,” 2019. [Online]. Available: https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e

[23] DeepAI, “Hyperparameter,” 2020. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/hyperparameter

[24] J. Brownlee, “A Gentle Introduction to k-fold Cross-Validation,” 2018. [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation/

[25] XGBoost, “XGBoost Parameters,” 2021. [Online]. Available: https://xgboost.readthedocs.io/en/latest/parameter.html

[26] Y. Stottner, “Why Data Should be Normalized before Training a Neural Network,” 2019. [Online]. Available: https://towardsdatascience.com/why-data-should-be-normalized-before-training-a-neural-network-c626b7f66c7d

[27] Scikit-Learn, “Preprocessing and Normalization,” 2021. [Online]. Available: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing

[28] J. Heaton, “Introduction to Neural Networks for Java, 2nd Edition,” Heaton Research, Inc., 2008.

[29] Heaton Research, “The Number of Hidden Layers,” 2017. [Online]. Available: https://www.heatonresearch.com/2017/06/01/hidden-layers.html

[30] Keras, “Optimizers,” 2021. [Online]. Available: https://keras.io/api/optimizers/

[31] Keras, “Model Training,” 2021. [Online]. Available: https://keras.io/api/models/model_training_apis/