Bike Sharing in Seoul: Optimal Predictions for Greater Reliability

Davis Alexander
Department of Computer Engineering
Stevens Institute of Technology
Hoboken, U.S.A.
dalexan4@stevens.edu

Joshua Hwang
Department of Electrical Engineering
Stevens Institute of Technology
Hoboken, U.S.A.
jhwang@stevens.edu

Harsha Tangirala
Department of Computer Engineering
Stevens Institute of Technology
Hoboken, U.S.A.
htangira@stevens.edu

***Abstract*—Ride sharing bicycles were monitored in Seoul, South Korea, while also noting several factors at each hour. Given the hour of the day and other environmental data, machine learning techniques will be used to predict the number of bikes required. Factors such as precipitation level, humidity level, and whether it is a holiday, are used to train several learning models ran as an ensemble, and the outputs' mean square errors averaged. Regression learning models, e.g. Random Forest Regression Tree and Artificial Neural Network Regression were used in our final model. Other available options are discussed, some which may provide much better results, e.g. Recurring Neural Networks.**

An ensemble of different regression models requires more internal accuracy testing. These accuracy results will be aggregated for the final model's overall accuracy. The chosen method allows for the use of many machine learning techniques to be tested for the best collection which can aggregate the most accurate prediction. The South Korean government performed the collection of data, so it is an almost ideal data set perfect for machine learning. In other words the data is very robust with no missing feature values. To prevent over-fitting from this ideal set of data, it required feature engineering and finding attributes which provided little to no information gain. Feature engineering and fine-tuning of the learning model has transformed the model into a viable option for further research. Our findings also shows how some of the individual environmental attributes correlate to the number of bikes rented in the hour.

I. Introduction

Bicycle ride-sharing data was collected in Seoul, South Korea to predict the number of bicycles needed per hour. The data used in our learning model was retrieved from University of California Irvine's machine learning repository [2]. Finding the right number of ride-sharing bikes would benefit Seoul's transportation system and reduce carbon emission. As the world becomes ever more environmentally conscious, this research and findings can prove to be invaluable. Other benefits include the reduction of waste in government spending. Not having enough bicycles or setting up bicycles too late causes missed revenue and an alternative mode of transportation being used. These missed opportunities turn into travelers using more reliable and faster means of transportation.

Discussed later in this report, are the feature engineering performed on the data set in preparation of its use by the learning models we have developed. As the data set is an official one conducted by the South Korean government, there are no missing values. One problem found however was the over-fitting of the data as the data set contained 14 columns of different attributes. Feature engineering solves not only the problem of over-fitting in the data set, but also it shows correlations between the individual features and the number of bikes rented.

Current research on similar data search for a learning model which can predict the number of ride sharing bicycles, or vehicles, needed for the most optimal supply. This research benefits ride sharing companies and other transportation entities as explained in earlier in the introduction. The machine learning model chosen will predict an estimated continuous value, while attempting to provide optimal accuracy. More formally the learning model designed for this will be given the hour of the day and several other factors, and the machine learning model will be used to predict the number of bikes required.

Regression models specifically, Linear Regression, Decision Tree Regression, Random Forests Regression, and Artificial Neural Network Regression, are used to train several learning models. This ensemble of models will each and individually make a prediction. Finally, the overall ensemble model will aggregate all of these answers and find an average for its final prediction [3][6].

Taking a look at the methods used by Sathishkumar and Cho, they all use regression methods, but do differ in architecture from the ones used by us. Due to the use of such different models, individually, the models' predictions are likely to have high variance. While each individual learning algorithm can be tested for being used solely as the final learning model, the ensemble will attempt to increase accuracy and reduce variance from the overall output.[5]. A final evaluation will be performed to find each model's effect on the ensemble, and which if any can optimize overall predictions.

The current ensemble learning model has an accuracy of 0.17% when used on the test set. A future addition to the learning model is a function to show the accuracy of the individual model. Although the ensemble learning model did not perform well, current model outputs show promise after more data cleaning and other tuning methods. The model will be tuned to reduce the error rate of the aggregated prediction.

One feature added to the current learning model is a switch

to turn bagging on or off in the individual models. Currently, when the switch is turned on for bagging, there is a slight increase in accuracy, which is not surprising. These issues with accuracy strongly represents a learning model which is overfitted. More features will be removed from the learning model as the many attributes is another indicator of overfitting.

II. Related Work

The authors Sathishkumar and Cho [7], have written two publications on predicting the ride-sharing bicycle demand given the same data as the ones used in ours. The data was collected very recently so the number of work on this data continues to grow. In Sathiskumar and Cho's publications, they present five different learning models: "CUBIST," "Regularized Random Forest," "Classification," "Regression Trees," "K Nearest Neighbour," and "Conditional Inference Tree." CUBIST learning models are a form of ensemble learning which has a decision tree data structure. Each branch serve as conditional gates. If the condition is met before a leaf node, the leaf node splits the data and fits a regression model to each one. This method should be used by novice users who cannot successfully apply neural networks or as an easy regression method.

The next model used was the regularized Random Forest. They note its advantage when handling "high dimensionality of the input." Each attribute is tested for its information gain, or how well it fits the data. This learning method keeps only the best features to make a decision. The advantage of this model is that it regularizes the data so that the random forest can greatly improve on its decisions. K-nearest neighbours (KNN) is used by the authors, which is defined by its prediction being determined by the number of neighbours, represented by the variable 'k.' The main advantage of using such a method is its reasonable output and its simplicity.

The last model used by Sathishkumar and Cho was the conditional inference tree (CIT). This model functions like decision tree models, but has a primary objective of assigning a "factor of significance." [6] This weighting method not only shows which attributes have the strongest relationship to the correct output, but it also provides the possible outcomes of of the model like a general decision tree algorithm would.

As this data set is fairly new, online databases like "Kaggle" do have it available, but with few users building learning models for it. Of the showcased models, one common factor in many of the models is the feature engineering required. Although data should always be cleaned and prepared, it is especially important for the Seoul bike data due to the large number of attributes; some are removed especially if they has little to no effect on the final predictions. Another important step taken is the normalization of the data. The data set contains different data types so without normalization, the learning model will unlikely be successful. This fundamental principle was added to our current learning model, and showed a huge improvement in accuracy. This is explained more in detail in section IV., subsection A.

III. Our Solution

A. Description of Dataset

As mentioned in the introduction, the data was retrieved from UC Irvine's repository which is a great database for many other machine learning problems. This machine learning problem of supply and demand is very common and there are several other data sets similar to the Seoul data set. Sathiskumar and Cho used their learning model from the Seoul data set, on "Capital Bikeshare" data with similar output accuracy [7]. Although we were unable to test the learning model on a different metropolitan, we should have similar results hypothetically with our chosen attributes.

Fig. 1. Bike units rented vs Hour of the day

One benefit this data set has is its complete with no missing data because this was a government regulated program and the only way data collection would be interrupted is if a sensor malfunctions, if they are using their own equipment. Otherwise, the data set can be easily collected from historical weather data.

Over the course of one year from December of 2017 to the end of November 2018, the number of ride sharing bicycles used each hour everyday were recorded. Thirteen different data types were collected as well at each hour: date, hour of the day, temperature (°C), humidity (%), wind speed (m/s), visibility (10m), dew point temperature (°C), solar radiation (Mj/m2), rainfall (mm), snowfall (cm), season, and if the date is a holiday or a functional day. Functional days are national working days in South Korea, which are not many. One observation found in functional days in South Korea, was that this was a good predictor of the rental of no bikes. These were causing obvious outliers within the data. Without removing the whole column, a more encompassing removal was the removal of any instance with zero bikes rented. As part of the data cleaning and preparation, other features were tested for their impact in the model's prediction accuracy.

The date has also been removed due to its variability and consistency. It varies every year due to leap years causing holidays to fall on different days. Some holidays occur on

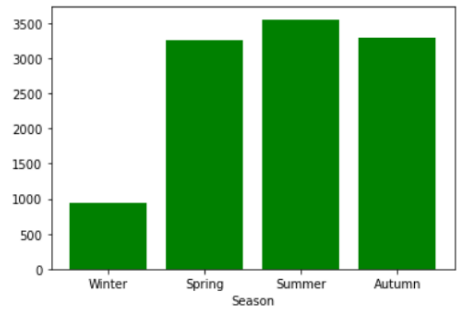

Fig. 2. Bike units rented vs Season of the year

specific days however, for example Christmas Day (December 25) or New Years Day (January 1). In place of the date and functioning day, the season and whether it is a holiday become the best determining factor in predicting bicycle counts per hour. This reduces the number of inputs or attributes considered in the final outcome.

B. Machine Learning Algorithms

The learning algorithms chosen for the ensemble stack are Linear Regression, Decision Tree Regression, Random Forests Regression, and Artificial Neural Network Regression. We chose to do regression analysis to monitor the individual learning models and to aggregate an optimal output. The regression mean squared error represents the models' performances.

Given the many different features like temperature, humidity, wind speed, visibility, hour of the day, if it is a holiday or not, doing a simple linear regression analysis of the data can act as a performance measure as different attributes are removed. It can be determined which attributes are statistically significant and which features can be ignored.

For simplicity we chose Decision Tree as one of the algorithms' since it can handle both numerical and categorical variables and is not easily influenced by outliers and missing data. Using decision tree's also gave us a chance to look at non-linear relationships between the variables. However, decision trees are prone to over-fitting so we added another algorithm to the mix.

Random Forests are a collection of decision trees. So rather than using a single optimized decision tree, the Random Forest can be trained to run predictions on each individual tree in the cluster to come up with the final prediction.

Finally, we also used Artificial Neural Networks (ANNs) because of its ability to learn and model non-linear and complex relationships and unlike most other prediction techniques, ANNs do not impose any restrictions on the input variables, like for example, how they need to be distributed. Prior studies have shown that they create better models when dealing with highly volatile data and non-constant variance because it has the ability to learn hidden relationships in data without imposing any fixed relationships.

Considering the ensemble bagging to be the total learning function, a holdout set is set aside as a test set for the accuracy of the ensemble model. Scikit's ensemble Bagging Regressor splits the training data into random sets, different sets for each individual regression model.

Linear Regression has been implemented simply by using python's built in code. No regularization terms have been added. For Decision Trees, the team used GridSearchCV to find the optimal parameters for "max_leaf_nodes" and "max_depth", as this pruned the tree. Due to this function, the tree models have consistently shown lower error rates. For Random Forests Regression, the same parameters from Decision trees are used. GridSearchCV with Random Forests took too much processing time without giving any results. For Multilayer Perceptrons, or Artificial Neural Networks, multiple tests were conducted to find the optimal "Alpha", "Learning Rate", and "Momentum". After many tests, the results were compiled together and the average of these values were taken. Alpha was set to 0.00000631, learning rate was set to 0.01, and momentum didn't impact the model, so it was ignored. After optimizing ANN, the error was reduced. The team focused on improving individual models to improve the overall accuracy. If each model was optimized to the best of its capabilities, the idea was, the average ensemble of them would yield better results and higher accuracy.

C. Exploratory Work With Time Series Forecasting

Since this resembled a forecasting problem, we looked into implementing a time series forecasting model. Within our data set, the rented bike count is the metric that is recorded every hour. Using a forecasting model, we could predict how many bikes we would need in the future so as to reduce number of unused bikes. There are two types of time series forecasting. The first one is called a univariate time series forecasting model where you only use previous values of the time series to predict future values. And then there is multi variate time series forecasting where you use predictors other than the series itself, in our case, we could use predictors like temperature, humidity, snowfall etc.

ARIMA or Auto Regressive Integrated Moving Average models try to explain a given time series based on it's own past values - what we call 'lags'. For example, if we were to expect car sales to follow a particular trend every year - a peak during the spring months and another peak during the fall months, we would say lag is 12 assuming we have monthly sales data. Alternatively, we could break up sales into 4 quarters of the year and say lag is 3. January through March, April through June and so on. That way we could study the seasonality within each quarter. The ARIMA model is characterized by 3 terms: p, d and q. We shall dive further what those terms mean.

Auto-Regressive (AR) means it is a linear regression model using it's own past to predict the future but linear regression works only when the predictors are not co-related. Now the question is how does one remove any co-relation to one's own past behaviour? In other words, how do you make the time series data stationary? A common approach is to difference it which means simply subtract the previous value from the current value. Depending on the complexity of the series, more

than one differencing may be needed. The d term in the model indicates the minimum number of differencing needed to make the series stationary. The p term refers to the number of lags as we've seen before. 12 month lag to capture seasonality in a year whereas a 3 month lag to capture seasonality in a quarter year, 24 hour lag to capture seasonality for a day and so on. Finally, the q term refers to the order of the Moving Average (MA) - number of lagged forecast errors that should go into the model.

Fig. 3. Moving average with window size of 24 hours

We found there were a few days where the rental company closed shop and the bikes rented that day was 0. Our first task in preparing the data for this model was to make it stationary and we had a difficult time doing this. Figure 4 gives us an idea of what the raw data (d = 0) looked like with a lag of 24.

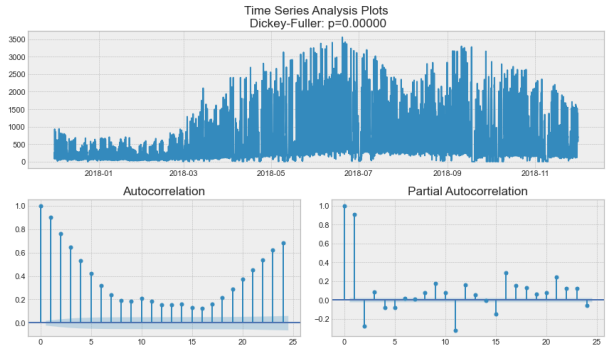

Fig. 4. Time Series Analysis - Original Series

In figure 5, we set d = 1, the lag goes into the negative very quickly which left us to suspect that the series might've been over differenced. Something else that caught our attention is that the p-value remained at 0. At this point we decided to look at alternative algorithms to lean on and not pursue time series forecasting models.

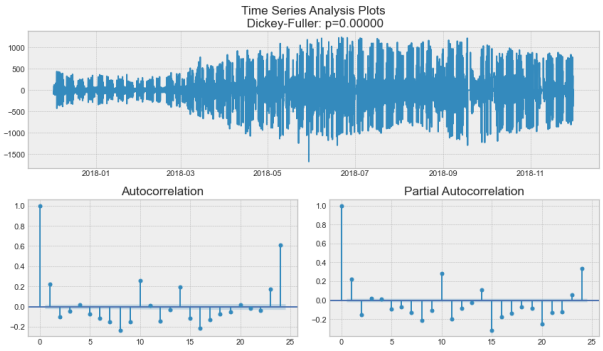

Fig. 5. Time Series Analysis - d = 1

D. *Implementation Details*

Feature	Feature Importance
Seasons	0.02664
Dew point temperature(°C)	0.15110
Temperature(°C)	0.15194
Humidity(°C)	0.14452
Rainfall(mm)	0.01616
Visibility(10m)	0.13354
Solar Radiation(MJ/m2)	0.08874
Hour	0.12178
Wind speed(m/s)	0.14528
Holiday	0.00842
Snowfall(cm)	0.01188

Normalized Feature Importance.

The number of attributes required data cleaning and feature removal, explained in Section *A. Description of Dataset*. Rapid prototyping was implemented at this point, and further feature removal will be done after analyzing each features significance in the most accurate output. After removing the obvious attributes, the data was split into a training set and testing set with a 80:20 ratio.

The different models were all defined in one module for easy troubleshooting. Next, all the models were trained and tested with the mean squared error serving as a performance measure. Combining the regression models' predictions, otherwise known as ensemble learning, an average is given for the final prediction.

In the development of the regression models, many of the hyperparameters were not changed from their default values. Many cross-validation techniques were used to fine-tune the learning models. The regression functions could be tested with bagging implemented within its operations. The *BaggingRegressor* function was used, which aggregates base regressors' predictions and outputs the average, or outputs a decision made by majority vote. The *GridSearchCV* function was another cross-validation tool used to fine-tune the parameters of the

tree models like the number of leaf nodes, or the maximum depth the tree can be.

IV. Comparison

Each model used in the ensemble model were tested individually on the test data for their accuracy. In the tables below, some regression models found in the preliminary results are missing from the final accuracy results. Fine-tuning the model and working with an ensemble allowed for this, finding and removing the method which decreases overall accuracy.

Algorithm	MSE
Linear Regression	223742
Decision Tree Regression	108020
Random Forest Regression	103655
MLP Regression	215177
Fusion Ensemble	135515

Preliminary Accuracy results.

Algorithm	MSE
Decision Tree Regression	71701
Random Forest Regression	68557
MLP Regression	103527
Fusion Ensemble	

Fine-tuned Accuracy results.

A. Improvements

This section describes the group's efforts to improve the accuracy. When the model code was first developed, the team had written the code to do the following: read the data file, split the data into 80% training and 20% testing data, and perform linear regression, decision trees, random forests, and artificial neural networks. At the end of this evaluation of each regression model, the individual predictions are combined via simple average ensemble to produce the final predictions list. Afterwards, the team used the function "accuracy_score" to evaluate how well the model performs. The initial accuracy score was 0.11%, which was abysmal, and the Mean Squared Error (MSE) was around 200,000. The team considered adding a bagging method for each of the regression models, and that improved the accuracy by 2%, and the MSE dropped to roughly around 160,000 - 180,000. Afterwards, the team performed min-max normalization before the data was split, and noticed that "accuracy_score" was an improper choice of accuracy estimation, because that is only used for classifier models. A good way to evaluate regression models is to use the "r2_score", a number ranging from 0.0 to 1.0, where 1.0 is the best. After this change, the accuracy was around 57% and the Mean Squared Error was roughly around 120,000 - 160,000.

Normalization is highly beneficial when the team focused on improving the accuracy of the models. The numbers for each of the rows become scaled between 0 to 1, rather than one column having small numbers and another column having large numbers. Having all the numbers be on the same scale allows the models to better learn from the data, since everything is on a balanced playing field. For this project, min-max normalization has been used and the Regression Score jumped from 55%-57% to 75%-77%. Min-Max normalization guarantees all the numbers have the same scale, however one drawback is, it struggles to handle outliers because the minimum and maximum numbers skew the data oddly. Future directions to improve accuracy will be to focus on reducing/removing outliers from the data, thus making normalization more robust.

After further research, and reading online articles, it was recommended that normalization should be done after the data has been split into training and testing data. This is done because if normalization is done prior to the split, training data information leaks into the testing data. Normalization after the split led our r2_score to be around 78%-80% and Mean Squared Error roughly around 80,000 - 100,000, which is very respectable. Further potential exploration into improving accuracy was done by the team, and one way the team thought about improving accuracy is to completely eliminate Linear Regression from the equation, as it has been consistently seen to have high MSE and it drags down the other models and increases error. After removing it, the team noticed the MSE hovered around 79000-80000, and the r2_score is 80%-81%. There wasn't a significant change in the r2_score, so having the Linear Regression in the ensemble equation didn't hurt the final prediction too much. The goal is to increase accuracy, and avoid over-fitting.

V. Future Directions

With the present model's accuracy, the ensemble will be evaluated for each regression model's influence in the ensemble's overall accuracy. An ensemble of learning techniques will perform best with the objective of predicting the number of bicycles needed per hour. The ensemble learning model will be tested on different countries' data to evaluate current techniques as well as searching for other attributes to improve our learning model. Once the overall algorithm has been refined, other machine learning algorithms can be added to see the effects to the ensemble's accuracy.

If there were more time for this project, genetic algorithms can be explored. Genetic algorithms has many relevant advantages to the current problem. In doing a little bit more research into the problem, we came across time series forecasting applications such as predicting power demand for a city or forecasting call volumes at a call center in order to schedule staff accordingly or forecasting supply and demand to optimize fleet management. All of these problems that dealt with uncertainty about the future use time series models. We feel our problem statement lies in the area of forecasting how many bikes may be needed at a time in the future. This is even suggested as a solution for time series data in the reconfiguration of power distribution systems [8]. Power distribution requirements relate very well to the problem we are solving for ride sharing. A genetic algorithm is perfect for providing one of the best models to fit a regression on a time-series data set. The downside to this however is that

this will increase the run time of the original machine learning algorithm significantly. In the proposed genetic algorithm, it would accept arrays of weights and swap some elements, or even ”mutate” them [3].

Other methods which may be explored are Recurring Neural Networks, as they are most appropriate for time series data. As the name suggests, this method implements a network of recurring neurons. Each single neuron receives an input and also receives its own output from one time frame before. Upon further research, the most appropriate form of RNN, is the use of ”cross-batch statefulness.” A fairly intuitive approach to this is to use keras's neural network library and add layers as needed. In order to prepare the data for this approach, the data will have to be split into batches, while also including their labels. The reason for this is that the time series data we are dealing with was collected over the period of one year, each instance separated by one hour. This is where ”Long Short-Term Memory” LSTM layer(s) are added to the empty Recurring Neural Network. LSTM cells are comprised of a main layer which takes in some input along with its previous state, and gate controller layers. Lastly, the LSTM layer will be initialized with ”stateful” equal to ”True.” This boolean parameter will allow the recurring neural network to retain its previous states, or have a ”memory” of previous states. Now the batches can be used to train the learning model, with each batch's effect saved to be aggregated with the following batch [4]. One way to begin implementing this method has been shared by Dr. Jason Brownlee. He shares the idea of training on large batch sizes, and then [1] ”a batch size of 1 when making prediction in order to predict the next step in sequence” [1]. This is an interesting approach to using cross-batch statefulness with LSTM layers.

One possible direction for the project is to research ways to normalize the input data and have weights, giving the ”Hour” predictor more significance. Hour of the day heavily influences the day-to-day demands of the workforce, as people need to get to their destinations more during the day-time, rather than night-time. Fixing the training data inputs of the individual machine learning algorithms will affect the overall accuracy.

Another direction is to focus on improving the blending algorithm for the ensemble techniques by using a hold-out training set. Currently, the algorithms focus on calculating individual regression predictions and consolidating them into an average for the final prediction. More emphasis on data processing and blending these predictions could be done to improve accuracy.

VI. Conclusion

As ride sharing becomes more popular and accepted as a primary means of transportation, predicting the number of bicycles needed per hour greatly benefits the vendors and sometimes the environment. The distribution of available bikes being as close as possible to the number of bikes required reduces waste, ensures availability for customers, and improves overall profit efficiency. Specifically in the case of bicycle ride sharing, carbon emissions are reduced, which is very important in the world's attempt in a more sustainable future [9]. Although regression models were chosen, the introduction of models specifically for time series data become the better option.

Past results showed signs of over-fitting so removing instances with zero bikes rented significantly improved the final results of the current learning model. The current learning model can almost be considered as choosing numbers for the winning lottery ticket. Sathishkumar and Cho used a conditional inference tree, a model which assigns a factor of significance to each attribute. This model may not be the final answer to the current over-fitting problem, but it can provide insight into which attribute can be removed. The current learning model can be run again with a more reasonable number of attributes.

A validation algorithm to consider are finding a range of bicycles required per hour opposed to an absolute value. This will compensate for any variance in the predictions, and allow for the use of classification algorithms. This may be the more appropriate method to give if deployed. The prediction can be left for more complex decision makers, i.e. the model's machine learning algorithm, based on this range.

Discovering machine learning models to predict the demand of ride sharing is still a fresh subject to explore. Finding the best one can increase profit margins for ride sharing companies, so this discovery would be highly sought after by a number of large corporations which use the ride sharing business model. Building an efficient network in bicycle ride sharing is just one sector helping environmentally. Many more networks which have definite benefits in the world's sustainability can also benefit from this.

References

[1] Jason Brownlee. How to use different batch sizes when training and predicting with lstms. *Machine Learning Mastery*, Aug 2019.

[2] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[3] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Sebastopol, CA, 2 edition, 2019.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] VE Sathishkumar and Yongyun Cho. A rule-based model for seoul bike sharing demand prediction using weather data. *European Journal of Remote Sensing*, 53(sup1):166–183, February 2020.

[7] VE Sathishkumar, Jangwoo Park, and Yongyun Cho. Using data mining techniques for bike sharing demand prediction in metropolitan city. *Computer Communications*, 153:353–366, March 2020.

[8] Bogdan Tomoiagă, Mircea Chindriş, Andreas Sumper, Antoni Sudria-Andreu, and Roberto Villafafila-Robles. Pareto optimal reconfiguration of power distribution systems using a genetic algorithm based on nsga-ii. *Energies*, 6(3):1439–1455, Mar 2013.

[9] Yongping Zhang and Zhifu Mi. Environmental benefits of bike sharing: A big data-based analysis. *Applied Energy*, 220:296–301, 2018.