Name of Student: Ryan Wong

School: International School of Beijing

Province: Beijing

Country: China

Name of Instructor: Tyler Beatty

Institution of Instructor: International School of Beijing

Title: A Deep Learning Approach to Predict the Number of Bikes a Bike Sharing Company Repairs

A Deep Learning Approach to Predict the Number of Bikes a Bike Sharing Company Repairs

Ryan Wong

International School of Beijing

Abstract

Bike sharing companies constantly need to repair damaged bikes and redistribute them to different stations; this could be done more efficiently with a prediction for the number of damaged bikes needing repair and distributions of users across each station. Existing work on bike sharing data is limited by the use of a linear regression model, which assumes that there is a linear relationship between the historical usage statistics and the future demand. In addition, prior work has not taken advantage of detailed data, such as number of users for each station. This paper uses deep neural networks to predict the future demand based on historical data, including: temperature, humidity, air pressure, users per station, and average duration of each trip. We compared neural networks to linear-based techniques on the mean-square error between the target values and predicted values. Experiments show that a deep learning approach produced more useful and accurate predictions than linear-based techniques on both the future demand and future number of repairs. This suggests that the data does not have a linear relationship. We also examined the results of using detailed data over less detailed data. Experiments show that more detailed data improved prediction accuracy of the neural networks, but reduced the accuracy of linear regression. Lastly, we removed individual input features; results show that neural networks are significantly impacted due to the removal of features but not linear regression. All code is on `https://github.com/dtarcug/Yau-Ryan-Wong-2021`.

Keywords: bike sharing, neural network, linear regression, repair, distribution

Table of Contents

1 Introduction

Bike sharing is a shared transport system that is becoming increasingly popular due to its low cost and health benefits. The system involves short-term rental bikes that are available for individuals. Most bike sharing systems operate by setting up multiple stations in a region for users to borrow and return. To rent a bike, users may go to one of the stations and enter payment information, after which the computer unlocks one of the bikes. To return the bike, users place the bike in any of the stations and the bike would automatically be locked.

Rental bikes are often damaged due to frequent use or mistreatment from users [1]. Bike sharing systems would not be as troubled by this issue if the number of damaged bikes needing replacement at each station could be predicted in advance. This paper aims to improve decision-making for bike sharing systems by creating a regression model that predicts the number of damaged bikes and distribution of users for each station per month.

This introduction will describe the context, data, and techniques that form the foundation to this work.

1.1 Related Work

One of the past attempts in tackling a similar issue is from Gaurav Dutta [2]. In Dutta's attempt, he uses a linear regression model on a bike sharing data set from UCI Machine Learning Repository, which does not include the usage of individual stations and average distance traveled per trip. Linear regression models are limited to linear relationships, which are highly unlikely to occur in a dataset with such high complexity. Therefore, using other regression models such as neural networks can potentially improve the results of the prediction. Another opportunity for improvement is to include more detailed data, such as distribution of users across stations instead of the total users of the company,

which may improve the prediction accuracy of the model.

Another similar work was made by Misael Uribe [3]. Since he used a similar data set as Dutta, it also does not include detailed information. In Uribe's work, he did not train a model to predict future bike usage, but trained a model to address null values in the data set. Some of the data points in the data set were not given, so Uribe used a random forest regressor to predict and replace these values. This is a good choice for his problem as random forest regressors do well on problems that do not require extrapolation. Our work is attempting a different problem and uses neural networks instead.

1.2 Data Sources

All of the data used are from New York. The data spans from June 2013 to December 2020, for a total of 91 data points (68 in train set and 23 in test set). We make use of the bike trip dataset [4] and monthly operating reports [1] provided by Citi Bike, and we also use the weather data [5] provided by Time and Date. The bike trip dataset includes monthly records of bike rentals, the latitude and longitude of each of Citi Bike stations, and the average trip duration. The monthly operating reports includes the number of repairs that Citi Bike made in each month. The weather data includes average temperature, average humidity, and average pressure.

1.3 Problem Definition

There are two tasks for the models to predict: distribution of users for each of the 10 clusters of stations and number of repaired bikes in the following month. **Task 1** models takes in the following inputs: month number, year number, number of starts for each cluster of stations in the month, average trip duration in minutes, average temperature in degrees Celcius, average humidity in percent,

and average pressure in millibars. It outputs the distribution of users in each of 10 clusters (explained later in the paper) of stations. **Task 2** models takes in the same inputs as task 1, but outputs the total number of bikes needing repair. The performance of the models are measured using mean-squared error (1).

The first task is outputting the distribution of users instead of the actual number of users across the clusters because we want to find how many new bikes should be put in a cluster relative to other clusters of stations. In addition, a distribution is easier to compare the losses when the total number of users is different.

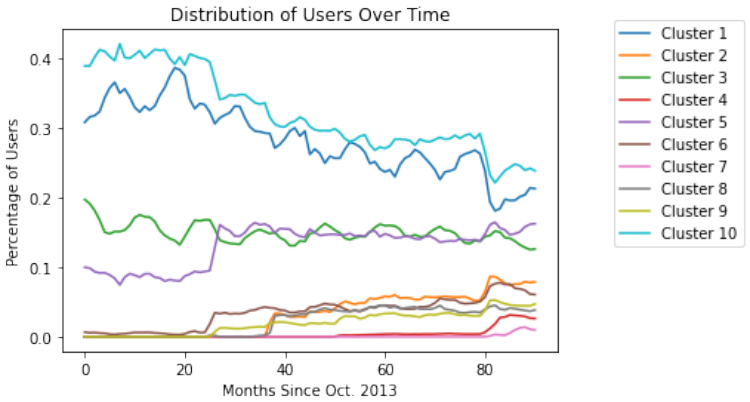

Figure 1: Target Values for Task 1

1.4 Multiple Linear Regression

A multiple linear regression model would be created as a comparison to neural networks. A linear model with n input features $x_1, x_2, ..., x_n$ and an output y could be written as

$$y = \beta_0 + \sum_{i=1}^{n} (\beta_i x_i) + \epsilon$$

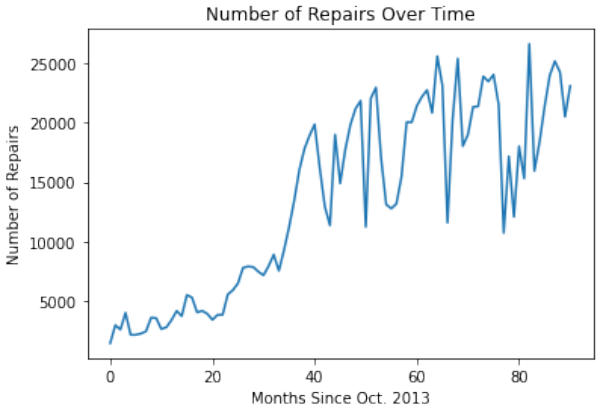

Figure 2: Target Values for Task 2

In the equation, β_0 could be interpreted as the y-intercept, and β_i where $i > 0$ are the slopes for each input feature. The linear model is simple (thus easy to learn on small amounts of data), but is not general (unable to model many functions accurately).

1.5 Neural Network

In contrast to linear regression models, deep neural networks [1] do not assume a linear relationship between the input and output. A neural network is a collection of artificial neurons that are connected in layers, as seen in figure 3. Each neuron takes in some input, either from the user, other neurons, or a bias term. It outputs a non-linear function of the weighted sum of inputs to the next layer of neurons. (This non-linear function is often called the activation function. Common activation functions include: Rectified Linear Unit (ReLU), sigmoid, Exponential Linear Unit (ELU), softmax, etc.) For instance, the activation function ReLU is given by the formula

$$ReLU(x) = max(0, x)$$

[1] This paper is going to focus on only the feedforward variety

where x is the weighted sum of inputs. During training, the weights and the biases are tweaked to improve on an objective. Neural networks may often overfit the train set; to minimize overfitting, k-fold cross validation could be used to detect if further training will worsen test performance. A deep neural network was used in this paper as they specialize in modelling high dimensional inputs and complex functions.

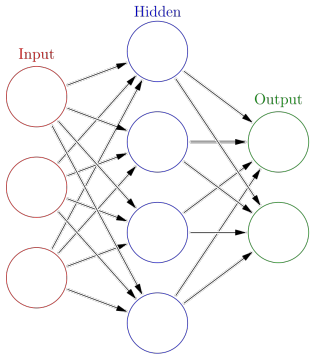

Figure 3: Example Structure of a Neural Network
(Figure reproduced from [6])

The network learns through the use of a technique called backpropagation. Backpropagation efficiently uses the chain rule to calculate the gradient of the loss function (a function which the network aims to minimize) with respect to each parameter of the network. In backpropagation, the gradient of parameters of the last layer are computed first, then propagated backward to simplify the gradient computation in the preceding layers of the network. After the gradients have been computed, gradient-based local optimization methods such as gradient descent or stochastic gradient descent could be used to update the parameters to locally minimize the loss function [7].

2 Data Pre-processing

There were some missing data and some data that did not make sense. In the data for number of starts of each month, some of the station ID were left blank, so the values for these points were ignored. In addition, the latitude and longitude of some stations did not belong to New York; since these stations are not within the region this paper is focusing on, their values were also ignored.

In total, there are more than 1000 Citi Bike stations, making this a high-dimensional problem relative to the amount of data available. In order to reduce the complexity of the problem, we use a clustering technique known as K-means clustering to partition the stations into 10 groups (the number of groups was chosen to balance the tractability of the problem with the granularity of the predictions). K-means partitions the input into a pre-defined number of groups, k, such that each point is near the average for the group it is assigned to. In our experiment, each station is represented by their latitude and longitude. The longitude of each station was multiplied by the cosine of its latitude to account for the different scales of latitude and longitude (the latitude of New York is approximately 40.7). K-means was used instead of other clustering techniques due to its simplicity and its use of distance-based measurements. The result of the clustering is shown in figure 4; the clustering had been done on multiple seeds and the results were similar. After clustering, the data set is randomly split into a train set and a test set, with 68 and 23 data points, respectively.

3 Method

While the prior introduction has provided the general context for the work of this paper, this methods section describes the particulars of the experimental evaluation as well as the specific neural models used.

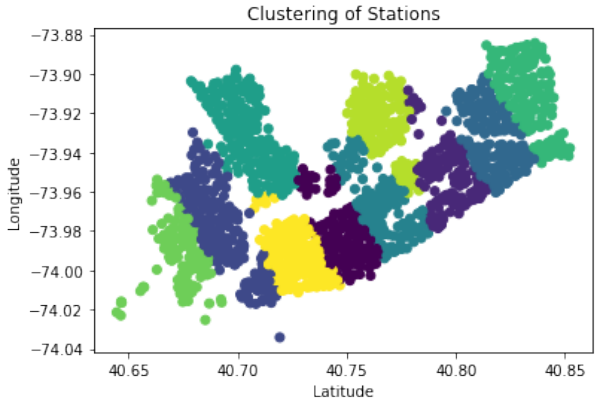

Figure 4: Clustering of Citi Bike Stations

3.1 K-fold Cross Validation

For the neural networks, we need to know when to cut off the training for them so that the they do not overfit the train data (make predictions based on the idiosyncrasies of the training points that generalize poorly to new data) [8]. As seen in Figure 5, although the model in the graph on the right performs

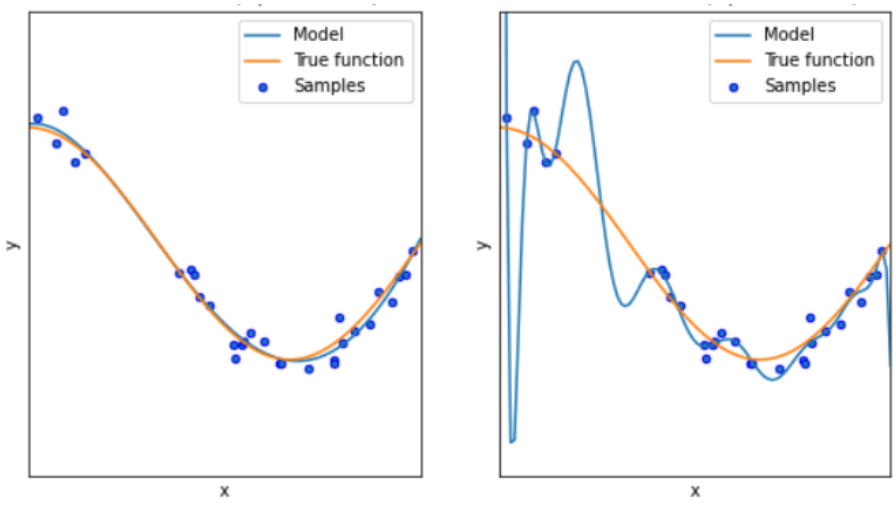

Figure 5: Example of Overfit (Figure reproduced from [9])

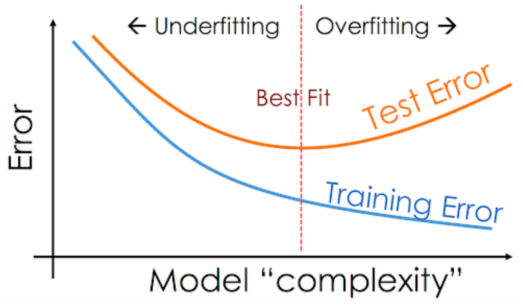

Figure 6: When to Stop the Training (Figure reproduced from work by Kumar [10].)

better on the train set, it would not perform as well as the model to the left if more samples were taken from the true function. Therefore, it is important to stop the training process before the network starts overfitting the train set. To set which epoch[2] to cut off training, we used K-fold cross validation, a validation technique used when there is limited train data. In k-fold cross validation, the train data is partitioned into k subsets of approximately equal size. Once for each of the subsets, it is treated as the validation set, the other subsets are used to train the data, and the network's performance on the validation set is recorded. Lastly, the recorded performances are averaged.

Before the training of the network used as the model, we first trained k networks using k-fold cross validation. For every 10 epochs of the networks, the average performance is recorded. When the average validation-set performance starts to decrease (the average loss increases), this is a sign of overfitting, so the training for the networks are stopped. After that, the network used as the model is going to train for the same number of epochs as the networks, but with the full train set instead of $k - 1$ subsets.

[2]Epoch: number of complete passes through the train set

3.2 K-means Clustering

K-means starts by assigning k randomly selected points as centroids (or "means"), which represents the center of each cluster. Then, each point is assigned to the closest centroid based on their squared Euclidean distance. Next, the mean of the points for each cluster becomes the new centroid. The previous two steps are repeated until the algorithm converges [11].

3.3 Adaptive Moment Estimation (Adam)

Adam [12] is an effective and widely-used optimization algorithm, which is used for updating a neural network's parameters. It calculates individual learning rates for each parameter of the network by estimating the first and second moments of the gradients. It starts by initializing two variables, m and v, to 0 representing the first and second moments, respectively. For each iteration t, the following operations are applied:

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2}$$

$$x_{t+1} = x_t - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}}$$

α is the initial learning rate of the network. g is the gradient of the loss with respect to the parameter, x. β_1 and β_2 are the decay factors for m and v, respectively, and are between 0 and 1. Since m and v are initialized to 0, the moment estimates are biased towards 0. To counteract this bias, $\hat{m}$ and $\hat{v}$ are used to update the network's parameter instead of m and v. ϵ is a number

extremely close to 0, such as 10^{-8}, to avoid division by 0 when updating the parameter.

3.4 Constructing Neural Networks

Two neural networks were constructed for each of the tasks, one with a softmax[3] layer as the last layer (for the first task) and the other with a regular linear layer (for the second task).

Before any training, the last linear layer of the network for the first task is likely to have output one value that is relatively large and the others relatively small. This would make the softmax layer output all 0's but a 1, which is hard for the network to optimize. Therefore, the network was first trained for 1000 epochs before adding the softmax layer.

Before the addition of the softmax layer, the first task's network would have 30 hidden layers, each having 10 neurons. For the second task, the network would have 35 hidden layers, each having 10 neurons. The depth and width of both of these networks have been tested and adjusted multiple times, and those were the ones that performed the best.

During training, the predictions made by the model is compared to the label with the Mean Squared Error loss function

$$MSE = \frac{1}{n}\sum(y_i - \hat{y}_i)^2. \tag{1}$$

Then, backpropagation is done to compute the gradient of this loss function with respect to each of the parameters in the network. With these gradients, the network changes the values of its parameters using the Adam algorithm. This process is repeated until the network performs worse on the cross validation.

[3]Softmax: transforms a vector of real values into a vector of the same size that sums up to 1

3.5 Multiple Output Linear Regression

Since a linear regression model only outputs one value and the first task requires output for each cluster of stations, we had to train multiple linear regression models, one for each of the clusters. Similar to what has been done to the outputs of the neural network, the outputs of the linear regression models were scaled with softmax. However, a validation process was not needed as linear regression is not an iterative algorithm, meaning that a model could be found mathematically in an instant instead of doing multiple iterations of the same process, such as gradient descent.

4 Experiments

This section describes the training on two related tasks. Two networks and two linear models were created for each of the two tasks.

4.1 Models

K-fold cross validation with 6 folds (6 folds were chosen due to runtime constraints) was used to determine the number of epochs the networks should be trained for, and the two networks were trained. Then, a multiple output linear regression model was trained on the first task and a regular linear regression model was trained on the second task. The accuracy of the networks and the linear models were compared on the test set.

We also tested whether the breakdown of users across the 10 clusters provided useful information beyond just the total number of users. The above process was repeated on modified input data: instead of including number of users in each cluster, it includes only the total number of users (the output is still distribution across 10 clusters). The accuracy of the models were compared

with the original models on the test set.

Lastly, we wanted to know which input features had the most impact on the prediction accuracy. The process was repeated with each input feature removed. The accuracy of the models were compared with the original models on the test set.

4.2 Train Results

During the cross validation of the first task, the loss did not have any increase in loss, so we just stopped the validation at the 3000th epoch and trained the network. The network's loss before adding the softmax layer was at approximately 0.001, after adding softmax, the loss improved to 0.0004. For the second task, the cross validation loss increased at the 450th epoch, so the network was trained for 450 epochs. The network's loss on the train set ended at approximately 15 million. The linear models' train losses were 0.01 and 7.7 million for the first and second task, respectively. (The figures below are losses on the test set, which is different to the train set)

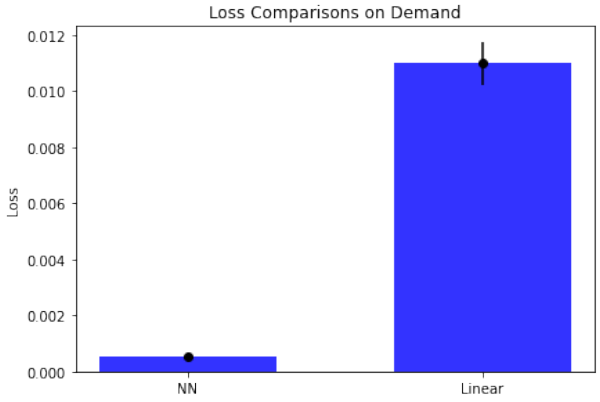

Figure 7: Loss Comparison on First Task

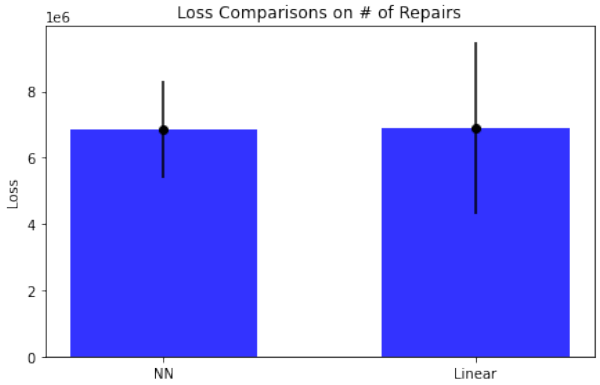

Figure 8: Loss Comparison on Second Task

5 Discussion

This section analyzes and interprets the results of the experiment. In addition, this section lists the limitations of the experiment and potential solutions to them.

5.1 Comparisons of Models

Based on figure 7, it could be seen that the neural network's test loss for the first task is much lower than that of the linear regression. It is clear that the neural network model is better for predicting the distribution of bike usage among the stations.

As for the second task, the neural network performs slightly better than linear regression on the test set. However, as seen in figure 8, the difference in loss is within the 90% confidence interval, along with the fact that neural network takes more time to train, linear regression may be a better choice in some scenarios.

In addition to comparing results between neural networks and linear models,

we also compared the models' prediction accuracy on a detailed input data set (stations grouped into 10 clusters) and a non-detailed input data set (the stations are all in one single group). As seen in figures 9 and 10, the neural networks perform better when trained on a more detailed data set than on a less detailed one. This result is expected because with more detailed data, the model can identify more relationships between the input and the output, therefore increasing the prediction accuracy. However, it was surprising that having less detailed data did not have a large impact on the loss for linear regression on the first task. As it is expected that knowing the distribution of users in the previous month would help predict the distribution in the next month.

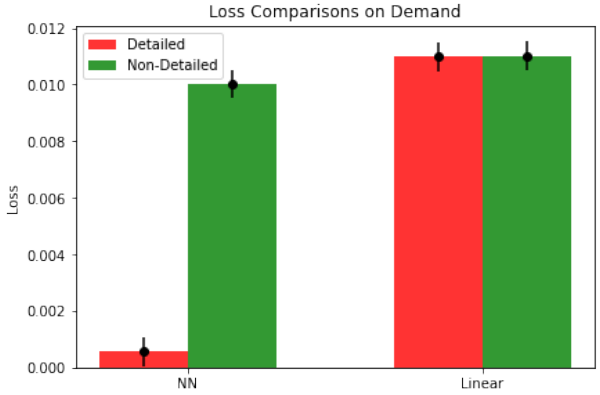

Figure 9: Loss of Detailed Data Compared to that of Non-Detailed Data for First Task

To see which input features have the greatest impact on the prediction accuracy of the models, we trained a neural network and a linear regression with each of the input features removed. From figure 11, it could be seen that the temperature has a large impact on the first task's loss for neural networks. This might be some parts of the city attract bikers only when the temperature is pleasant. On the other hand, figure 12 shows that the impact of removing any of the input features on the prediction accuracy of the linear model is much less than the impact on neural networks, this is also the result in the second

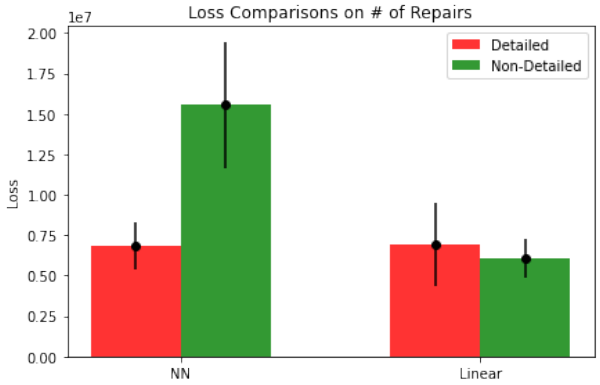

Figure 10: Loss of Detailed Data Compared to that of Non-Detailed Data for Second Task

task shown in figure 14; this suggests that neural networks are more sensitive to the inputs than do linear models. In predicting the second task, figure 11 shows that the month had the largest impact on the prediction accuracy. The month might affect the results because there would be different number of users. Ultimately, in the case of having missing input features, linear regression may be a better choice as it is less sensitive to the removal of features.

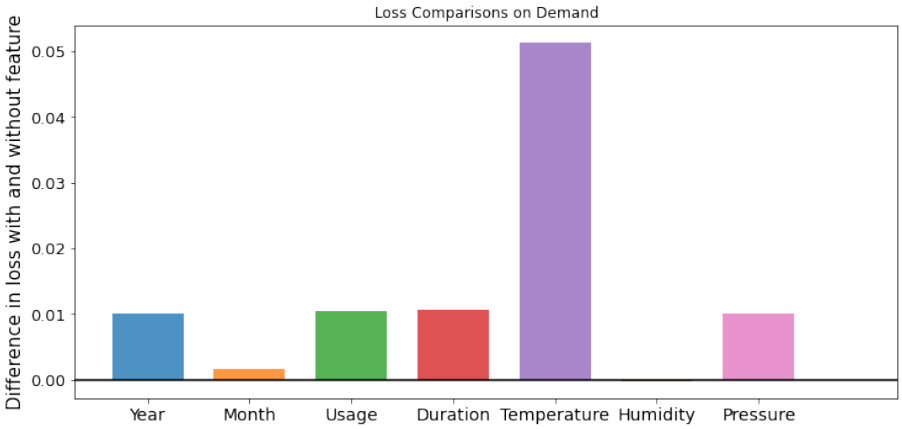

Figure 11: Changes to the Neural Network's Loss on First Task from Removing each Input Feature.

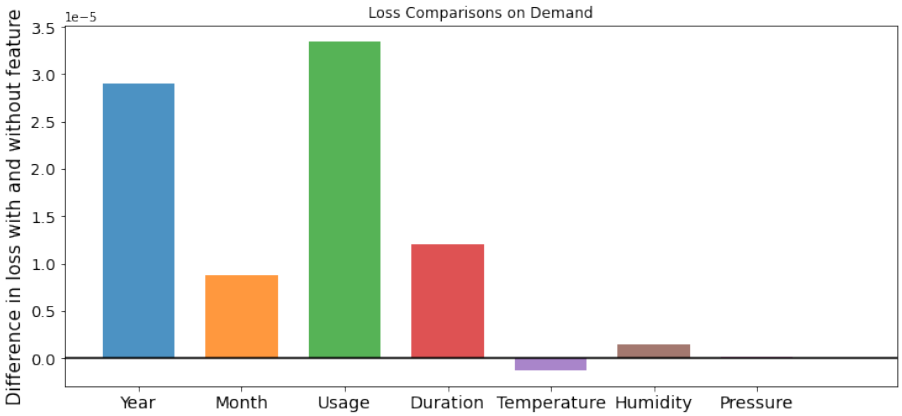

Figure 12: Changes to the Linear Regression's Loss on First Task from Removing each Input Feature.

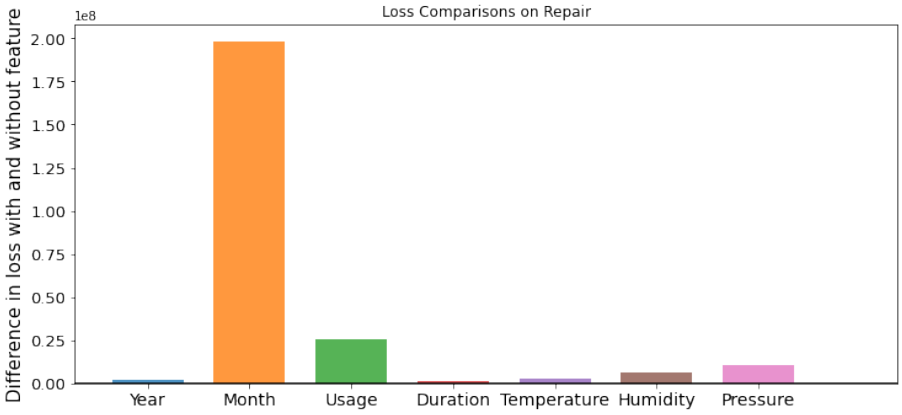

Figure 13: Changes to the Neural Network's Loss on Second Task from Removing each Input Feature.

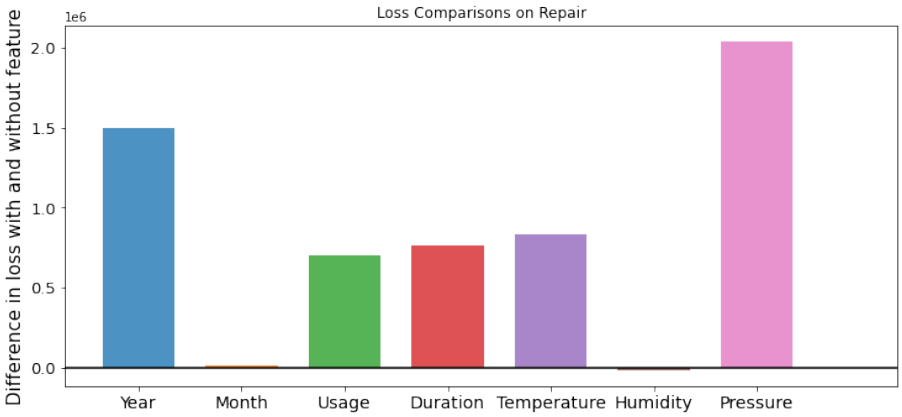

Figure 14: Changes to the Linear Regression's Loss on Second Task from Removing each Input Feature.

5.2 Limitations

One limitation of this experiment is that the models are trained on data from one company in one city, which means it can only be used for predicting values of that company and may not generalize to other bike sharing companies. In order for the models to be able to predict other bike sharing companies' values, the models should be trained on that company's data and the weather data of their region. Also, training on other data sets might require some adjustments to the architecture based on the structure and complexity of the data.

Another limitation is that the models' predictions of the usage distribution are grouped into 10 clusters. This would allow the company to know the approximate location of where to distribute the newly repaired bikes, but not the exact station. To minimize the issue, the number of cluster could be increased or each station could be its own cluster. Doing so would provide more precise information; however, this could also negatively impact the robustness of the predictions.

Also, the data set the models were trained on contained few data points (only 68 data points in train set), which likely hindered the performance of the

models. For example, as seen in figure 2, the target values are extremely noisy, so it would take more data points for the models to learn a good fit. There are not many solutions to this limitation other than to wait for some period of time for Citi Bike to gather more data points.

6 Conclusion

In this paper, we used neural networks and linear regression to predict the number of bikes Citi Bike has to repair and its distribution of users across stations. We analyzed these two regression techniques and suggested which technique to use for each task. Also, we examined the effects of detailed input data and the removal of individual input features on the prediction accuracy of the models. Results show that neither detailed inputs nor the removal of input features had much effect on linear regressions' accuracy; as for the neural networks, detailed input data improved the models' performance on the loss function up to 25-fold and the removal of input features may increase the loss up to 125-fold.

The experiment mentioned in this paper could be ran by other bike sharing companies by using their own data set. We believe that our work could help Citi Bike and other bike sharing companies better manage their budget and distribute their bikes more efficiently at the start of the month.

7 References

[1] A. Taylor, "The bike-share oversupply in china: Huge piles of abandoned and broken bicycles." https://www.theatlantic.com/photo/2018/03/bike-share-oversupply-in-china-huge-piles-of-abandoned-and-broken-bicycles/556268/, Mar 2018.

[2] G. Dutta, "Bike sharing : Multiple linear regression." https://www.kaggle.com/gauravduttakiit/bike-sharing-multiple-linear-regression/notebook, 2020.

[3] M. Uribe, "Applied exploratory data analysis, bike-sharing. the power of visualization, python." https://towardsdatascience.com/applied-exploratory-data-analysis-the-power-of-visualization-bike-sharing-python-c5b264 Jan 2020.

[4] "Index of bucket tripdata." https://s3.amazonaws.com/tripdata/index.html.

[5] "Past weather in new york, new york, usa." https://www.timeanddate.com/weather/usa/new-york/historic?month=1&year=2013.

[6] Glosser.ca, "Colored neural network." https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg, Feb 2013.

[7] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*, pp. 65–93, Elsevier, 1992.

[8] W. S. Sarle *et al.*, "Stopped training and other remedies for overfitting," *Computing science and statistics*, pp. 352–360, 1996.

[9] M. Tripathi, "Underfitting and overfitting in machine learning." https://datascience.foundation/sciencewhitepaper/underfitting-and-overfitting-in-machine-learning, Jun 2020.

[10] A. Kumar, "Overfitting and underfitting represented using model error vs complexity plot." https://vitalflux.com/machine-learning-terminologies-for-beginners/, Dec 2020.

[11] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[13] K. Gurney, *An introduction to neural networks*. CRC press, 2018.

[14] G. F., "How bike-sharing works." https://www.economist.com/the-economist-explains/2017/10/03/how-bike-sharing-works, Oct 2017.

[15] K. Bryce, "The many benefits of bike sharing programs." https://www.commuteoptions.org/the-many-benefits-of-bike-sharing-programs/.

[16] "Citi bike monthly operating reports." https://www.citibikenyc.com/system-data/operating-reports/.

8 Acknowledgement

A few years ago when I was on my way to school, I saw a pile of damaged bikes on the side of the road. Bike sharing has been more and more popular recently, but it also led to more damaged and abandoned bikes. I thought that bike sharing companies must have a hard time repairing or making new bikes, so I wanted to help the bike sharing companies using my knowledge in machine learning.

Firstly, I would like to thank Sam Saarinen for teaching me the basics of machine learning and giving useful and detailed feedback on my drafts. Without him, this paper would not be possible.

Next, I would like to thank Mr. Beatty, a computer science teacher in my school, for igniting my passion for computer science and programming. He also offered me help throughout the writing process.