A Deep Learning Based Analysis of the Toronto Bikeshare Ridership Dataset

Montgomery Gole
Department of Computer Science at
Toronto Metropolitan University
Toronto, Canada
mgole@ryerson.ca

***Abstract*—Toronto Bike Share's open source dataset has the potential to describe, and model the system's usage due to its large amount of data. It is interesting to wonder how the dataset may perform when deep learning is applied to specific user problems. By using data cleaning techniques, it was found that the dataset is not very clean in terms of its accuracy of trip observations, and leaves it futile to predict the amount of bikes at a station. Furthermore, in an attempt to predict trip duration, it is found that doing so with deep learning, specifically deep regression, one can accurately predict the actual duration label of a trip in minutes. This results in an average error of ~4 minutes and 48 seconds, but an error of 1.8 is found when predicting the End Time of a trip. Also, the inclusion of weather data does not seem to affect these predictions. Finally, more work is needed to ensure the robustness of measurements (ex. converting the error of the End Time label to actual units of time), and to find accurate station bike amount data.**

Keywords—deep learning; neural network; bikeshare; open source data; exploratory analysis; Toronto Bike Share

I. Introduction

Toronto Bike Share's yearly user, and trip amount has been growing steadily since 2016 [1], accounting for 3.57 million rides in 2021. Toronto Bike Share (TBS) collects data on each trip where one of their bikes is used, and shares it under an Open Government Licence [2]. From January to September of 2022, there were 3,620,479 rides observed in the dataset. As the dataset holds so many observations, it may have the potential to accurately describe, and model usage of the TBS system. This may lead one to beg a few questions regarding the dataset. The first question is (RQ 1) what are the characteristics of the TBS dataset? Therefore, this paper first explores the dataset, attempting to find simple patterns within the dataset, and acts as preliminary thought towards the second question (RQ 2) where this paper tries to find some implications of deep learning on the dataset, specifically, what are the implications which deep learning has on the TBS dataset? To answer this question, this paper strives to predict how many TBS bikes are present at a given station at a certain time, and to predict the duration of a TBS bike trip, both using deep learning techniques. The ability to predict these features of the TBS dataset is important. Regarding predicting the amount of bikes present at a certain station at a certain time, it could potentially solve a problem faced by TBS users. This problem is seen when a user is unable to accurately predict–at a given time in the future–whether they will be able to take a bike from a certain TBS station, or dock it at a certain TBS station. This is because the station may have no available bikes or docks, respectively. Said problem could disrupt their commute, and plans, leading the user to travel to different bikeshare stations while attempting to find one with suitable conditions for their bike's retrieval or docking. Currently, there is no non-real time solution for this problem in Toronto other than a human using their own intuition/estimation. While the problem of estimated time of arrival is not new, or unsolved, it is interesting to see whether the TBS data, with some augmentation and when applied with a neural network, can give accurate estimations, especially due to the fact that there is no actual route information given other than each trip's departure location, and arrival destination.

II. Literature Review

An exploratory data analysis of the TBS network, Zhang [3] found that users ride more frequently when the weather is more conducive of a safe trip. Not only do rides increase during the warm months in Toronto, but also during times which have lower fog/haze, precipitation, wind speeds. Regarding spatial data, they found that trips often occur closer to Toronto Transit Commission (TTC) subway stations. They attempted to predict how many trips would take place during a given hour via linear regression with poor results, concluding that they should use a non-linear regression method to do so.

Based on car traffic data, Wang, et al.–researchers from the ride sharing service, Didi Chuxing [4]–present a machine learning approach which sees the problem of estimated time of arrival (ETA) as a "spatial-temporal" regression problem. The data which they implement holds features that relate to: spatial information based a on the route with a given vehicle with travel to get from one point to another, temporal information (departure time which includes the year, month and day, a holiday indicator, and rush hour indicator), traffic information, personalized information(vehicle make, driver profile, rider profile), and

augmented information like weather data, or road restrictions. They use MAPE (mean absolute percentage error) as a loss function to match their assumption that a user's expectation of an ETA's accuracy is inversely proportional to the actual duration of the trip (ex. A user may be satisfied with a 55 minute ETA being incorrect by +- 5 minutes, but not with a 10 minute ETA being +- 5 minutes).

VE, et al. attempt to find predict the duration of bike share trips within Seoul, Korea[5]. To do so, they compare the linear regression, gradient boosting machine, K nearest neighbour, and random forest machine learning algorithms on their problem. While preprocessing their data, they added weather data like temperature, precipitation, solar radiation, snow fall, and ground temperature—as it is stated in the paper that "weather patterns information is considered a primary determiner in predicting the duration of the bike rental trip", as well as haversine distance between the "dropoff", and "pickup" points to each trip in the set. They found the highest correlation between the haversine distance feature, and duration label. Their random forest approach yielded the best testing results with a mean absolute error (MAE) of 2.92 minutes.

III. Methodology

Each of this paper's research question needs the TBS dataset to be cleaned in a certain way. Therefore, the methods for each research question are different, leading this section to describe the preprocessing, and experimentation regarding each question in their own separate subsections. The preprocessing sections include showing which features were removed, created, and how they were augmented to fit each problem. While the experimentation takes place in the Analysis and Results sections.

A. What are the characteristics of the TBS dataset?

The data set is split into *n* comma separated value (CSV) files, where *n* is the amount of months since, and including January 2020. For each file, it supposedly contains *z* observations, where *z* is the amount of rides which occurred in the file/month. Each row in the dataset is a specific trip taken by some user. A row is formatted as seen in Table 1.

TABLE I.

Column Name	***Column Type* (units or values)**
Trip Id	Integer
Trip Duration	Integer(seconds)
Start Station Id	Integer(7000-7681)
Start Time	DateTime
Start Station Name	String
End Station Id	Integer(7000-7681)
End Time	DateTime
End Station Name	String
User Type	String(Casual Member or Annual Member)

Row schema from TBS dataset. (Transposed for readability)

Preprocessing: From January-September of 2022, TBS observed 3,620,479 rides [2]. However, since accurate station information for each ride is vital for this paper's own accuracy, any ride without a proper Start Station, or End Station, were dropped. An improper Station means that instead of an observation This led to 242,313 observations being removed from the dataset.

Analysis: Regarding TBS usage by time, the month, and hour are the the two most influential temporal metrics on how many users take a ride. This can be seen in **Figure 1-4**, which show TBS ride amounts with respect to the month, hour, minute of the hour, and day of the week.

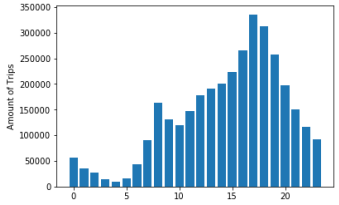

Fig 1. TBS rider amount by hour of day (0-23)

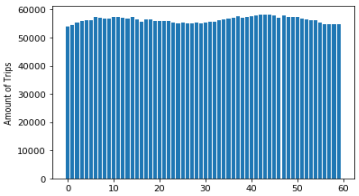

Fig 2. TBS rider amount by minute of hour(0-59)

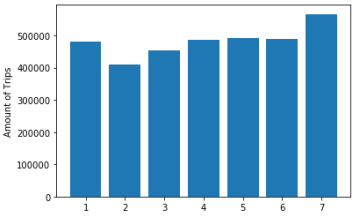

Fig 3. TBS rider amount by day of the week (1-7)

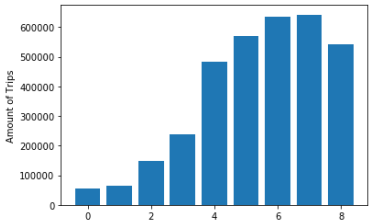

Fig 4. TBS rider amount by month of year(0-8)

B. *Can one estimate how many bikes are present at a given station at a certain time?*

Preprocessing: The algorithm referenced in this section is seen in **Figure 5.** To predict the amount of bikes present at a bikeshare station, one must have a label for this data in their operating dataset. As there are no hypothetical *bikesPresent* labels (one for the start station, and one for the end station per observation) in the TBS dataset, this paper has attempted to create them. Conducting preliminary testing only on the January 2022 subset of the TBS Ridership dataset. This was done by: setting a bike_amount counter for each station. Then iterating through the dataset by oldest trip to newest trip for each station's appearance in an observation, adding 1 to the counter for each trip which had a start station as the current bike_amount's station, and subtracting 1 from the counter for each trip which had an end station as the current bike_amount's station. Finally, appending the given value to the observation's new endStation_bikeAmount or startStation_bikeAmount column respectively.

An issue with this approach is that one would at least need to know the amount of bikes at each given station the first time they appear in the given dataset, initializing bike_amount to be equal to this value. A potential solution to this problem would be to set bike_amount to 0 for each station, conduct the algorithm described above, and assuming that each station will actually have 0 bikes, and reach its maximum amount of bikes in it at some point in the subset, offset each bikeAmount by the minimum value. Effectively setting the minimum value to 0, and the maximum value to the amount of bikes allowed at the station, producing the true bike amount values in the dataset.

Analysis: N\A

Results: Interestingly, the approach above results in the set of each bikeAmount column for every station having a minimum value of 0, but a maximum value of 125, which is larger than 56–the highest capacity of a bike share station in the network [6]. As seen in **Figure 7,** some bike share stations are seen to eventually have more bikes than their maximum capacity. Furthermore, it was found that 28% of bike share stations supposedly contained more bikes than docks, which is not possible. While this is probably an issue with the TBS dataset but it is currently unknown why this incorrect measurement occurs. A single missing, or false trip may greatly impact the quality of a prediction for the amount of bikes at a station, therefore it is seen as unnecessary to predict the amount of bikes at a station with the given TBS dataset. This is due to a lack of accurate training, and testing data available. This issue will be discussed more in the *Future Work* section.

```
function offsetAmounts(arguments: bikeAmount):
    return bikeAmount+((-1)*minVals[station])

For each station:
    # Initialize diffs and mins
    Set counts["startAmount"][station] to 0
    Set counts["endAmount"][station] to 0
    Set minVal[station] to 0
    Set basePrediction to 0
    # Get current station data
    Set stationOnly to dataframe to trips which reference station
    # Iterate through stationOnly data
    For each row in stationOnly:
      If index, row["Start Station Id"] is equal to station
        pred minus 1
        Add pred to counts["startAmount"][station]
        Set row["start_bikeAmount"] to pred
      If row["End Station Id"] is equal to station
        pred plus 1
        Add pred to counts["endAmount"][station]
        Set row["end_bikeAmount"] to pred
    # Get minimum value of counts
        Set minVal[station] to minimum of (counts["startAmount"][station] and counts["endAmount"][station])
    Set the start_BikeAmount of each row where Start Station Id is equal to station
    Set the end_BikeAmount of each row where End Station Id is equal to station
    # Offset start and end bike amounts
        Set each start_BikeAmount referenced for station to call(offSetAmounts(each start_BikeAmount))
        Set each end_BikeAmount referenced for station to call(offSetAmounts(each end_BikeAmount))
    Reset basePrediction to 0
```

Figure 6: Pseudocode for algorithm to measure amount of bikes at a given station

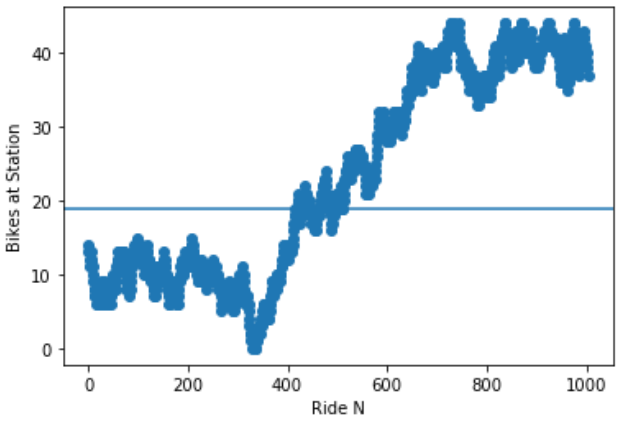

Figure 7: Station 'Bay St / College St (East Side)' incorrectly having more bikes than its maximum capacity

C. *Can one predict the duration of a TBS bike trip with a neural network?*

The procedure performed to determine whether one can predict the duration of a TBS bike trip with a neural network begins with data preprocessing to fit the problem. The problem is a regression problem, as the duration of a trip has a large range, and we would like to find a function that can represent *f*(TBS dataset columns)=Trip Duration. Once preprocessing has been completed, said regression analysis was completed with a deep neural network trained and fit with Keras. The model was first trained and tested with a portion of the dataset, specifically rides from January of 2022.

Preprocessing: As each station's ID (ex. 7000 for Wellesley Station Green P) doesn't have a clear relation to its actual location relative to the other BikeShare stations, it was deemed necessary to find the coordinates of each station, given its name. While an open source solution to this problem would seem to suffice, OpenStreetMap (OSM)[7], does not have location data on a considerable number of intersections in Toronto. This is significant because most TBS station names are intersections listed as "North-South Street Name / East-West Street Name", for example, the station called,

Adelaide St W / Strachan Ave, is not listed in OSM. After attempting to work around this problem, it was found that Google Maps' Geocoder API [8] gives results for each station with intersection, therefore it was used to find the coordinates of each station. The coordinates added to each observation were then used to create a new column in the dataset which describes the distance between a trip's departure point and end destination. Distance was measured by calculating the length of the geodesic line between the two points, in metres using *geopy*'s geodesic module[9]. Then, the startCoordinate and endCoordinate columns were be removed, as their necessary information to the problem is in the *distance* column.

In order allow a neural network's interpretation of the *Start Time*, and *End Time* feature of the dataset, they were split into 4 features: *Start\End Month*, *Start\End Day*, *Start\ End Hour*, and *Start\End Minute.*

Further cleaning of the dataset involved removing outlier trips in the dataset. The outliers were seen in the trip duration, and distance columns. As TBS only allows a ride to be 30 minutes before a user is charged $4 per extra 30 minutes. As seen in **Figure 8**, rides with a trip duration > 50 account for ~0% of the rides seen in the January-September 2022 TBS subset, therefore, rides longer than 50 minutes were removed from the dataset. Similarly, all rides which were shorter than 1 minute were removed from the dataset, as these are probably false trips[9].

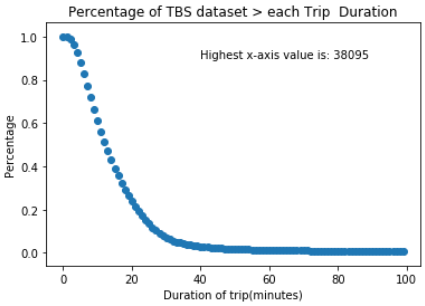

Figure 8: The percentage of trips with a duration greater than X minutes

A similar method was applied to the distance column of the dataset. As seen in **Figure 6**, the longest trip taken was ~26 KM(kilometres), and rides with distance > 8 KM account for 0.5% of the subset. This led to removing all trips longer than 8,000 metres. While it may seem intuitive to remove all trips with a distance of 0 metres, as it is most likely a false trip, it instead could mean that the user had a real trip, but docked their bike at their departure station. Since these trips account for ~100,000 observations out of ~3,000,000, and they all meet the trip duration criteria given above, they were kept in the subset.

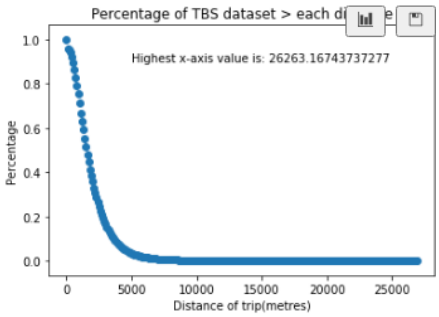

Figure 9: The percentage of trips with a distance greater than X metres

Finally, by calling the Meteostat[10] API, Toronto weather data for each day observed in the TBS subset was appended. This weather data included in the final cleaned, and wrangled dataset includes: average temperature (celcius), total precipitation (millimetres), snow depth (millimetres), wind direction (degrees), windspeed(KM\H), and air pressure(hPa). One issue with this data is that snow is not always measured on a given day, and is represented as not-a-number (NaN) in the data. All snow data which is NaN isn't removed, but instead changed to 0.

The final schema of a row for the dataset to be analyzed can be seen in Table 2.

TABLE II.

Column Name	***Column Type*(units or values)**
****Trip Duration***	Integer(mins)
Start Month	Integer(1-9)
Start Day	Integer(1-7)
Start Hour	Integer(1-24)
Start Minute	Integer(1-60)
****End Month***	Integer(1-9)
****End Day***	Integer(1-7)
****End Hour***	Integer(1-24)
****End Minute***	Integer(1-60)
Distance	Float(metres)
*****Average Temperature***	Float(celcius)
*****Precipitation***	Float(millimetres)
*****Snow Depth***	Float(millimetres)
*****Wind Direction***	Integer(degrees 0-359)

Column Name	***Column Type*(units or values)**
****Trip Duration***	Integer(mins)
*****Wind Speed***	Float(KM\H)
*****Pressure***	Integer(hPa)

*Row schema from TBS dataset. (Transposed for readability) *: Potential Label, **: Weather Data*

Analysis: The actual experimentation consists of performing a regression with a neural network to predict the subset's (to be referred to as the dataset in this section) Trip Duration based labels. The regression was done enough times to satisfy the multiple combinations, and permutations of the dataset's features. As seen in figure n, there are 4 of these variations. They can be split into two main groups: those with weather data, and those without; and further split into predicting the End Time label, or the Trip Duration label. The inclusion of the End Time label instead of just analyzing the trip duration label, is due to the fact that one could find the trip duration by simply subtracting the Start Time by the End Time. In order to have a relative baseline for the performance of each regression model, the Random Forest's testing performance from *Seoul bike trip duration prediction using data mining techniques*[5], was compared against. This comparison consisted of the matching metrics being collected: Root Mean Square Error (RMSE), Mean Absolute Error(MAE), and goodness of fit (R^2).

Results: As seen in Table 3, with respect to MAE and the End Time label, a neural network based regression, performed on the TBS dataset, performs better than the Random Forest presented in *Seoul bike trip duration prediction using data mining techniques*[5]. However, the baseline metrics outperform each variation of data with a neural network with respect to the R^2, and RMSE values. An interesting observation is the fact that the inclusion of weather based data does not significantly affect the neural network's performance metrics.

TABLE III.

Model & Label	***Weather Data***	***RMSE***	***MAE***	***R^2***
Random Forest Duration(minutes)[5]	True	6.25	2.92	0.93
Neural Network Duration(mins)	True	7.73	4.57	0.19
Neural Network(End Time day, month, hour, minute)	True	6.79	1.8	0.82
Neural Network Duration(mins)	False	8.06	4.80	0.13
Neural Network(End Time day, month, hour, minute)	False	6.78	1.8	0.82
Random Forest Duration(minutes)[5]	True	6.25	2.92	0.93

Performance metrics of each data variation and model, each row after the first is related to the TBS dataset. RMSE, and MAE are better when closer to 0, while R^2 is better when closer to 1

IV. DISCUSSION & LIMITATIONS

The TBS usage data could be seen to match one's assumption about how often, and when humans go outside, and would use a bike. TBS is used more during the warmer months, weekends, and afternoon hours. An interesting theme found in the first section of this paper, and continues on, is the accuracy of the TBS dataset. From the months of January-September of 2022, it saw ~240,000 or ~8% of its observed trips to be inaccurate, specifically due to them not containing sufficient station data.

Said inaccuracy most likely affected the ability for this paper to accurately find the amount of bikes present at a given station, at a given time, as ~27% of stations tested on had a bike amount greater than its maximum capacity. However, it must be noted that it is possible for an issue with the algorithm used to find how many bikes are present at a given station, and it must be verified for its own accuracy.

When estimating the duration of a trip in the TBS network via regression with a neural network, it was found that while it can be done accurately via ETA estimation, and performs a bit better than a Random Forest algorithm on similar data, the error measured for the end time labels doesn't just reflect the error in the minutes of the end time but instead each feature of the end time label, making it possible for predictions to be off by hours, days, or months, instead of minutes. Furthermore, the actual estimation of trip duration performed much worse than the baseline presented. Both labels' prediction were insignificantly impacted by the inclusion of weather data.

It is clear that while deep learning can be used with fair accuracy on the duration of trips on a TBS ride, less complex Machine Learning Algorithms should be implemented on the dataset to show a deep neural network's relative performance on the dataset.

V. FUTURE WORK

The quality, and accuracy of results found in this paper could be increased by using the *City Bikes* [6] API real-time data to continuously collect data on TBS station bike amounts, and performing an analysis on said data, instead of the TBS open source data. It would be beneficial to collect this data over a year to increase its accuracy. Furthermore, while the neural networks trained, and tested on for the duration prediction portion of this paper were fairly large, it could be beneficial to increase their breadth, and depth. This feat was not possible with this paper's experimenter's hardware to due computation, and time constraints. Also, the End Time label errors should be adjusted according to their units of time, not

just their values. Finally, it would be interesting to see how a machine learning based(without neural networks) replication of Seoul bike trip duration prediction using data mining techniques would perform on the TBS open source dataset.

VI. Conclusion

Most usage of the Toronto Bike Share(TBS) network occurs during the summer, and 3-8PM. The high amount of usage makes its quality of user experience a significant facet of the system. Said quality could be greatly improved if one was able to accurately predict how many bikes will be at a certain station, at a certain time. However, this is not possible to do so with the TBS dataset for reasons which are not easily assumed, and should be further explored. Overall, deep learning can clearly be used to produce accurate estimates of bike trip duration within the TBS network, but the topic of deep learning's implications on the dataset should be further explored.

[1] Bike Share Toronto first quarter (Q1) 2022 update," Toronto Legal Documents, 04-Feb-2022. [Online]. Available: https://www.toronto.ca/legdocs/mmis/2022/pa/bgrd/backgroundfile-199512.pdf.

[2] "Bike Share Toronto Ridership Data," CKAN. [Online]. Available: https://ckan0.cf.opendata.inter.prod-toronto.ca/en/dataset/bike-share-toronto-ridership-data. [Accessed: 23-Dec-2022].

[3] S. Zhang, "Toronto Bike Share Data Analysis," Medium, 16-Apr-2021. [Online]. Available: https://shirleyzhang2.medium.com/toronto-bike-share-data-analysis-943d5810c717.

[4] Z. Wang, K. Fu, and J. Ye, "Learning to estimate the travel time," Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Jul. 2018.

[5] S. V E, J. Park, and Y. Cho, "Seoul bike trip duration prediction using data mining techniques," IET Intelligent Transport Systems, vol. 14, no. 11, pp. 1465–1474, Nov. 2020.

[6] "Citybikes API documentation," Documentation | CityBikes API. [Online]. Available: https://api.citybik.es/.

[7] OpenStreetMap. [Online]. Available: https://www.openstreetmap.org/.

[8] "Google Maps API: Geocoding," Google. [Online]. Available: https://developers.google.com/maps/documentation/geocoding/overview

[9] "Welcome to GeoPy's documentation! ," Welcome to GeoPy's documentation! - GeoPy 2.3.0 documentation. [Online]. Available: https://geopy.readthedocs.io/en/stable/.

[10] "The weather's record keeper," Meteostat. [Online]. Available: https://meteostat.net/en/.