# Predicting the Number of Ended Trips by Station in a Bike-Sharing System

Filipe Gonçalves
*DETI*
*Universidade de Aveiro*
Aveiro, Portugal
filipegonc@outlook.com

***Abstract*—Bike-sharing systems (BSS) are ever growing and becoming more popular by the day. Consequently, managing one is becoming more and more complex. From identifying possible problems with the bicycles, to redistribute them to their respective stations and place, from understanding the traffic flow to predict the hours of more affluence, the job of an operator of a system like this is even more difficult. The purpose of this paper is to facilitate the work of an operator, by building a predictive system based on stations and hours of a day. As such, it will be discussed different methods and compared different results, to reach a final solution.**

***Index Terms*—Bike-sharing, Bike-Sharing system, bicycle, Linear Regression, Deep Learning, feature selection, K-Fold Cross Validation, Long-Short Term Memory**

## I. Introduction

BSS typically operates through a network of docking stations strategically located throughout a city. These stations serve as pick-up and drop-off points for bicycles, and users can rent a bicycle from one station and return it to any other station within the system predisposed area.

The main challenges these systems have are the possible vandalism, stealing or any other malicious intent towards the bicycle; redistributing the bicycles into all stations, and knowing which stations should the bicycles be going to; and, finally, their operating system, which needs to respond in real time to any type of alerts and problems the system and any bicycle can have.

In order to try to ease the work done by any operator and the redistribution management, there is a need to build predictive systems, which will give a rough estimate of which stations should be delivered more bicycles, at each given hour.

The prediction system built and analyzed will predict the number of bicycles that will end a trip in a given hour, for a given station, and therefore any operator should be able to understand, based on these results, to which station the redistribution team should deliver all their bikes.

## II. State of Art

A predictive system like this is not at all new, with many different methods, models and ways to execute and create them. There are examples with Deep Learning methods as well as Machine Learning ones.

We can have models using Random Forest Regression to combine a large enough number of decision trees built on bootstrapped samples of the dataset and take the mean of the outputs from each different decision tree to make its predictions, as seen in the work made by Zhejiang University in China [4].

There are also different types of system to predict the traffic clustering in a BSS, using different stations clusters, similar to the ones explained in more detail later, and understanding the time span of each trip, as seen in the work made by The Hong Kong University of Science and Technology in Hong Kong [5].

Another work made by School of Software Engineering with the College of Computer Science and Technology in China [6] created a hybrid system, which uses simultaneously a Normalisation process, K-Fold Cross Validation and a Support Vector Machine (SVM) Predictor.

However, the paper wrote by University of Bordeaux in France [7] is the closest to what we are trying to build, which tries to predict the bicycle usage up to one day. It compares different Regression models such as Ridge Regression, Adaboost Regression, Support Vector Regression, Random Forest Regression and Gradient Tree Boosting Regression.

## III. Models

### A. Dataset Description

The data used was created using the data from the Kaggle community free datasets, and is composed of two main projects: a general bicycle sharing system with trips made by the bicycles by hour, for more then a full year [1]; and a more specific dataset with more than 700 station descriptions [2].

We can see the datasets and their features in the next two figures, Figure 1 and Figure 2:

| | instant | dteday | season | yr | mnth | hr | holiday | weekday | workingday | weathersit | temp | atemp | hum | windspeed | casual | registered | cnt |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 2011-01-01 | 1 | 0 | 1 | 0 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.81 | 0 | 3 | 13 | 16 |
| 3 | 2 | 2011-01-01 | 1 | 0 | 1 | 1 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.8 | 0 | 8 | 32 | 40 |
| 4 | 3 | 2011-01-01 | 1 | 0 | 1 | 2 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.8 | 0 | 5 | 27 | 32 |
| 5 | 4 | 2011-01-01 | 1 | 0 | 1 | 3 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 | 0 | 3 | 10 | 13 |
| 6 | 5 | 2011-01-01 | 1 | 0 | 1 | 4 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 | 0 | 0 | 1 | 1 |
| 7 | 6 | 2011-01-01 | 1 | 0 | 1 | 5 | 0 | 6 | 0 | 2 | 0.24 | 0.2576 | 0.75 | 0.0896 | 0 | 1 | 1 |
| 8 | 7 | 2011-01-01 | 1 | 0 | 1 | 6 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.8 | 0 | 2 | 0 | 2 |

Fig. 1. Day Datasets [1]

| | Station ID | Capacity | Latitude | Longitude | Station Name |
|---|---|---|---|---|---|
| 0 | 1 | 19 | 51.529163 | -0.10997 | River Street , Clerkenwell |
| 1 | 2 | 37 | 51.499606 | -0.197574 | Phillimore Gardens, Kensington |
| 2 | 3 | 32 | 51.521283 | -0.084605 | Christopher Street, Liverpool Street |
| 3 | 4 | 23 | 51.530059 | -0.120973 | St. Chad's Street, King's Cross |
| 4 | 5 | 27 | 51.49313 | -0.156876 | Sedding Street, Sloane Square |
| 5 | 6 | 18 | 51.518117 | -0.144228 | Broadcasting House, Marylebone |

Fig. 2. Stations Dataset [2]

The final dataset is generated by simulating the trips into a small cluster of 40 stations, with all of the features.

### *B. Evaluation of Periodicity*

In the case of a medium sized dataset like the one created and because we are working with time tables, there is a need to search for periodicity and automatic predictability.

Finding periodicity will give our model a more accurate evaluation, and in case of finding automatic predictability, the model will not only get more efficient and more accurate, but will also decrease its loss and error.

Firstly it was searched by periodicity based on weeks by a full year, in Figure 3:

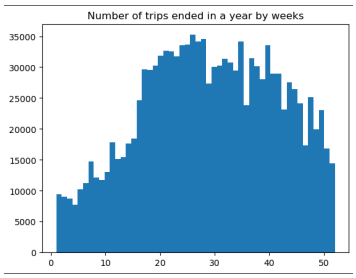


Fig. 3. Number of trips in a year by weeks

The conclusions are simple - as the year advances to its midterm, the summer season, there are a lot more uses of the system, in contrast to the early stages of the year, with less than a third usages in some weeks to the maximum.

Then, it was search for any type of periodicity between weekdays, by every hour, for every week, in Figure 4:

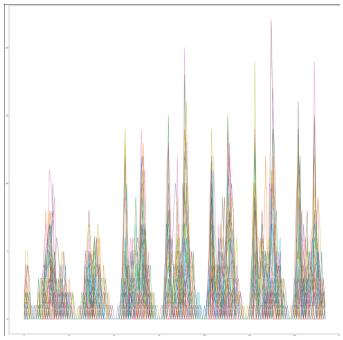

Fig. 4. Number of trips in a year by weekday

What was found out was that each day starts with a decline of trips and the peak is always between the 11 and 17 hours of the day for the weekend, and in the end of the day for a working day. It is also interesting to see, that in a normal working day, we get two maximum values, corresponding to the start and end of a business day.

It was after decided to search even deeper and try to find periodicity between weekdays, in Figures 5 and 6:

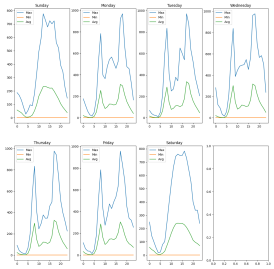

Fig. 5. Number of trips in a year by weekday by hour - separated

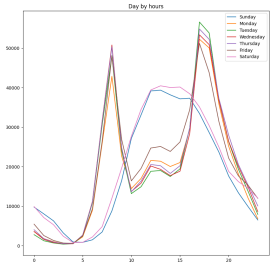


Fig. 6. Number of trips in a year by weekday by hour - together

In this case, the conclusions are exactly the same as before - during a working day, we get local and global maximums at the start and the end of the days, and in the weekends we get maximums around the afternoon.

Finally, there was a need to look into periodicity between stations and try to even more minimize errors by decreasing the number of stations in our initial cluster:

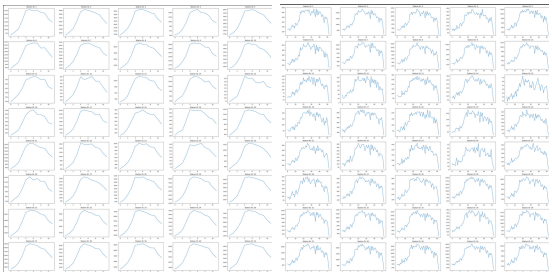

Fig. 7. Number of trips by month and week for every station

As can be seen in Figure 7, there seems to be some sort of periodicity between some stations, but its not understandable which stations contain similarities, and which stations should be implemented into which cluster.

For this purpose, we compacted every station into one graph, separated by trips by month and by week. The final result can be seen in Figure 8.

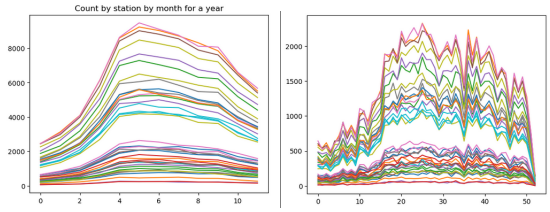


Fig. 8. Number of trips by month and week for every station - together

As consequence, and as can be seen in Figure 8, the initial cluster composed of 40 stations, should be separated into two different clusters, with stations of different characteristics in each.

Figures 9 and 10 show the two different clusters created:

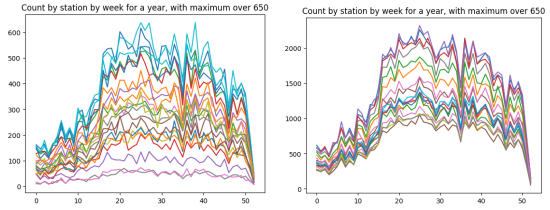


Fig. 9. Cluster 1 and Cluster 2 - By Week

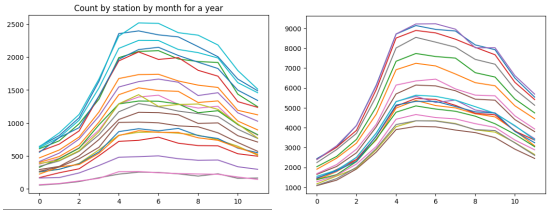


Fig. 10. Cluster 1 and Cluster 2 - By Month

With this, it was decided to use the Cluster 1, to train, test and validate our models, as it is the cluster with less difference between values.

### C. Regression Models

While using a Deep Learning model over a simple Linear Regression is normally better, we decided to test and compare the results using both Linear Regression and Deep Learning models.

As to predict the number of trips that should end for the following hour in a certain station, we need to at least have: the number of trips that ended in the more latest hour; the station identification; and the day this is occurring.

Nevertheless, sometimes the model can't accurately predict the result with only this information, and to battle this problem, we decided to create different models, each with the addition of a new feature as follows in Table I:

| Method | Features |
|---|---|
| Method 1 | Count(t) |
| Method 2 | Method 1 + Count(t-1) |
| Method 3 | Method 2 + Count(t-1week) |
| Method 4 | Method 3 + Count(t-2week) |
| Method 5 | Method 4 + Count(t-3week) |
| Method 6 | Method 5 + Count(t-4week) |

TABLE I
METHODS USED FOR THE REGRESSION MODELS

Some valuable information we get from this table: **t** is the hour and **Count(t)** is a function that gives the number of trips that ended in the hour **t**; the spacing between features is of a week, based on the periodicity evaluated before; in a way to be more readable for the next subsections, whenever it will be mentioned any method, it will be by their naming, as stated in Table I.

To separate the data between training, testing and validation data, we used K-Fold Cross Validation, as it is a more real and validated way to split our dataset and train and test our models.

However, K-Fold Cross Validation can have a default way of splitting the dataset into folds, but when working with time, having a random shuffle and a random way to separate our trips is not the best method.

With the intention of validating the latest affirmation, the firstly thought model was separated into three different Linear Regression models: splitting the dataset by time; splitting the dataset by stations; splitting the dataset normally.

*1) Date Separation:* After training and testing the different models, the results for the mean squared error (MSE) for the date K-Fold Cross Validation split were the following, Figure 11.

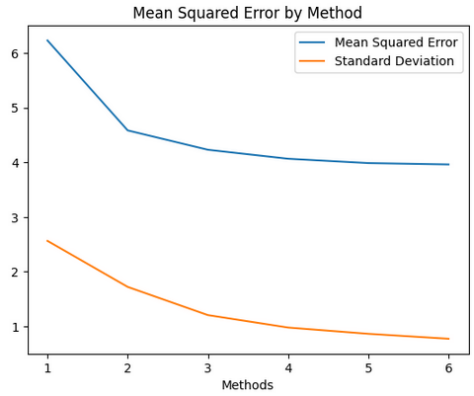


Fig. 11. Date Methods

*2) Stations Separation:* After training and testing the different models, the results for the mean squared error (MSE) for the stations K-Fold Cross Validation split were the following, Figure 12.

*3) Normal Separation:* After training and testing the different models, the results for the mean squared error (MSE) for the date K-Fold Cross Validation split were the following, Figure 13.

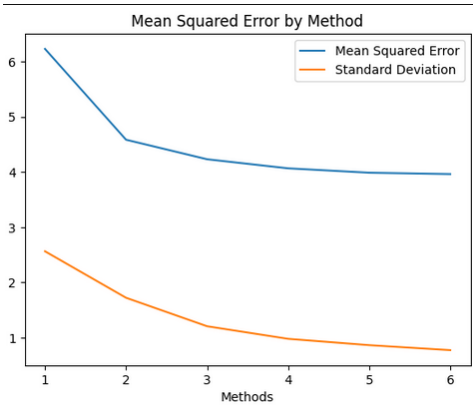


Fig. 12. Station Methods

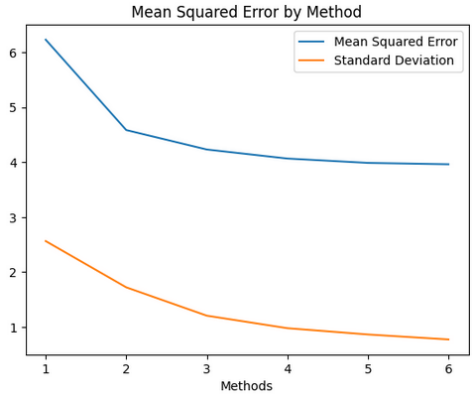


Fig. 13. Normal Methods

Comparing these separations and the general methods, we can conclude that using Linear Regression, the way we split the data is not at all useful, as the results are very very similar.

We can also say that in terms of a theoretical situation, the best method to choose from should be the last, Method 6, which is the one with the least standard deviation and consequently the least mean squared error. However, in a practical situation, the difference in MSE between Method 6 and Method 5 is not advantageous for the additional feature.

With this conclusions, we decided to implement the Method 5 into three Deep Learning models, with the same differences as the last ones, and comparing the cross validation separation, as well as the final results.

### *D. Feature Extraction and Selection*

Until now, there was no need to over complicate our models, which only used simple features delegated as essential. Still, to create and use the Deep Learning models, only using simple features will not get good improvements in the models, and as such knowing which features are the best and which are the worst to implement in the models is essential.

For this purpose, the features selected were based on the best correlation values between the not used features and the **Count(t)** feature.

The results were mostly the same for the different grouping styles used, with the exception of one or two features, and the mostly correlated being shown and explained in Table II:

| | |
|---|---|
| Temp | Temperature (ºC) |
| Temp Feeling | Temperature felt by humans (ºC) |
| Humidity | Humidity percentage (%) |
| WeatherSituation | Value to represent the seasons |
| Windspeed | Wind speed percentage (%) |
| Year | The year the trip was rode in |

TABLE II
FEATURES - DATE SEPARATION

### *E. Deep Learning Models*

The Deep Learning models created contained two hidden layers - one dense neural network and one Long-Short Term Memory (LSTM) layer with a hyperbolic activation function.

Each model was trained for ten fold splits, using group and normal K-Fold Cross Validation, where each method had 1 epoch and a default batch size of 1.

Comparing the results obtained before using Linear Regression and now using Deep Learning, for the best method, Method 5:

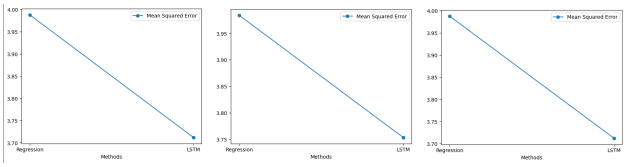

Fig. 14. Date Grouping, Normal Grouping and Stations Grouping - Comparison between Linear Regression and Long-Short Term Memory

As seen from the last figure, Figure 14, it can be concluded that using a neural network like LSTM is beneficial, compared to using a simple Machine Learning model like Linear Regression.

In all three cases it's best to also say that the time difference while training and validating is very different. While using Linear Regression can take up to mostly 3 to 5 minutes, using Long-Short Term Memory will take mostly 25 minutes.

Consequently, in theoretical terms we should be using the best method with the best error values, which is the Deep Learning methods, however for a practical solution, we should use the Linear Regression models, as, even though they have worse error values, having a model which needs almost half an hour to train is very dispendious.

With all this being said, we decided to use the LSTM method for all three separation models, as it should get better results with the addition of the new features.

The next methods will have the base features mentioned in Table III, with new ones being added for each new method.

| | | | |
|---|---|---|---|
| Station ID | WorkingDay | Hour | Count(t) |
| Count(t-1) | Count(t-1week) | Count(t-2week) | Count(t-3week) |

TABLE III
BASE FEATURES FOR THE DEEP LEARNING MODELS

*1) Date Separation:* For this separation method, there were created five different methods, each with different features added to the base features that all methods have. The methods can be seen by the Table IV.

| Method | Features |
|---|---|
| Method 1 | Temp |
| Method 2 | Temp + Temp Feeling |
| Method 3 | Temp + Temp Feeling + Humidity |
| Method 4 | Temp + Temp Feeling + Humidity + WeatherSituation |
| Method 5 | Temp + Temp Feeling + Humidity + WeatherSituation + Windspeed |

TABLE IV
METHODS USED FOR THE DEEP LEARNING MODELS - DATE SEPARATION

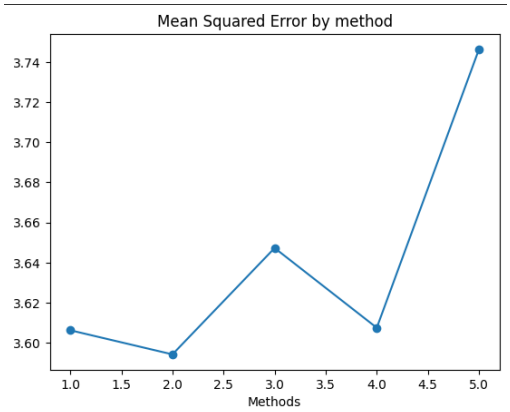


Fig. 15. Mean Squared Error by Method - Date Separation

Figure 15 shows the results for the MSE values each Method before mentioned achieved, and as can be seen Method 2 contains the lowest value. It also indicates that adding a new feature will not always improve the model and many times it will get a worse result with more features.

As mentioned earlier, for a theoretical approach it was decided to choose Method 2 for the reason stated earlier, however, in a practical manner the difference in MSE values between Method 1 and Method 2 is not as great as to be efficient to add a new feature.

*2) Stations Separation:* For this separation method, there were created eight different methods, with the same characteristics as before. The methods can be seen by the Table V.

| Method | Features |
|---|---|
| Method 1 | Temp |
| Method 2 | Temp + Temp Feeling |
| Method 3 | Temp + Temp Feeling + Humidity |
| Method 4 | Temp + Temp Feeling + Humidity + WeatherSituation |
| Method 5 | Temp + Temp Feeling + Humidity + WeatherSituation + Windspeed |
| Method 6 | Temp + Temp Feeling + Humidity + WeatherSituation |
| Method 7 | Temp + Humidity + WeatherSituation + Windspeed |
| Method 8 | Temp + Humidity + Windspeed |

TABLE V
METHODS USED FOR THE DEEP LEARNING MODELS - STATIONS SEPARATION

Figure 16 shows the results for the MSE values each Method before mentioned achieved, and as can be seen Method 3 contains the lowest value.

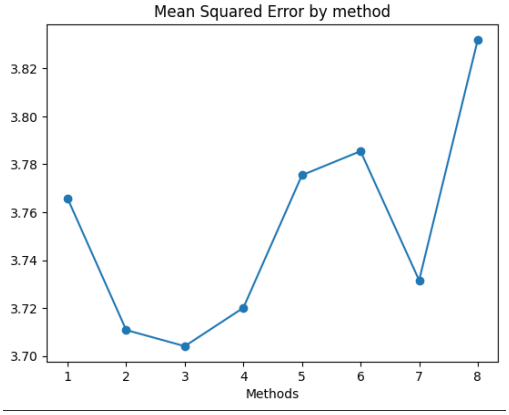


Fig. 16. Mean Squared Error by Method - Stations Separation

As mentioned earlier, for a theoretical approach it was decided to choose Method 3 for the reason stated earlier, however, in a practical manner the difference in MSE values between Method 2 and Method 3 is not as great as to add a new feature and maintain its efficiency.

*3) Normal Separation:* For this separation method, there were created eight different methods, with the same characteristics as before. The methods can be seen by the Table VI.

| Method | Features |
|---|---|
| Method 1 | Temp |
| Method 2 | Temp + Temp Feeling |
| Method 3 | Temp + Temp Feeling + Year |
| Method 4 | Temp + Temp Feeling + Year + Humidity |
| Method 5 | Temp + Temp Feeling + Year + Humidity + WeatherSituation |
| Method 6 | Temp + Temp Feeling + Humidity + WeatherSituation |
| Method 7 | Temp + Temp Feeling + Humidity |
| Method 8 | Temp + Temp Feeling + WeatherSituation |

TABLE VI
METHODS USED FOR THE DEEP LEARNING MODELS - NORMAL SEPARATION

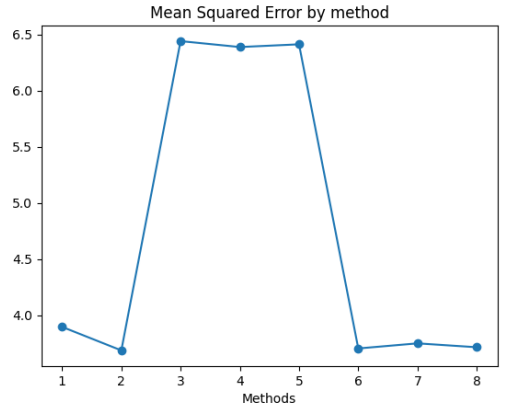


Fig. 17. Mean Squared Error by Method - Normal Separation

Figure 17 shows the results for the MSE values each Method before mentioned achieved, and as can be seen Method 2 contains the lowest value.

In this case, both for a theoretical or practical approach using Method 2 seems to be the best way, as it contains the lowest MSE value, and a big enough difference to say that it still is efficient with the addition of one feature.

## IV. Results

Comparing altogether the Linear Regression best model, the LSTM base model and the LSTM best method for each separation type, the Figures 18, 19 and 20.

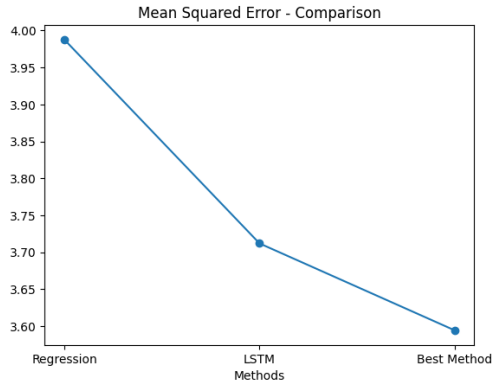


Fig. 18. Mean Squared Error Comparison - Date Separation

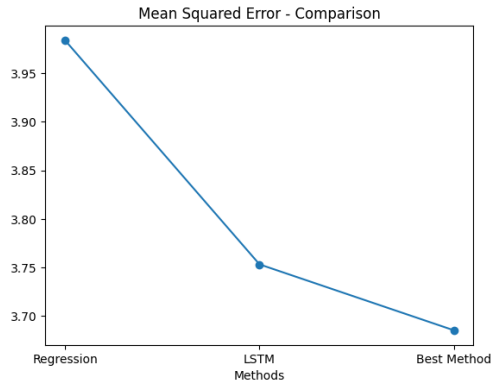


Fig. 19. Mean Squared Error Comparison - Normal Separation

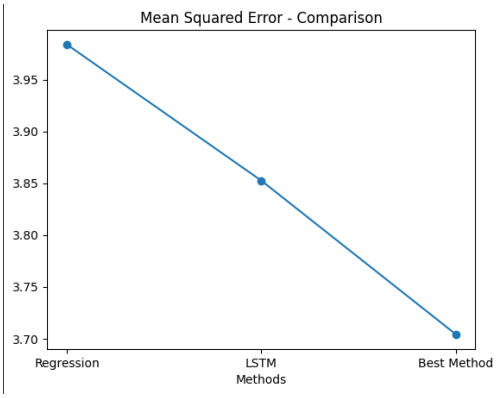


Fig. 20. Mean Squared Error Comparison - Stations Separation

As can be seen, the Regression models are the worse models, with the best method, in between 3.95 and 4.00 MSE. Consequently, using the same features with LSTM gets better results, but because of the earlier mentioned time span its not usable in practical projects. Nevertheless, using a specific set of new features, the results get better and better, with the biggest difference of 0.4 MSE to the Linear Regression models, using time split.

The question now is, if we train the model for enough time, does it get so much better that we should use LSTM instead of simple Linear Regression in a normal project?

For that purpose each method was executed for 30 epochs, with a batch size of 1 and for 10 K-Fold Cross validation splits, reaching the following results, as seen in the Figure 22.

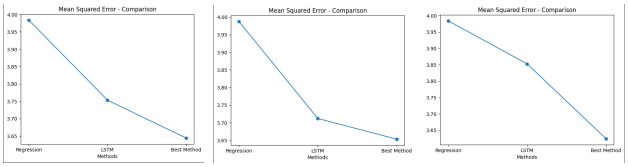


Fig. 21. Date Grouping, Normal Grouping and Stations Grouping - Comparison between Final LSTM Methods

The results are exactly what we expected - increasing the number of epochs will minimize the MSE for every model, until it reaches a certain point which can't decline any further.

The results were very satisfactory, as there were differences of 0.4 MSE in every separation form. However, each model took around 13 hours to compute the 10 K-Fold Cross Validations split with 30 epochs each, being way out of reach for any practical situation where we need to consistently train our models.

Maybe with a better Regression model the Regression methods could achieve better results. As talked before, like the work made by the University of Bordeaux in France [7], using Ridge Regression, Random Forest Regression or even Gradient Tree Boosting Regression could get us better results and improve our prediction system.

It's also important to mention, that our models could get up to a total of 34% of accuracy.

## V. Discussions

Comparing the separation methods between themselves, Figure 22, we can see that using a split based on stations is better than using any other kind of split, with differences close to 0.02 in the mean squared error values.

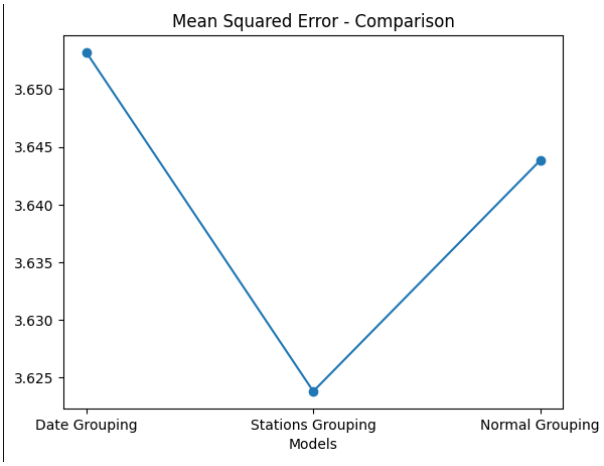


Fig. 22. Comparison between Final LSTM Methods based on split factor

## ACKNOWLEDGMENT

This work was inspired by the dissertation paper by Diogo Macedo [3]. I would also like to thank professor Pétia Georgieva for the help and mentoring during the build and analysis of the project. Finally, I want to thank the Kaggle community for creating the base datasets in this project.

## REFERENCES

[1] Akash Patel, "Rental Bike Sharing Dataset - Kaggle".
[2] Eden Au, "London Bike Sharing System - Kaggle".
[3] Diogo Macedo de Sousa, "Decision Support Service for Bewegen Bike-Sharing Systems".
[4] Zidong Yang, Ji Hu, Yuanchao Shu, Peng Cheng, Jiming Chen, Thomas Moscibroda, "Mobility Modeling and Prediction in Bike-Sharing Systems".
[5] Romain Giot, Raphaël Cherrier, "Predicting Bikeshare System Usage Up to One Day Ahead".
[6] Haitao Xu1, Jing Ying, Hao Wu1, Fei Lin, "Public Bicycle Traffic Flow Prediction based on a Hybrid Model".
[7] Yexin Li, Yu Zheng, Huichu Zhang, Lei Chen, "Traffic Prediction in a Bike-Sharing System".