Final Year Project

Foundations Report

Thomas Igoe

Student ID: 17372013

A thesis submitted in part fulfilment of the degree of

BSc. (Hons.) in Computer Science

Supervisor: Professor Tahar Kechadi

UCD School of Computer Science

University College Dublin

May 7, 2021

Table of Contents

Abstract

Bicycle sharing services are present in many different cities all over the world today, with more than 1,000 services being active in 2016 [1]. These services provide a carbon friendly public transport method for the cities they are active in, however a common problem with these services is managing station population. Most traffic is flowing in the same direction at the same times (rush hours) meaning that manually relocating bikes is a necessary sub-service to keep stations with an acceptable population of bikes. To aid this I propose a machine learning approach to the problem which should be able to predict the population flow of stations using historical data of the stations' population, weather data, among other sources. These predictions should then be used to create a model that will describe rebalancing jobs that need to be completed to keep all stations in the bike sharing system at a desirable population level.

Chapter 1: Project description

1.1 Introduction

Many modern services today are moving more towards a pay-as-you-go, sharing economy business model. This is primarily driven by the rise of robust computing systems that can facilitate these services and allow shared resources to be allocated and paid for with ease. Bike-sharing services however are relatively low tech, they do not follow a structured schedule like public transport (such as trains and busses), they are not available on request (as in the case of ride-sharing services, such as Uber and Lyft). This is because bikes are essentially allowed to roam free inside the BSS as users take them where they need to go. However due to the fact that most people commute the same way at the same time[2], stations adjacent to the central business district will often become overpopulated in the mornings as people go to work. This poses a problem, as most standard bike sharing services only have a limited number of racks to hold bikes, and a full station means that users will have to find a nearby station with empty space to deposit their bike. This is inconvenient, costs time and energy, making people less likely to use the BSS in future. Similarly, stations in the suburbs (not CBD-adjacent) will become underpopulated or empty, resulting in users who want to use the system being turned away as they cannot get a bike to use.

Because of this, BSS companies must manually redistribute bikes to ensure that their stations are not over or under-populated. However, you cannot just address stations that are already at capacity, as the time it takes to address the problem has already created issues for many customers. This opens up room for a smart approach to the problem.

In this project, I will attempt to create a machine learning approach to the problem, that uses BSS historical data, supplemented with weather data to predict the status of a BSS in the future. Using this knowledge, BSS companies will then be better able to address the issues of over and under-population before they become an issue for their customers. This machine learning algorithm will use BSS data supplemented with weather data, as weather has been shown to have a noticeable effect on the usage of bike-sharing services.

1.1.1 Mandatory Goals

1. Gather data
2. Clean and explore the data sets
3. Use a machine learning approach to predict the population of stations at a given time
4. Use the predictions from the previous step to calculate what the optimal number of bikes to be relocated to/from that station is.
5. Present this information to the user in station pairs, as a "job" for a driver

1.1.2 Advanced Goals

1. Build a user friendly UI front end or app to present the data

2. Write a research paper to present my findings

1.2 Project specifications

Language: Python
Technologies: LSTM Machine learning
Repository: https://csgitlab.ucd.ie/tigoeUCD/fyp-bss-rebalancing

Chapter 2: Related Work and Ideas

2.1 Bike sharing service behavior

Bike redistribution is a necessary subservice of any bike sharing service (BSS). As a public transport service, it is subject to the regular traffic flow of the city, meaning that the majority of traffic flows into the central business district (CBD) in the mornings, and out of the CBD in the evenings (more commonly known as rush hours). For example, Dublin's population fluctuates by roughly 26% during the day as people commute to work (as of 2016) [2]. This creates problems for stations in CBDs or in peripheral areas. If large volumes of traffic are flowing into a given station, the station will quickly become congested, filling its limited storage capacity. This creates an unfortunate situation whereby a user of the service attempts to return their bike to a station only to find that they cannot, and need to find the next nearest station to their destination causing frustration and lost time. (This can be somewhat mitigated by methods such as Dublin bikes grace period whereby users can claim a no-cost time extension on their bike by visiting the overpopulated station [3]). Similarly, a high volume of bikes flowing out of a station can leave it empty, and results in potential customers being refused and resorting to traditional transport methods, defeating the purpose of the BSS. Because of these problems, a bicycle redistribution sub service run by the BSS is necessary.

Most BSSs have the infrastructure to query a station to find how many bikes are available at a station in real-time. Because of this, an intuitive solution to this problem would be to simply send relocation vehicles to stations that are either devoid of bikes, or are full to capacity. However the problem with this approach is the time it takes to relocate the bikes. By the time the problem stations are addressed, many customers have already been refused from those stations. From this one may suggest to address stations that are nearing a problem state, and while this is an acceptable, intuitive solution, stations which are mostly full, or mostly empty are not nearly as problematic as full/empty stations as we know that nobody is attempting to use that station. Because of this stations may also spend a lot of time near capacity without being problematic for BSS customers.

This project aims to address this issue by predicting the population of the stations ahead of time by using a machine learning algorithm, allowing us to address these population problems before they occur. Training will be done on historical records of station population data. Other sources may also be used to train the algorithm if a reasonable argument that they affect the demand for bikes can be made (such as current weather conditions).

However the next problem is to match underpopulated stations with correspondingly overpopulated stations, attempting to manually keep track of the predictions of all stations would be a logistical nightmare, therefore this project will aim to calculate the optimal number of bikes to relocate from said stations, and pair them appropriately. This will essentially generate a list of jobs to be assigned to the relocation vehicles appropriately.

There are other benefits to this system, such as the increased efficiency and reduced environmental impact of the relocation subservice. For example, bike sharing has saved an estimated 22,000 tonnes of CO2 in the US between 2010 and 2014 [4]. However the biggest detractor from this, is the environmental cost of relocating bikes by use of a CO2 producing vehicle. Ensuring that the relocation subservice is efficient and making fewer trips by maximising the efficiency of trips that

are made is important to minimizing one of the few negative impacts on the environment of the BSS.

2.2 Related work

There are many papers currently out there exploring possible methods for predicting the flow of traffic in a BSS, with many taking different approaches.

"A relocation strategy for Munich's bike sharing system"[5] is an analysis of a specific bike sharing system in Munich from March to December 2014, with a proposed user incentivised relocation method. This relocation method would incentivize users to help relocate bikes away from overpopulated systems by dynamically changing the price. The paper also goes into detail about factors that can influence the daily rentals of bikes in the BSS.

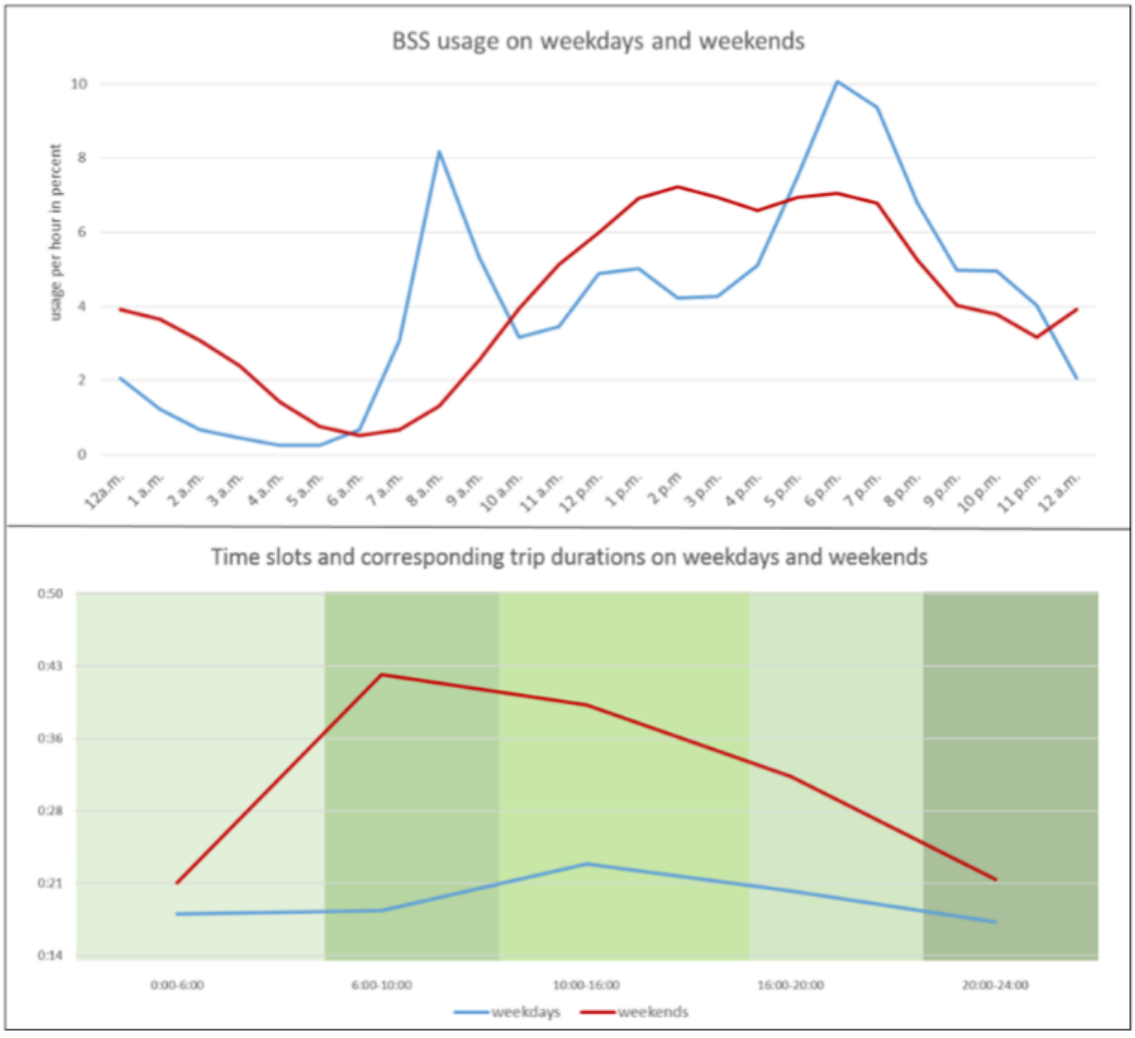

Figure 2.1: Daily demand trends for bikes over time. Reproduced from [5]

Firstly, the paper goes into detail about the analysis of trends in the Munich BSS. Both weather conditions and events such as football matches can greatly impact the day-to-day the demand of the system. However outside of this, clear patterns can be seen by graphing the demand over time on weekends, versus weekdays. On weekdays, the system's demand has very clear peaks during rush hours (7-9am, and 5-7pm), with a smaller peak at lunchtime (11-1). Weekend trends are

entirely different, with more smooth trends all around, plateauing around midday and staying there until roughly 8pm, most likely matching the greater presence of people using the BSS for pleasure, rather than commuting. (See fig 2.1)

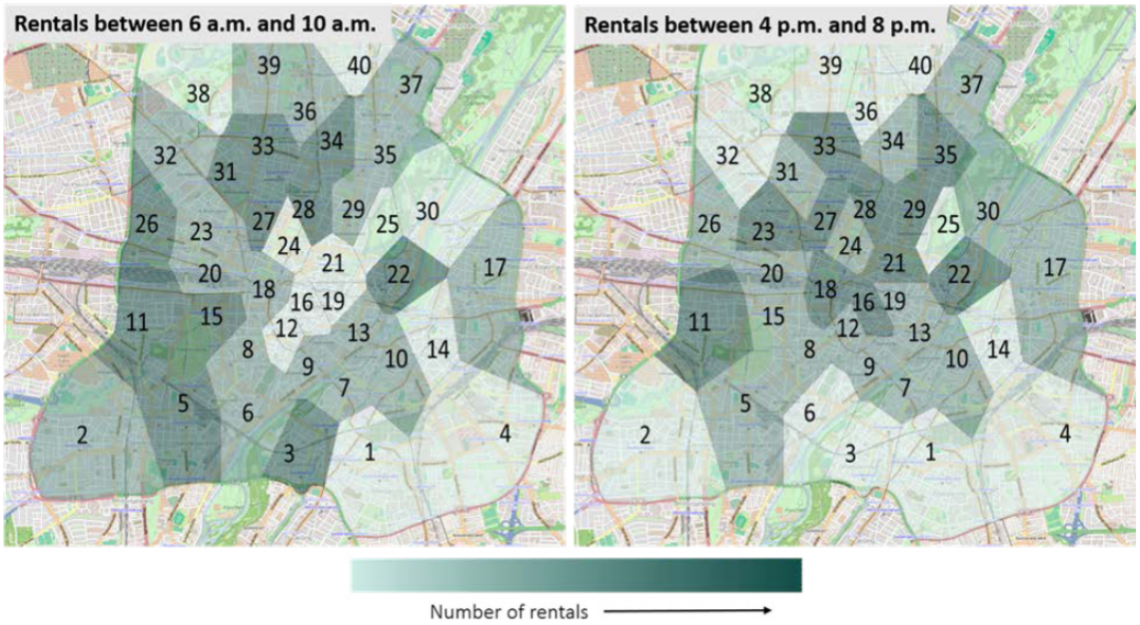

Figure 2.2: Demand in different city areas in the morning and evening. Reproduced from [5]

The report also defines distinct zones and their respective rental rates, showing the demand on stations in those zones in the morning and the evening. In the morning orbital/suburban areas have higher demand, and in the evenings, central/CBD stations do. This further reinforces my point earlier about majority traffic flowing in one direction. (See fig 2.2)

The paper then goes into its methods for managing the bike fleet. It does this by attempting to predict the state of the BSS at a given time from historical bike rental data, and creates new measures to quantify the quality of the station at a given time such as the "demand factor", which is a ratio of a station's rentals to its stock at a given time. It uses these measures to quantify the quality of the BSS in future, and ultimately predict where bikes will be, and where they will be needed. This information is used to rebalance the BSS, and even generate a route for the BSS relocation service to take.

However this paper also explores a very interesting alternative solution to the standard manual BSS rebalancing. User based rebalancing is an idea to take the state of the BSS and dynamically adjust the cost for users renting from a given station. The paper then goes on to explore the advantages and disadvantages of such a system, suggesting that it be used in conjunction with a standard driver redistribution system for highly imbalance systems (>15%). The system most notably requires no manual intervention, allowing the BSS to save time and fuel on relocation efforts. The system works best however when the destination of the user is known as well, as in some situations, rentals from a discounted station could be deposited in less desirable locations, resulting in an even more unbalanced system.

"A dynamic pricing scheme with negative prices in dockless bike sharing systems"[6] is a similar paper with an extreme example of this idea where a system could actually have "negative" prices (i.e. pays the user to take trips that are beneficial to the equilibrium of the system).

"Predicting bike sharing demand using recurrent neural networks"[7] uses deep LSTM to predict the demand of bicycle sharing based on historical BSS, and weather data. The paper starts with an analysis of it's New York based dataset, similar to the previous paper, and finds many similar trends appear, periods of high demand, morning and night in orbital areas and CBD respectively.

It also uses local weather data to further enhance the predictions made by the machine learning algorithm (temp, precipitation and humidity).

This information is then used in a LSTM neural network to construct a model to predict the arrival and departure rates of bikes at a given station, and hence can be used to predict the net demand.

This is the paper I have taken the most inspiration from when deciding on my approach for predicting the population of stations. It proves that LSTM can be a good algorithm for time-series data, and can take input from multiple data sources (in this case, weather and BSS historical). The paper finds that the model works decently well when evaluated on a test set, however the model can be skewed by areas with frequent events such as the Museum of Modern Art. Overall it is a good demonstration of how LSTM can be applied to this problem domain.

"A Destination Prediction Network Based on Spatiotemporal Data for Bike-Sharing"[8] is another paper about predicting BSS data, but rather than predicting the population at a given time, it is more focused on predicting individual journeys, and their respective destinations.

This paper also goes into some detail about the data cleaning process before the project, which is useful for me to have a second opinion about the data cleaning process when I attempt this for my own project. For example, they talk about removing entries with latitude or longitude far outside the area of operation for the BSS (Beijing in this case).

This project uses FP analysis to extract useful information from the dataset, and builds four different item subsets to extract useful information (user-origin-destination, user-origin, user-destination, and origin-destination). Each set can potentially identify useful trends, for example origin-destination can identify frequent routes traveled by people in the system. These subsets are then used to build the candidate set, and ultimately predict a user's destination.

All of this information is then used to create the "destination prediction network", which is the main algorithm used for this paper. It is based on a combination of a LSTM algorithm to predict the destination based on historical user data, a CNN used to learn the spatial relationships between stations in the BSS and a FCNN to learn how external features (such as meteorological data) which will affect demand on the BSS.

Putting this algorithm into practice, when a bike is rented from a station, it's destination prediction network (DPN) is used to attempt to predict that user's destination. During evaluation, the paper finds decent success with this methodology after testing it on real world data from Mobike in Beijing

This is an interesting alternative approach for the problem, however I feel that the added complexity of predicting where a particular user is likely to end up is unnecessary for the purposes of this project when the main goal is to rebalance bikes. A large part of this project is also based on user identifying information, which I do not have access to, and creates a host of privacy issues, were I to recreate this more closely.

"Efficient Spatial-Temporal Rebalancing of Shareable Bikes"[9] is another paper I have taken inspiration from, as it goes into great detail about a possible rebalancing algorithm, calculating the recommended number of bikes to relocated to/from a given station provided that the population of the station can be predicted ahead of time. This paper goes into detail about a method for calculating the optimal number of bikes to add to or remove from a station given that the population of the stations in the BSS can be predicted ahead of time. The paper goes into further detail about a proof for the stability of this system by designing a Lyapunov function, and contains a pseudo code example of a method for calculating this rebalancing. This paper was a big inspiration for this project, as it provides a clear method for maintaining a stable equilibrium in a BSS and aligns very closely with my project.

"Exploring the weather impact on bike sharing usage through a clustering analysis"[10] is an exploration of the effect weather has on the usage of Washington DC's bike-sharing system. It first creates a correlation matrix for the different features in the weather dataset, as certain features will be more highly correlated than others. For example, precipitation and relative humidity are highly correlated. It then goes into an exploration of the specific weather features and their impacts on BSS usage. Overall temperature, followed by precipitation were the most important. However, because of the high number of entries in the Dublin weather dataset with 0 precipitation, and the high correlation between precipitation and humidity, I elected to include both in this project. This would allow the algorithm to get a more accurate representation of the precipitation in Dublin at a given time.

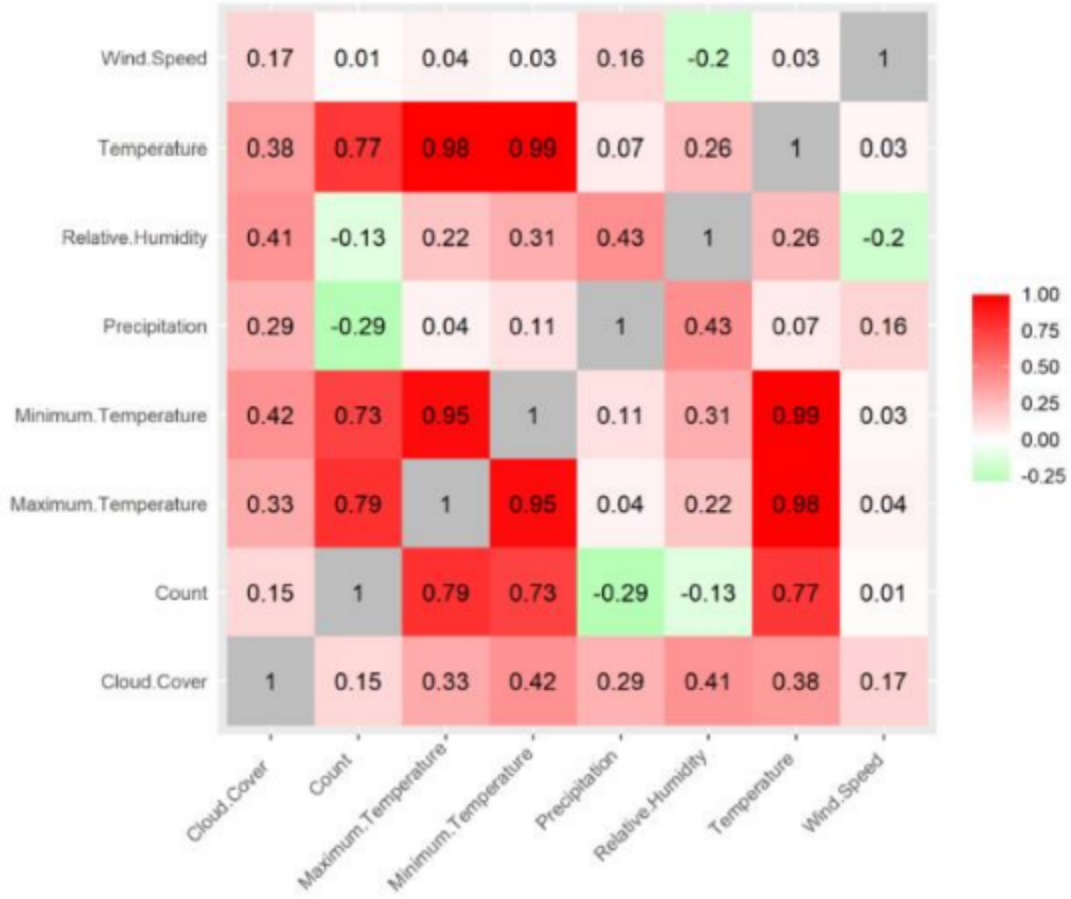

Figure 2.3: Correlation between different weather features. Reproduced from [10]

Overall this paper was important for proving the effect weather has on BSS usage, as well as which features were most important to predicting the future states of BSS.

2.3 Summary

These are some of the papers I have found on the subject about dynamic BSS bike relocation. However, most of these papers all approach the topic from a purely theoretical standpoint, any implementation done by these authors has mostly been for the purposes of verification and measuring accuracy of the methodology. During my research, I have not found any publicly available software to dynamically rebalance BSS systems, which leaves a niche for this project to fill.

The easy availability of large quantities of data that are relevant for this project is also ideal, and should make the data collection aspect relatively easy to complete.

Chapter 3: Data Considerations

3.1 Dataset Sources

For my project, I primarily focus on data from Dublin city, as that is the city I am most familiar with, however the concept should be universal, and be able to be applied to other cities without much difficulty.

The primary focus of my data collection is on the bike services themselves, and their historical records of station population over time, as this will be the best, most direct way of predicting how full those stations will be in the future. This data is freely available from smartdublin [11] both historical and real-time. It provides APIs to request real-time and historical data, as well as CSVs containing quarterly historical data going back to Q3 2018. This dataset is ideal, and provides plenty of training data for my model, as well as a way of requesting real time data for testing.

Upon a cursory analysis of the data, it seems very straightforward. The table provides a total number of stands, as well as the number of bikes and spaces available for each of its five minute intervals. The data is clear, numerous, and (comparatively) extremely granular. This, alongside the total absence of personally identifying data makes this practically a perfect dataset for the purposes of this project. Because I have already downloaded the data, the majority of the work for this step will be to conglomerate it into a single schema.

My secondary focus was weather data, which proved more difficult to obtain than the station data. The data is thankfully available at met eireann's website [12]. However the data will require more effort to clean than the station data for several reasons. The data is available on an hourly basis which is fairly granular but not to the same extent as the BSS data. However the dataset also provides a great deal of unnecessary information such as soil temperature at varying depths which must be removed before use. Finally this data has a more in-depth licensing agreement, which I must take care to observe during my project, but ultimately this should prove a very robust dataset and can hopefully add to the accuracy of the algorithm's predictions.

3.2 Dublin bikes dataset

This is the heart of the project, and it is important to get as much data as possible. Fortunately, approximately 2-3 years worth of data is available at smartdublin.ie[11]. While this is nowhere close to the full historical dataset of DublinBikes, as the system has been operating in Dublin since 2009[13], it is a very easy to access resource that gives plenty of data to work with. The historical data is available through an API or as static CSVs hosted on the smartdublin website. I use wget to download the static files, during the preprocessing stage of the project.

This data contains no identifying information and is available for use under the Creative Commons Attribution license.

The data in question is presented as CSV files, with each file representing one fiscal quarter's worth of data, starting in Q3 2018 up to Q4 2020. The data is presented in 5-minute intervals,

giving an excellent, extremely granular view of the BSS. However, the data needs to undergo some cleaning, as it has some cases of missing values, duplicate values, and many redundant rows. For example, each row lists the station id, station name, and geographical coordinates despite none of those changing from row to row. By the end of the cleaning phase, the data will be converted into CSV files, each representing a station in the BSS, and the files will be reduced to the date and time, as well as an integer describing the number of bikes in that station at that time (removing all redundant rows).

This must be done for the purposes of training the machine learning algorithm. Each of these files allows the system to be trained on one station at a time, as each station has unique trends that need to be learned, independently of one another. This is clearly demonstrated by the figures below, which graph the average population of two stations during specific years (fig 3.1 / 3.2).

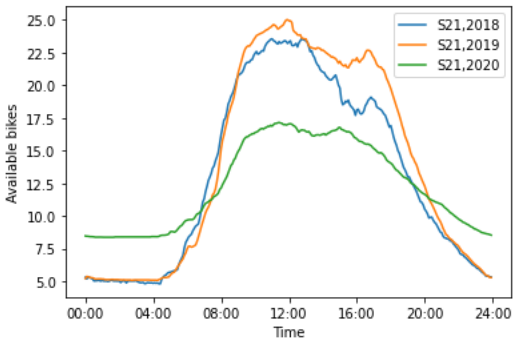

Figure 3.1: Leinster Street South

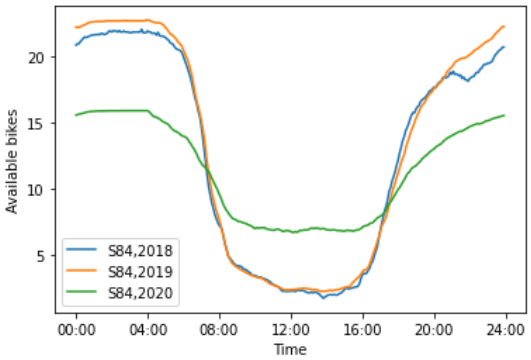

Figure 3.2: Brookfield Road

3.3 Weather data

Weather has been shown to have an impact on the usage statistics of bike-sharing systems[10]. Because of this, I aim to improve the accuracy of the predictions of the ML algorithm by supplementing the BSS data, with corresponding weather data. To do this, I need historical weather data for Dublin to match up with my BSS data.

Met Eireann provides historical weather data for all of its weather stations. This also contains no personal information and is available under the creative commons attribution license.

The weather data used in this project is from the Phoenix Park weather station and is available on an hourly basis. This is less than ideal, Phoenix park is relatively far from the city center when compared to most of the stations in Dublin bikes' area of operation. While Met Eireann has records that are more central (Merrion Square) and more representative of the weather at BSS stations, this weather is only available on a daily basis. The large magnitude of data loss is not a worthwhile tradeoff which is why the Phoenix park data is used.

An alternative to Met Eireann as a source for weather data is any number of weather service APIs, such as visualcrossing.com. However, this presented two main problems. Firstly, while it is possible to request weather data for Dublin, it does not specify where in Dublin, meaning that the data could be coming from a source even further from the city than the Phoenix park weather station. Secondly, these options are usually priced, or place a cap on the number of requests users can make. With the number of requests that would need to be made, this would put a large bottleneck on progress. For these reasons, the project uses the static Met Eireann data instead.

This data is available as a static CSV from the Met Eireann website, which is also retrieved using wget during the preprocessing phase. This data contains hourly weather data going back as far as 2003. This is a lot of unnecessary data for the purposes of this project, so only the weather data that correspond to dates in the BSS data are kept, the rest is discarded. Similarly, the data contains fields that are not relevant to this project (eg "msl" which corresponds to Mean Sea Level Pressure). It is also missing some data, so this must be addressed.

3.4 Summary

Overall, the data for this project is extremely convenient. The data is available for my own locality (Dublin) meaning I can be personally familiar with the geography of the BSS and can have a more innate understanding of the system. The BSS data in question is readily available, in a convenient format and is extremely robust. The dataset is large (>2 years) and is extremely granular in its description of the BSS (in 5-minute intervals).

The weather data is not quite as robust, but still acceptable for the purposes of this project. The data is only available in 1-hour intervals instead of 5-minute, meaning that some assumptions must be made as we match BSS data entries to their nearest weather counterparts. All weather data is also coming from a single weather station (Phoenix Park) rather than bespoke stations close to their BSS station counterparts, which could be problematic with Ireland's notoriously changeable weather. However it should provide an adequate approximation of weather which can help inform the LSTM's predictions.

These factors combined with the facts that neither dataset contains any identifying information, and available to freely use under the creative commons license means that Dublin is an ideal candidate for this project.

Chapter 4: Outline of Approach

The project primarily uses python 3.8, using Tensorflow as the primary machine learning library. The project is broken down into roughly 7 parts. Six of which have a corresponding Jupyter-notebook which is used to run that part of the project. The front-end is run through a separate script using Flask. The goal is to predict the state of Dublin Bikes BSS ahead of time, as well as explore and understand the features and parameters which optimize this process.

This chapter describes how the implementation works to achieve these goals.

4.1 Preprocessing

This has been partially covered in data considerations, so this will somewhat be a summary.

The data is retrieved from their respective websites using wget. Met Eireann for Dublin's weather data, and smart Dublin for Dublin bike's historical data. Both are cleaned, (removal of missing data, unnecessary attributes (such as station name), removal of duplicate rows, etc).

The date-time field is very important for recognizing trends within the BSS. However, the format needs to be converted into 3 separate attributes to be used by the machine learning algorithm. Firstly the field "int_time" is an integer representation of the time of day for a given data point. Because the data is supplied in 5-minute intervals, there can only ever be a maximum of 288 (24hrs * 12 5-min intervals) data points per day. This will then be able to capture daily trends within the system. Secondly, the date is converted to "int_date" which is an integer representation of the date of the year (for example 10th of February becomes 41) (ranges from 0-365). The reasons for this are twofold. It can capture seasonal trends, and it should also be able to capture public holidays and other days which may result in lower commuter numbers Finally, "int_day" is an int representation of the day of the week (0-6). This should be able to catch the difference between weekdays and weekends, as weekdays have higher demand on the BSS.

All of these fields are extrapolated from the date, and added to the dataset as a new column. This new dataset is then converted to a CSV file and saved during the reorganization process, where data is split up into station specific CSVs.

The final step taken in pre-processing is attaching the weather data to the BSS data for ease of use. However, because of the different levels of granularity in each of the data-sets, some approximations have been made. While one solution could have been to simply remove the BSS data points that did not have a corresponding weather entry, this would have dropped around 95% of the BSS data. Because of this, I decided to approximate the weather, by matching each BSS entry, to its closest corresponding weather entry. This unfortunately will lower the accuracy of the weather to its real value at the time, however, the difference is at most 30-minutes, which is acceptable given the less-than-ideal data conditions.

This is done and then attached as a final CSV, which is saved under "/reorg_w_weather". All of these steps are implemented, and can be run in the "preprocessing.ipynb" Jupyter notebook. These files are now considered fully pre-processed and can be used in the future sections of this project for the purposes of analysis and training the LSTM algorithm.

4.2 Pre-analysis

This section primarily deals with the investigation and analysis of the data before applying the machine learning algorithm. While many of the trends that appear in BSSs have been explored by papers in the "related work" section it is important to explore some of these in relation to Dublin specifically. The trends may not apply to Dublin in the same way, or to the same degree. It is also important to examine the impact that the year 2020 had on usage stats for Dublin Bikes.

4.2.1 Daily Trends

Firstly I simply graph the average daily trend of several BSS stations against one another. This demonstrates the differences in trends between the different stations. Generally speaking, the station population remains mostly static in the mornings between the hours of 00:00 and 04:00, then varies wildly during the day as people use the stations. It then becomes more stationary in the evenings after 18:00. This also demonstrates the point made by this, that certain stations will have similar usage stats (eg CBD-adjacent vs Suburb adjacent).

This is done by generating a python dictionary, where the keys are each of the 288 possible "int_times" and the values are the total sum of populations at that time. These values are then divided by the total count of days looked at, to get the average pop at that time, which is used to plot the graph below 4.1

```
dataset = pandas.read_csv(filename, usecols=['TIME', 'AVAILABLE BIKES', 'int_time'])
dataset['DATE'] = dataset['TIME'].apply(lambda x: extract_date(x))
dataset['YEAR'] = dataset['TIME'].apply(lambda x: extract_year(x))

# Calculating average pop at all times during the day
time_total = {}
time_count = {}

times = dataset['int_time'].unique()
times.sort()

# fill dict
for time in times:
    time_total[time] = 0
    time_count[time] = 0

if year != 0:
    dataset = dataset.loc[dataset['YEAR'] == str(year)]
    if len(dataset) == 0:
        print("No data exists for " + str(year))
        return

for i, row in dataset.iterrows():
    time_total[row['int_time']] = time_total[row['int_time']] + row['AVAILABLE BIKES']
    time_count[row['int_time']] = time_count[row['int_time']] + 1

averages = [x / y for x, y in zip(time_total.values(), time_count.values())]
```

```
label = 'S' + str(station)
if year != 0:
    label = label + ',' + str(year)

plt.plot(times, averages, label=label)
```

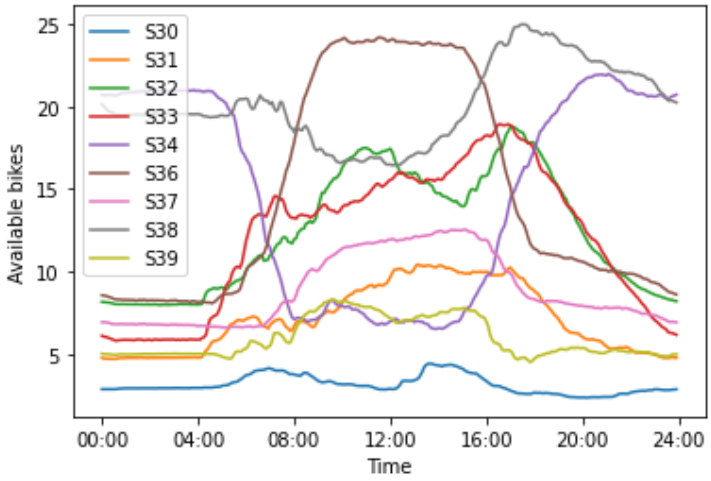

Figure 4.1: The population of several stations in the Dublin Bikes BSS over the course of an average day

4.2.2 2020 individual impact

Next, I explore the differences between 2020, and 2019/2018. As with many other aspects of life, the covid-19 pandemic has had a large impact on the usage of BSS. For the purposes of this project, It is important to understand how this has impacted Dublin Bikes. Graphing several different stations' average daily trends demonstrates this effect well. These graphs were generated with the same method as above, with the year passed to the method, to specify for the system to use a subset of the data, as well as label the graph accordingly.

Stations such as Leinster street south 4.2 that are very central to Dublin experienced a decrease in usage, with lower peaks, and higher off-times.

Meanwhile, peripheral stations such as Kilmainham Gaol 4.5 actually experienced more traffic than typical, experiencing the opposite effect to central stations with higher peak times and lower off times.

Finally, intermediary stations, which generally see a massive increase in population at all times. The usage patterns however are generally the same. 4.6 / 4.7

Overall 2020 had a significant effect on the usage of Dublin Bikes which needs to be taken into consideration for future parts of the project, which is demonstrated clearly by this section.

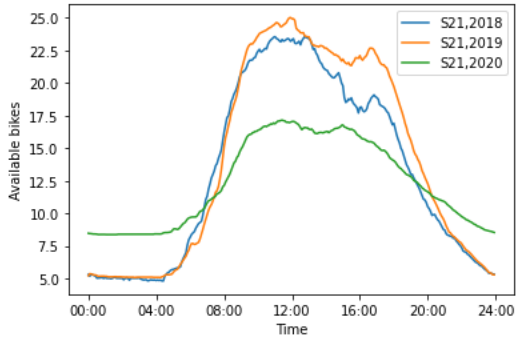

Figure 4.2: Leinster Street South

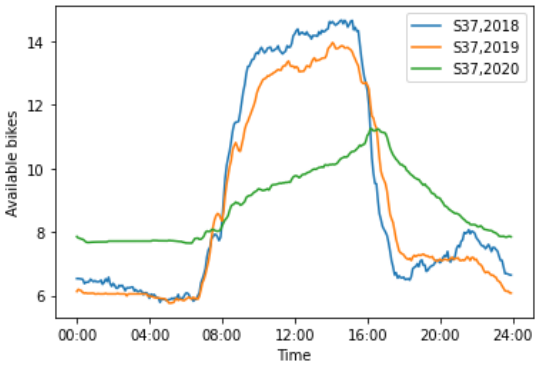

Figure 4.3: St. Stephen's Green South

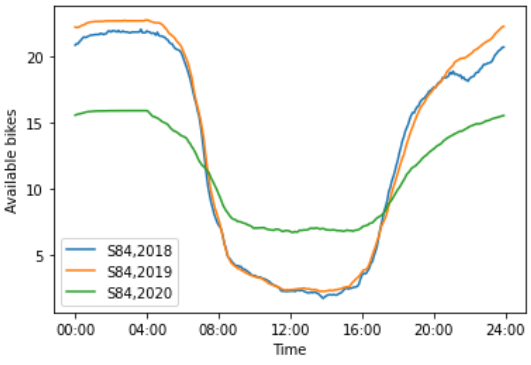

Figure 4.4: Brookfield Road

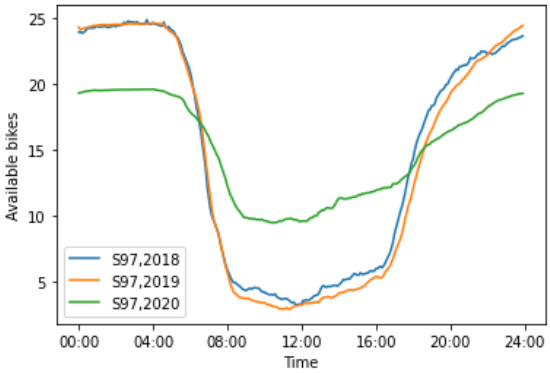

Figure 4.5: Kilmainham Gaol

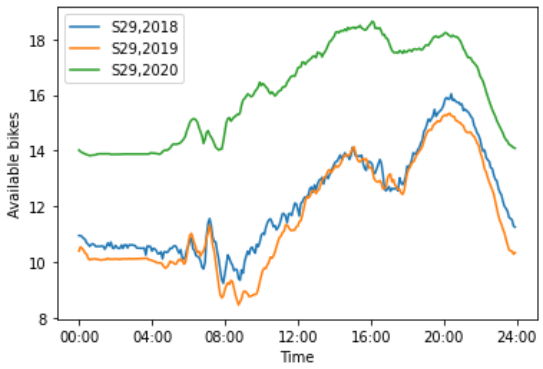

Figure 4.6: Ormond Quay Upper

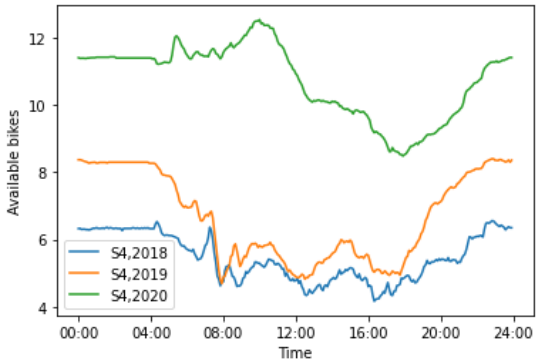

Figure 4.7: Greek Street

4.2.3 2020 General impact

As a summary of these findings, the average difference for each station is calculated and displayed. The average differences range from 2 all the way up to 11. This concisely demonstrates the overall impact of 2020 on the year's BSS usage, and the importance of taking 2020 into account for the purposes of predictions.

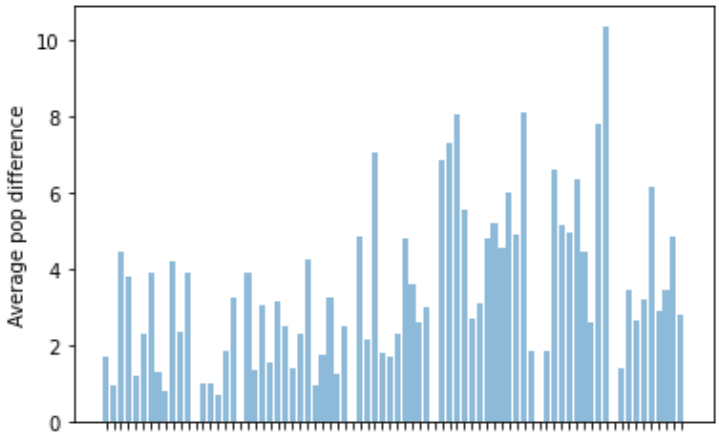

Figure 4.8: The average differences of stations between 2020 and their normal population levels

4.2.4 Weather Impact

While the impact and correlation that weather has on the BSS is explored in a later section, a preliminary analysis is performed here. The impact weather has on the BSS is shown by "Exploring the weather impact on bike sharing usage through a clustering analysis"[10], an attempt was made to show a simplified version of the study on the Dublin BSS. However, the results from this methodology were not as prominent and shows the need for the deep dive Quach and company[10] takes.

This was implemented through a similar system as the average day graphs above, however instead of passing the parameters to the method, a slightly modified method which gets subsets of the dataset where all values for each subset are either above or below a pre-specified threshold. This threshold was selected by trial and error to find the most prominent difference between the two graphs.

```
for col, threshold in columns.items():

    dataset = pandas.read_csv(filename).head(150609)

    if col == 'rain':
        dataset = dataset[dataset[col].str.strip().astype(bool)]

    dataset[col] = dataset[col].astype('float64')
    above_dataset = dataset.loc[dataset[col] >= threshold]

    below_dataset = dataset.loc[dataset[col] < threshold]

    plot_average_day_from_dataset(above_dataset, label=col + " >= " + str(threshold))
    plot_average_day_from_dataset(below_dataset, label=col + " < " + str(threshold))

    show_average_day_plot()
```

For the three weather attributes used, the difference is shown between population over time of a station above and below a certain threshold (fig 4.9/4.10). However, a reason these results may be underwhelming is due to the impact that time of day has on these attributes. For example, temperature will go up in the middle of the day, and go back down at night, meaning that impact could be so great as to eclipse any impact that the daily temperature variation may have.

4.2.5 Summary

All of these results can be seen and modified in the "pre-analysis.ipynb" Jupyter notebook.

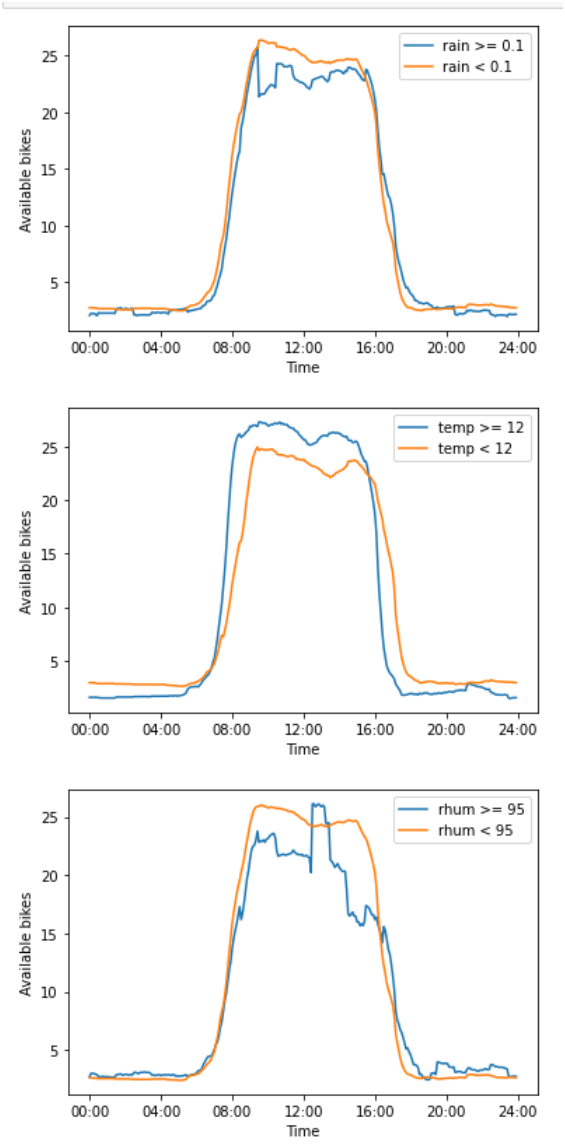

Figure 4.9: Ormond Quay Upper

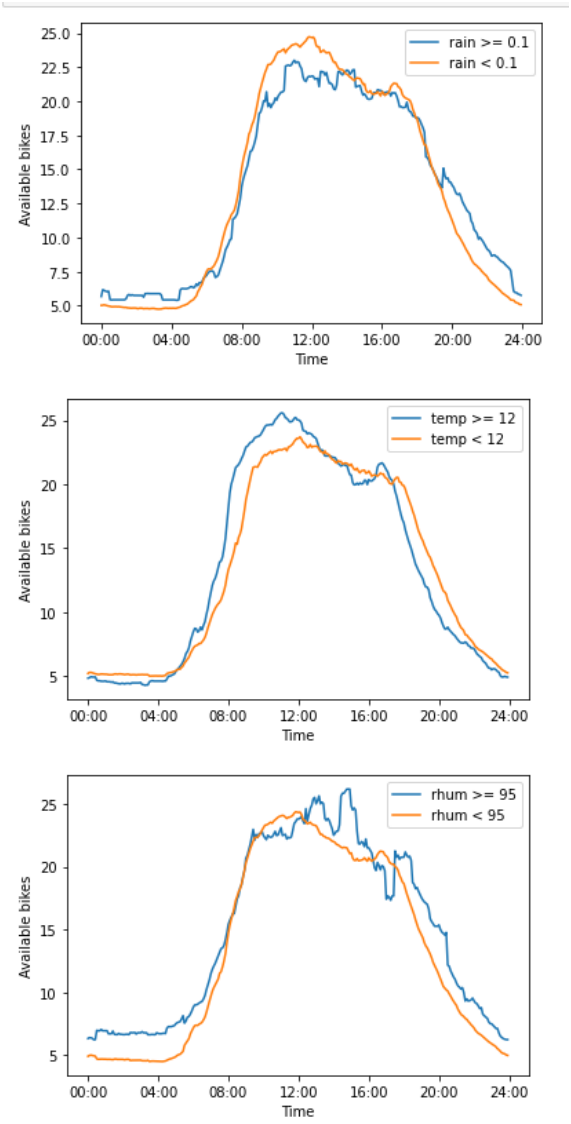

Figure 4.10: Greek Street

4.3 LSTM Training

This section explores the initial setup of the long short term memory (LSTM) algorithm. This will then be used in the next section for feature optimization and model generation.

As explored in the "related work" section, the LSTM algorithm is ideal for time series data, as it has both long and short-term memory. Long-term memory identifies large patterns (such as daily or seasonal trends) and short-term memory helps inform predictions by remembering recent predictions. It has been used in a variety of time series learning applications, and many of the related papers use it as their learning algorithm of choice.

The algorithm was built using the python library Keras, assembled with 40 unit LSTM and 1 unit Dense (for the one-dimensional output). The model also used epochs 150, and batch size 64. Most of these parameters were selected after trial and error testing, in an attempt to find the optimal balance between performance and compute time.

The loss function selected was 'mean_square_error', as this emphasizes the penalty placed on large errors. It is more important for the purposes of this project that predictions be broadly correct.

The difference between stations that are full, and stations that are nearly full is minimal, as they both pose a significant threat to the BSS, and need to be addressed soon. Small inaccuracies in predictions are not nearly as problematic for the purposes of this project hence the use of MSE as the loss function.

This section makes some preliminary predictions from a randomly chosen station and evaluates its prediction using a short custom list of metrics (MAE, MSE RMSE, and R2 score) to evaluate the initial efficacy of the system.

4.4 LSTM Feature optimization

This section is primarily focused on investigating the effects each feature has on the LSTM's metrics. This can then be used to select the feature subset which provides the best results for the project.

This is done by testing the powerset of features (set of all subsets) and comparing the performance of each. This is saved to a CSV file which generates a large spreadsheet of feature subsets and their corresponding evaluation metrics, as seen in fig 4.11.

	A	B	C	D	E	F	G	H	I	J	K
1	int_time	int_date	int_day	rain	temp	rhum	mae	mse	rmse	r2	timestamp
2	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	3.1688075	20.490044	4.52659294	0.3514005192	05/04/2021 15:3
3	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	4.428482	36.173138	6.014410833	-0.1450378827	05/04/2021 15:4
4	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	4.379529	33.96716	5.828135145	-0.07520897483	05/04/2021 15:5
5	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	4.4353113	36.82066	6.068002937	-0.1654553644	05/04/2021 16:0
6	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	4.445962	31.265474	5.591553838	0.01031093537	05/04/2021 16:1
7	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	4.4321456	30.992638	5.567103164	0.01894759119	05/04/2021 16:2
8	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	3.1613526	19.913088	4.462408301	0.3696636457	05/04/2021 16:3
9	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	2.8957913	*17.384443*	*4.169465587*	0.4497064032	05/04/2021 16:4
10	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	4.382298	33.418987	5.780915782	-0.05785708562	05/04/2021 16:5
11	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	3.1782124	20.709522	4.55077161	0.3444978744	05/04/2021 17:0
12	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	4.430717	36.04908	6.004088598	-0.1410331305	05/04/2021 17:1
13	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	4.3790417	33.8579	5.818754052	-0.07167735581	05/04/2021 17:2
14	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	3.1305602	19.869692	4.457543253	0.3710373594	05/04/2021 17:3
15	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	4.292548	30.132395	5.489298206	0.04617792354	05/04/2021 17:4
16	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	4.4453125	30.3854	5.512295418	0.03816915158	05/04/2021 17:5
17	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	4.446735	31.253714	5.590502089	0.01075085811	05/04/2021 18:0
18	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	3.1276174	19.825712	4.452607349	0.3724295051	05/04/2021 18:1
19	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	4.201975	31.616081	5.622817909	-0.00078718111	05/04/2021 18:2
20	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	4.1774526	28.662079	5.353697681	0.09271986061	05/04/2021 18:3
21	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	4.3713846	30.251118	5.500101609	0.04248518774	05/04/2021 18:4
22	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	4.507245	31.213741	5.586925926	0.01194857351	05/04/2021 18:5
23	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	3.0139894	18.411686	4.290884052	0.4171896751	05/04/2021 19:0
24	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	3.1796227	20.129328	4.48657194	0.3628622135	05/04/2021 22:4
25	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE	2.8979354	17.385105	4.169544955	*0.4497229581*	05/04/2021 22:5
26	FALSE	TRUE	TRUE	TRUE	FALSE	FALSE	4.3813434	33.368153	5.776517343	-0.05617585444	05/04/2021 23:0
27	TRUE	TRUE	FALSE	FALSE	TRUE	FALSE	3.2474313	20.792631	4.559893765	0.3418222892	05/04/2021 23:1
28	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE	3.0233064	19.27507	4.390338277	0.3898597412	05/04/2021 23:2
29	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	4.211096	29.227419	5.406238887	0.07482441855	05/04/2021 23:3
30	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE	3.1579463	20.221466	4.496828445	0.3599458383	05/04/2021 23:4
31	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	4.283425	30.25616	5.500560038	0.04232561427	05/04/2021 23:5
32	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	4.437955	30.321127	5.506462289	0.04026928277	06/04/2021 14:0
33	TRUE	TRUE	FALSE	FALSE	FALSE	TRUE	3.1380932	19.911184	4.46219501	0.3697239414	06/04/2021 14:1
34	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	2.922057	17.992151	4.241715603	0.430469685	06/04/2021 14:2
35	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	4.0767694	30.782368	5.548185983	0.02560335963	06/04/2021 14:3
36	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	3.125035	19.780035	4.447475129	0.3739181576	06/04/2021 14:4

Figure 4.11: A sample of the generated feature subset report

```
def powerset(s):
    x = len(s)
    masks = [1 << i for i in range(x)]
    for i in range(1 << x):
        yield [ss for mask, ss in zip(masks, s) if i & mask]


def test_powersets(start_position=0,
                   file='../datasets/bss/dublin/reorg_plus_weather/station_2.csv',
                   train_start_date=None,
                   train_end_date=None,
                   test_start_date=None,
                   test_end_date=None,
                   ):
    attr_list = [
        'int_time',
        'int_date',
        'int_day',
        'rain',
        'temp',
        'rhum'
    ]

    y = list(powerset(attr_list))
    print(len(y))
    y = sorted(y, key=len)
    y.pop(0)

    for x in y[start_position:]:
        print(str(start_position) + "/" + str(len(y) - 1))
        start_position = start_position + 1
        print(x)
        warnings.filterwarnings("ignore")
        if train_start_date is not None and train_end_date is not None and test_start_date
            train_model(file,
                        train_start_date=train_start_date,
                        train_end_date=train_end_date,
                        test_start_date=test_start_date,
                        test_end_date=test_end_date,
                        cols_to_use=x,
                        verbose=0)
        else:
            train_model(file,
                        cols_to_use=x,
                        verbose=0)
        print()
        keras.backend.clear_session()
```

This code generates the feature report. After generating the powerset of all the features in the dataset, it then loops through each subset in the powerset. For each one it generates a model with the parameters explored in the previous section, makes sample predictions, and then outputs the metrics to the output CSV.

While this information highlights some particularly effective combinations, this needs to be converted into a more digestible format. To do this, for each feature, the average metrics for the subset that used that metric, and that did not use that metric are calculated and the difference between them is found. This quantifies the average impact the feature has on the model (fig 4.12).

```
writer = csv.DictWriter(csvfile, fieldnames=col_names)

attr_list = ['int_time','int_date','int_day','rain','temp','rhum']

stats_list = ['mae','mse','rmse','r2']

for attr in attr_list:
    stats_with_attr = stats[stats[attr] == True]
    stats_without_attr = stats[stats[attr] == False]

    line_dict = {}
    line_dict['feature'] = attr
    for stat in stats_list:
        average_with = stats_with_attr[stat].sum() / len(stats_with_attr)
        average_without = stats_without_attr[stat].sum() / len(stats_without_attr)

        line_dict[stat] = (average_with - average_without)


    writer.writerow(line_dict)
```

feature	mae	mse	rmse	r2
int_time	-1.185461493	-11.60374888	-1.15054799	0.3672963564
int_date	-0.00147442629	0.4943632531	0.05607754855	-0.01564980592
int_day	-0.1163489837	-1.478672632	-0.1476734776	0.04680319055
rain	0.02460539537	0.3560973476	0.03599404505	-0.01121593896
temp	0.04087739295	-0.8833119348	-0.05978305437	0.02796044892
rhum	-0.09526790518	-1.671893679	-0.1487577354	0.05292135355

Figure 4.12: The differences (deltas) between models with and without each feature for a specific station

As can be seen from metric deltas, int_time is by far the most effective feature in the feature space with the greatest deltas across the board with an r2 score of 0.37.

"Int_date" was not as effective as hoped with marginal improvement to the MAE and r2 score but an increase in the RMSE. This indicates that it has a negative impact on outlier predictions. While the initial intention of the int_date was to catch seasonal trends, it seems that the weather information does an adequate job of addressing this (As weather changes with seasons). Its other function was to catch days such as public holidays where commuter numbers will be lower. However, due to the relatively small number of public holidays in the year, this also does not have a large impact on the dataset.

"Int_day" also provides minor improvement. Unlike public holidays which only occur about 3% of the time, noticing the difference between weekends and weekdays (a much larger 2/5 split) is more important for the performance of the algorithm.

Rain was the only feature to actually unequivocally **reduce** the efficacy of the algorithm. This is surprising considering the paper by Quach[10], which listed precipitation as the biggest contributing factor. However the earlier mentioned issue with a large number of the entries containing 0 for rain could be the reason for this discrepancy.

Temperature provides improvements to the RMSE and harms the MAE and r2 scores, giving the opposite effect to the "int_date" feature. It is able to improve predictions that are closer to the real answer, however fails when it comes to predictions that are further off.

Finally, relative humidity provides improvement to all metrics, indicating that it is the most relevant feature to the task at hand. It provides the best metrics out of all the weather stats, and 2nd best overall.

4.5 LSTM model generation

This section focuses on the generation and saving of the many LSTM models that are used in this project.

Because of the time, it takes even for a decent computer to generate a fully trained LSTM model with the correct parameters, it is necessary to save the models. These models can then be loaded back into Python as desired to query them again. By using the inbuilt saving function in Keras sequential model, these models can be saved as .h5 files. As a side effect of this, the min-max scalar objects must also be saved to successfully scale and de-scale the values we attempt to predict (this is done through python library pickle).

In this section, two types of models are generated. "Full" models which use the full range of selected features and "simple" models which only use features that are generated from the date/time ("int_time", "int_date", "int_day"). The simple models can then be used for easier querying and testing. As mentioned above, there was difficulty in getting a weather API for this project considering the large volume of queries that would be required. This simplifies that process somewhat and allows for easier testing.

4.6 Prediction report and Job generation

This section focuses on the generation of a report to predict an entire day in the Dublin bikes BSS. These reports are then used to generate rebalancing jobs for drivers to use to optimize the BSS.

As a way of helping BSS operators manage bike fleets better, we reach the crux of the project. Predicting an entire day of BSS data at once. This takes in a given date and generates the station population for each station at one-hour intervals. While more granular could be generated, this strikes a good balance between granularity and overwhelming amounts of data.

These reports are then used to generate prediction jobs for BSS drivers. This is done by examining the state of the BSS at a given hour, and calculating the relative capacity of each station (a percentage representing how full each station is). Lists of over, and underpopulated stations are then assembled by taking a subset of stations that are outside a predetermined "safe" capacity (between 0.2-0.8). These lists are then sorted by relative capacity so the most problematic stations (stations closest to 0 or 1) are at the top of their respective lists. The system then generates jobs by

pairing the most problematic stations and adding that to a list of jobs. This job is then "simulated" in the system by updating the populations of the stations involved, and recalculating the relative populations of the stations. The lists are then sorted again, and this process is repeated until either list runs out of stations outside the predetermined safe population threshold.

These jobs are then generated as tuples in the form (ID of station to move from, ID of station to move to, Number of bikes to move). While these are not saved, however they can be viewed in a more user-friendly way through the flask front end.

```
[
(44, 11, 11),
(45, 40, 9),
(97, 30, 9),
(91, 31, 8),
(94, 41, 9),
(115, 54, 7),
(85, 62, 7)
]
```

Figure 4.13: A job generation example

4.7 Frontend

Because of the potential commercial and public use for this project, a front end was developed to more easily allow users to avail of this service. This front end also makes it easier for drivers to interact with the service and request jobs.

As the world moves more towards a pay-as-you-go, sharing economy bike-sharing systems such as Dublin bikes may become even more popular. However, because they do not operate on a fixed schedule and are not available on-demand BSSs are relatively low tech and are hard for users to plan their day around. This project could potentially be very useful to release to the public in the future.

Because of this, a UI interface was developed with HTML in the python library Flask. This allows users to interact with the LSTM models by inputting the data manually. This can be done through one of three methods.

Firstly, singular predictions (fig 4.14). Users can select a station from a drop down of all stations, and input the date and time with a bootstrap form. The date time is then converted into the three date-time attributes mentioned earlier for the purposes of prediction (int_time, int_date, and int_day). Users can also choose between "simple" and "full" models by toggling a checkbox. Simple models will only use the station, and the date-time inputs. However with full model enabled, users can also enter weather stats to improve the predictions.

Users can also generate a list of predictions (fig 4.15). Similar to the first interface users are required to enter a station and date-time. However due to the aforementioned problems with weather API, this method always uses the "simple" version of the LSTM model.

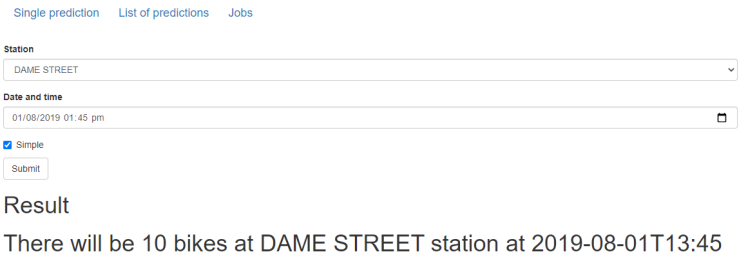

Figure 4.14: The flask interface for singular predictions

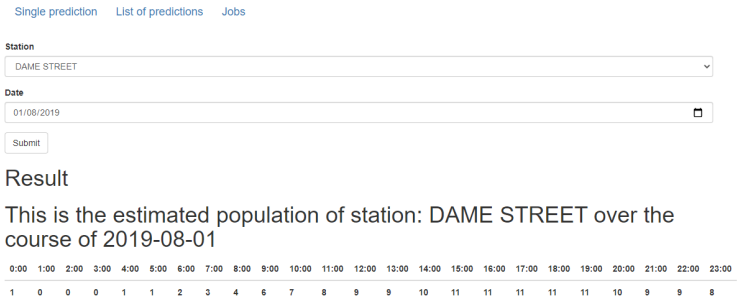

Figure 4.15: The flask interface for a list of predictions

Finally the job generation interface (fig 4.16). This interface uses the same methodology as described in the previous section (Prediction report generation). The system uses a pre-generated report to find stations' relative capacity (unless the report has not already been generated in which case it generates one there). The relative capacity of each station is then used to find the most problematic stations (IE: stations near 0 or 100% capacity) and generates jobs for them, as they are considered to be the highest priority. The system then save the job, simulates the job in the current state of the BSS and recalculates the relative capacity until no jobs remain.

However in this section the jobs are then presented to the user in a user-friendly way, with both the station ID, as well as the station name.

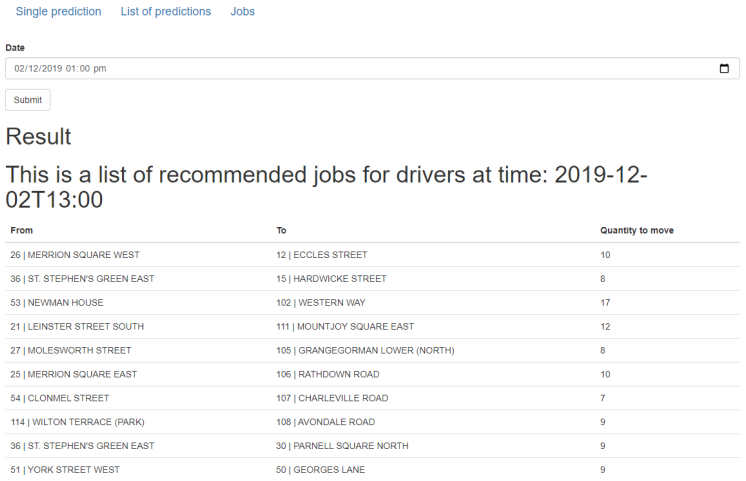

Figure 4.16: A list of jobs

Chapter 5: Project Workplan

5.1 Timeline

This timeline of events aims to complete the project over the 14 weeks of trimester 2 (18 January - 23 April)

- Data collection, pre-processing and data Representation (Week 1-3)

  While the data collection is complete, pre-processing and representation still needs to be done

- Data analysis (Week 3-4)

  Analysing interesting patterns and preparing it for the machine learning implementation

- Project implementation & results (Week 4-10)
  - Implementation of a machine learning model to predict the population of bike stations based on clean data from the previous step
  - Suggesting optimal number of bikes to relocate using data from the previous step.

- Testing and Validation (Week 11-12)

  Using a combination of quantitative evaluation for the algorithm and qualitative for the user interface to evaluate the projects effectiveness

- Project final Report (Week 13-14)

  I am to write parts of the report as I am doing the project as I will be able to more accurately describe my experience while it is fresh in my head

5.2 Gantt Chart

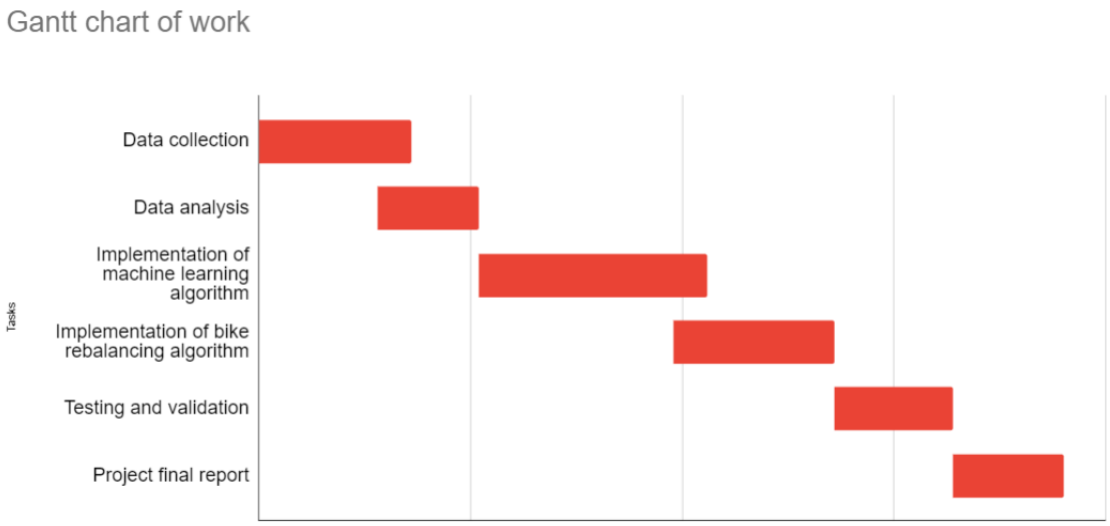

Figure 5.1: Gantt chart of estimated work timeline

Chapter 6: **Summary and Conclusions**

The metrics used to evaluate the performance of the algorithm indicate that it is not perfectly accurate. More work could be done in future, or it could be improved with a larger, more high-quality dataset (more historical BSS data/more granular weather data). However for the purposes of this project, the performanced achieved should be more than enough to give adequate guidance to users of Dublin's BSS and would indicate that the project is largely a success.

The approach explored here as a whole, could reasonably be applied to any city's bike sharing system, as there are no aspects which are specific to Dublin. The trends in other BSSs could be drastically different however, as some cities may have a more impactful seasonal trend, or are more affected by weather with regards to bike demand. However the experiments outlined here are broadly applicable, and should be able to be adapted to other BSSs with little difficulty.

For future work I would like to see the frontend developed further to create a usable application for the public. When there exists a project such as this with an easy to understand real world benefit, a quite useful application could be produced. Not only would this provide a useful tool for the public, it can also be used as a teaching example for LSTM and the power of neural networks.

In this project I present an applicable way of using historical bike sharing system data, augmented with historical weather data to predict future population of BSS stations through use of a long short term memory recurrent neural network. The approach explores the impacts of the weather data on predictions, and a method for examining the impacts of each feature in particular on the LSTM. The results produced are sufficient for the purposes of this project, however there is some room for improvement with regards to the accuracy of predictions.

Acknowledgements

I would like to thank professor Tahar Kechadi for his input and guidance throughout this project. I would also like to thank both smart dublin and Met Eireann for their fully public datasets which formed the backbone for this project.

Attribution

Bibliography

1. *Number of BSS* https://www.seattletimes.com/seattle-news/transportation/will-helmet-law-kill-seattles-new-bike-share-program/. (accessed: 15.12.2020).
2. *Dublin commuter stats* https://www.cso.ie/en/releasesandpublications/ep/p-cp11eoi/cp11eoi/dtpn/. (accessed: 15.12.2020).
3. *Dublin bikes leniency policy* http://www.dublinbikes.ie/How-does-it-work/Frequently-Asked-Questions/Stations-and-bikes#faq0. (accessed: 15.12.2020).
4. *Bike sharing carbon impact* https://www.triplepundit.com/story/2014/value-bike-sharing-looking-beyond-carbon-emissions/41181. (accessed: 15.12.2020).
5. Reiss, S. & Bogenberger, K. A Relocation Strategy for Munich's Bike Sharing System: Combining an operator-based and a user-based Scheme. *Transportation Research Procedia* **22,** 105–114. ISSN: 2352-1465. http://www.sciencedirect.com/science/article/pii/S2352146517301515 (2017).
6. Zhang, J., Meng, M. & David, Z. W. A dynamic pricing scheme with negative prices in dockless bike sharing systems. *Transportation Research Part B: Methodological* **127,** 201–224. ISSN: 0191-2615. http://www.sciencedirect.com/science/article/pii/S0191261518310270 (2019).
7. Pan, Y., Zheng, R. C., Zhang, J. & Yao, X. Predicting bike sharing demand using recurrent neural networks. *Procedia Computer Science* **147.** 2018 International Conference on Identification, Information and Knowledge in the Internet of Things, 562–566. ISSN: 1877-0509. http://www.sciencedirect.com/science/article/pii/S1877050919302364 (2019).
8. A Destination Prediction Network Based on Spatiotemporal Data for Bike-Sharing. https://www.mendeley.com/catalogue/e6c818e5-0b61-3730-a4b2-4e2ce08c21ff/ (2019).
9. Deng, Z., Tu, A., Liu, Z. & Yu, H. Efficient Spatial-Temporal Rebalancing of Shareable Bikes (Student Abstract). *Proceedings of the AAAI Conference on Artificial Intelligence* **34,** 13775–13776. https://ojs.aaai.org/index.php/AAAI/article/view/7159 (Apr. 2020).
10. Quach, J. & Malekian, R. *Exploring the weather impact on bike sharing usage through a clustering analysis* 2020. arXiv: 2008.07249 [cs.LG].
11. *Dublin bikes historical data* https://data.smartdublin.ie/dataset/dublinbikes-api. (accessed: 15.12.2020).
12. *Historical weather data* https://www.met.ie/climate/available-data/historical-data. (accessed: 15.12.2020).
13. *Dublin's long-awaited wheel deal on track for September roll-out* June 2009. https://www.irishtimes.com/news/dublin-s-long-awaited-wheel-deal-on-track-for-september-roll-out-1.783944?via=mr.