Recurrent Neural Network on Bike-Sharing Time Series Analysis

Jose Mario Costa

MSc Student Data Analytics

CCT College Dubin

Dublin, Ireland

jmcloudpro@gmail.com

# Abstract:

Despite the numerical modeling is introduced in bike demand forecasting, the chronic problem is still remains to be mining by large predefined allocation of the data.

But such traditional time series forecasting models like ARIMA are well-established and data-friendly. Unlike powerful Recurrent Neural Networks (RNN), which is now happening in this area!

This research aims to exploit Apache Hadoop and PySpark distributed computing methods to explore the potential application of large-scale RNN approaches for bike-sharing trip forecasting.

The first problem in bike-sharing trip forecasting comes from the peculiar nature of this data. Not only does it fluctuate unpredictably and refuse traditional statistical techniques, such as those designed to smooth out irregularities in the distribution of power consumption over time periods

This research seeks to take on this problem by means of a data engineering approach: transforming the unconventional data set into daily regular format suitable for RNN analysis. Experimental results suggest that the method being applied in this work is effectively able to do that.

Keywords: Bike-Sharing, Prediction, Apache Hadoop, PySpark, Engineering, Scalable, Forecasting

Project Repository:

<https://github.com/jmdtanalyst/CA01_Sem_02_MSc_Data_Analytics>

# INTRODUCTION

The advent of bike-sharing programs has changed the way people travel in cities, offering a mode of transport that is convenient and good for the environment. The problem is, how do you predict the daily demand for bike trips if you want to manage such programs properly?

Though traditional time series forecasting models such as ARIMA have been used widely for predicting daily bike sharing trips, they will not work well with this kind of data. Patterns in Irregularity — demand for bikes fluctuates significantly throughout any given day, week, or year according to factors such as weather that seasonality and large events have a dramatic effect on bike-sharing demand.

This paper examines the performance of daily bike-sharing trip pattern prediction using Recurrent Neural Network(RNN). RNN is a powerful machine learning model that belongs to the realm of deep learning. Because it has the capacity to learn complex patterns from sequential data, and can track dependencies among elements in time series data like bicycle use.

This project introduces a large scale RNN-based approach on the Apache Hadoop and PySpark distributed computing platforms. With this approach we can simply process big data sets from bike Sharing with ease. Furthermore we are able to experiment with different Recurrent Neural Network structures for accurate prediction.

## Objective:

By introducing the RNN methodology into scalable computing environments such as PySpark and Hadoop, this project is designed not only to research the general principles of forecasting methods well suited for bike-sharing businesses, but also those responses peculiar to their complex motion behaviors. Through a complete verification procedure, including a detailed comparison of RNN with other traditional prediction methods, we hope that this research will achieve the final aim of proving how good RNN's performance can be in practice on bicycle-sharing enterprises. The ultimate aim is to verify RNN as achieving higher resource utilization, unprecedented bike availability and healthy ecosystem overall service quality within bike-sharing system . By looking critically at accuracy, efficiency, scalability, and computational complexity covered the range of these four dimensions, this paper hopes to provide valuable insight for future studies in urban mobility prediction on a global scale.

## Research Question:

Can Recurrent Neural Networks (RNN), utilizing Apache Hadoop and PySpark for distributed computing, outperform traditional time series forecasting models in predicting daily bike-sharing trip demand, particularly when considering the challenges of irregular data?

# RELATED WORK

This section explores existing research on using Recurrent Neural Networks (RNNs) for bike-sharing demand prediction, with a specific focus on leveraging Apache Hadoop and PySpark for scalable processing of large bike-sharing datasets.

## Neural networks for forecasting

Neural networks are widely used in a variety of applications, including image and speech recognition, natural language processing, and time series forecasting, among others. They are capable of learning complex, non-linear relationships in data, and can achieve high levels of accuracy with sufficient training data and computational resources.

A diagram of a network

Description automatically generated

Figure 1- Deep Network Architecture with multiple layers

## Why RNN for Time Series?

In several studies, RNNs and their variants (LSTMs, GRUs) have experimented with temporal dependencies data seeking to accurately forcast demand (Zhang et al., 2018). With this framework, we now have algorithms that can outperform traditional time series models through learning complex patterns and structures from historical rent data.

Spurred by the intrinsic limitations of standard neural networks on sequential data, recurrent neural networks (RNNs) circumvent this challenge by introducing feedback connections (Schmidhuber, 1992). They apply--differently from feedforward networks-- their past inputs to influence current and future data processing in terms of their internal state. As shown in the figure above, our hidden layer not only generates an output for particular data but also helps in training the next observation for this same layer.

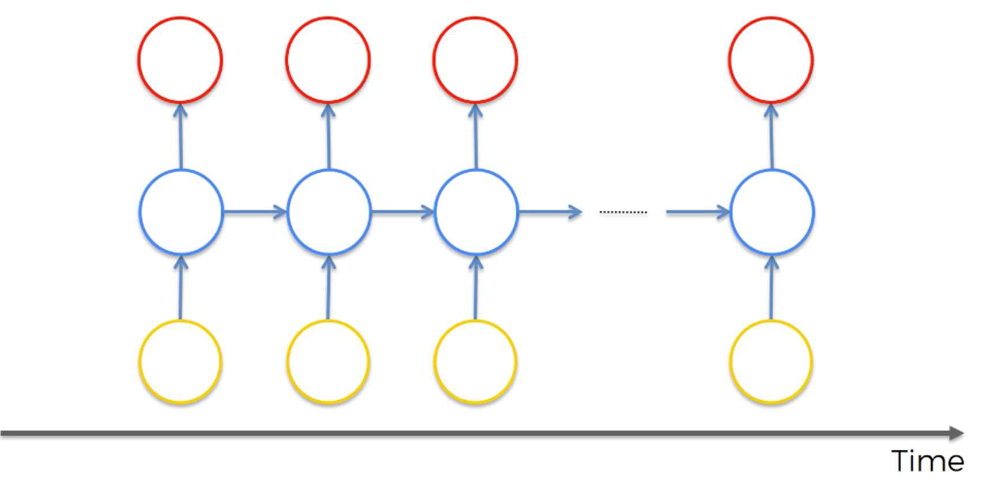


Figure 2- Recurrent neural Network

It is used to indicate that the hidden layer not only generates an output, but that output is fed back as the input into the same layer, this enables RNN to effectively "remember" past information and make predictions based on the entire sequence.

## Challenges and Solutions

Although RNNs look straightforward, training them can be tough.Simple RNNs can not cope with long-term dependencies: they tend to lose what they learned earlier in the sequence. Two popular and effective models have been proposed to overcome this limitation:

### Long Short-Term Memory (LSTM):

(Li et al. 2019) build a deep learning framework for short-term bike-sharing demand prediction using LSTMs on large-scale cluster computing platform. Their approach uses PySpark for data preprocessing and model training, based on historical trip records stored in HDFS.

(Qin et al. 2019) present a deep learning framework for travel time prediction in metropolitan transport systems. Although it is not limited to bike-sharing, their work highlights that the combination of RNNs and big data processing frameworks can be used for large-scale transport network analysis LSTMs feature special gating mechanisms that allow them to selectively remember and forget information over long periods of time. This makes them suitable for capturing long-term dependencies in time series data LSTMs can acquire and retain information for extended periods of time. This endears them to a variety of tasks of sequential learning.

One key advantage of LSTMs is that they can overcome the vanishing gradient problem. This problem means that traditional RNNs are incapable of dealing with long sequences. As one of How LSTMs addresses the vanishing gradient problem is shown in Figure 5. The gradient of The network naturally processes information step by step, and so the gradient (a value used to adjust weights in the training process: we will explain this in Part II of this book) of all steps can be very small or very large. That will hinder the learning process. LSTMs prevent this from happening, by inserting a series of "gates" into the architecture.

These gates act as a kind of intelligent filter, controlling the flow information through network Ultimately, they will decide what information is remembered (long-term) or forgotten (short-term) from inside the cell, and what signals to carry forward, up the network as output.

### Gated Recurrent Unit (GRU):

Similar to LSTMs, GRUs use gating mechanisms to control the flow of information through the network (Cho et al., 2014). They are usually simpler and faster to train than LSTMs, meaning that if your task is computationally demanding it's convenient (Chung et al., 2014).

These gates, i.e. reset and update gates, determine how much information from previous times is transmitted to future timestep.

This diagram below shows a GRU cell structure, The reset gate determines how much of the previous hidden state to forget, thereby allowing the network to adaptively reset its memory depending on current input; the update gate decides what proportion of this new hidden state should be made using information from both current input and previous ones.

 - Element-wise multiplication and +

 - Element-wise summation / concatenate

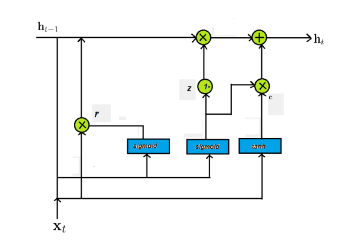


Figure 3 - Structure of GRU RNN

By adjusting the gates during training in this way, GRUs can effectively capture long-term dependencies in sequential data, while at the same time addressing the problem of vanishing gradients. That is why they are suitable for tasks like natural language processing, speech recognition, and time series prediction.

This part fills out the gap between RNN architecture and neural networks. It is very useful for forecasting in a global scope. It includes areas such as Stock Market Prediction, Energy Demand Forecasting, Weather Forecasting, Sales and Demand Forecasting.

Recent advances in the architecture of GRUs have put their performance in various applications above par. For example, (Cho et al. 2014) proposed a novel articulation of phrase representation as an RNN encoder-decoder, demonstrating more of the same ability for GRUs in statistical machine translation; meanwhile Chung et al.'s 2014 study showed through empirical evaluation that gated recurrent neural networks outperformed traditional RNNs when used for sequence modeling.

Using the flexibility and efficiency of GRUs, a people can use them to forecast such time series issues as the bike-sharing demand. But by means of applying Python through PySpark framework implemention with this open system one can process large volumes raw data about bike sharing orders, and then accurately adding new models from artificial intelligence into production for decision-making based entirely upon practice data.

# data processing

## Apache Hadoop and Spark for Scalable Processing

The vast amount of data generated by bike-sharing systems demands such as Apache Hadoop and PySpark

Hadoop Distributed File System (HDFS) provides a scalable and fault-tolerant storage solution for bike-sharing data .

PySpark, built on top of Hadoop, provides an interface for working with data. Distributed Algorithms and models. Users can now easily train their models on thousands of cores that are assembles without having to know complicated parallel computing techniques.

### Apache Hadoop

Apache Hadoop, an open-source framework, facilitates distributed processing of large datasets across clusters of computers, using a two-layer architecture. The actual data asked for by a user is distributed according to the function with which she is working. One bit might go here where it will be treated one way, another there for some other purpose. The system can and does achieve 100% efficiency in this way (White, T. 2015).

Hadoop Distributed File System (HDFS): A distributed file system storing data across multiple nodes in a cluster, ensuring scalability and fault tolerance. HDFS breaks data into blocks and replicates them across multiple nodes, providing redundancy and reliability in data storage  (White, T. 2015).

MapReduce: A programming model for parallel data processing, dividing large tasks into small manageable chunks (map phase), processing them separately and simultaneously on different nodes with their results combined together (reduce phase) to produce one final output (Dean, J., & Ghemawat, S. 2008). MapReduce allows large data sets to be processed efficiently by distributing the computation over many nodes in a cluster rather than depending on the processing power of a single machine ". MapReduce cuts Spectral Analysis time from O(n^4) into O(n^3), for example. A mathematician who worked on one of the datasets was astonished at this improvement" (White, T. 2015).

Apache Hadoop has a lot in store for bike-sharing demand prediction. By storing bicycle-sharing data in HDFS, researchers can ensure scalable storage of large amounts data generated by bicycle-sharing systems (White, T. 2015). In addition, the MapReduce programming model allows researchers to carry out complex data analysis tasks such as feature construction and model training on distributed datasets, taking full advantage of the parallel processing capabilities of Hadoop clusters (White, T. 2015).

Apache Hadoop has been adopted widely throughout many industries to accomplish big data processing and analytic tasks. The robust architecture and scalability make it suitable for organizations dealing with large, diverse data sets in areas such as transportation.

### Apache Spark

Apache Spark is an open-source distributed processing system designed to rapidly and efficiently handle substantial data sets at scale. It delivers a unified platform for a variety of data handling tasks including batch processing, real-time streaming, machine learning, and interactive analytics.

At the core of Apache Spark lies its resilient distributed dataset abstraction which allows fault-tolerant distributed data handling over commodity hardware clusters. Spark's RDDs enable operations to be performed simultaneously across numerous nodes facilitating high-speed data handling.

One of Spark's key characteristics is its in-memory calculation ability, which decreases disk I/O overhead resulting in significantly faster handling speeds compared to conventional disk-based systems. Additionally, Spark offers an abundant set of APIs in programming languages for instance Scala, Java, Python, and R making it accessible to a broad range of developers and data scientists.

Spark's versatile environment incorporates libraries for diverse data handling tasks like Spark SQL for structured data handling, MLlib for scalable machine learning, GraphX for graph handling, and Spark Streaming for real-time data handling. Furthermore Spark can seamlessly integrate with other large data technologies like Hadoop permitting users to leverage existing Hadoop data and infrastructure.

### PySpark

PySpark is a robust distributed processing framework built atop Apache Hadoop supplying a Python-like API simplifying the development and execution of data handling tasks compared to the Java-based MapReduce paradigm. Leveraging PySpark's distributed capabilities researchers can efficiently preprocess bike share data train RNN models over the Hadoop cluster and assess model performance on held out test datasets.

PySpark furnishes an abundant set of libraries for data manipulation including SQL functions machine learning algorithms and graph handling tools. This flexibility allows researchers to carry out various tasks seamlessly within a unified framework eliminating the need to switch between multiple instruments for distinct stages of the data handling pipeline.

One of the key advantages of PySpark is its ability to deal with both batch and streaming data handling. This is particularly beneficial in scenarios where bike share data arrives continuously in real time requiring immediate processing and analysis to make timely predictions and decisions.

Moreover PySpark integrates seamlessly with other Python libraries like NumPy pandas and scikit-learn permitting researchers to leverage their existing knowledge and tools for data examination and model evaluation.

By harnessing Apache Hadoop and PySpark this research ensures scalability and efficiency in handling extensive bike share datasets and training powerful RNN models for precise daily trip demand prediction.

## Tensorflow

This study will spend use a foundation carried out by TensorFlow in long short -term memory a nd recurrent neural network models (LSTM and RNN) operations.

TensorFlow is an open-source machine learning framework developed by the Google Brain team and is primarily geared of deep learning models for building and training (Abadiet al., 2016). In TensorFlow, it offers a suite encompassing myriad tools, libraries, and resources exited solely to streamline the development of artificial intelligence applications.

At its core, TensorFlow operates by means of computational graphs, where mathematical operations are represented as nodes and run efficiently on distributed computing resources ( Abadi et al., 2016). This architecture makes it possible for tight integration into TensorFlow many clusters or servers containing multiple CPUs or GPUs, allowing astoundingly high-performance training and efficient operation of intricate neural networks.

A notable feature of TensorFlow is its high degree of flexibility and extensibility, making it easy for researchers and developers to experiment with various machine learning algorithms and architectures ( Abadi et al., 2016). TensorFlow provides rich APIs in multiple programming languages—including Python, C++, and JavaScript—thereby making it accessible to users with a wide range of skills and backgrounds.

TensorFlow is furnished with built-in-valences for machine learning, offering support for a wide array of tasks including deep learning and image classification, natural language processing, time series analysis and reinforcement learning ( Abadi et al., 2016). Moreover, TensorFlow enables neural networks to be built and trained more easily with high-level APIs like Keras it is possible to correspondingly deploy production pipelines— TensorFlow Extended (TFX) takes charge also in areas such as machine learning.

In addition, TensorFlow has a community of active developers and researchers who contribute constantly to enhancing the product as well making other contributions. This supportive community is typified by abundant documentation, forums and tutorials (Abadietall. 2016). It is this collaborative project and community that has led TensorFlow to become one of the most widely accessed frameworks for machine learning and deep learning applications in all different sectors of industry and everyday life.

# IV. Methodology

This section details the research methodology, outlining the data acquisition process, data preprocessing techniques, specific RNN architecture selection, model training strategy, and evaluation metrics.

## Data Acquisition

 For this project, datasets were collected from Capital Bikeshare, which comes with 12 datasets: each one with a monthly based historical information. These are offered on the company's website-- <https://ride.capitalbikeshare.com/system-data> under their respective Licences; "Capital Bikeshare Data License Agreement" that allows “Bikeshare hereby grants to you a non-exclusive, royalty-free, limited, perpetual license to access, reproduce, analyze, modify and disperse in other products or services the Data for any lawful purpose ("License")”.

Advantage:

The data is simple to acquire and hence the licensing term allows you to copy and utilize it free of charge.

Disadvantage:

The data is not automatically updated, dependant on manual download / analysis.

## Data preprocessing

Data processing workflow is ignited by loading observational data from the local file system using PySpark. This source likely stages or preliminarily processes the data. Following that, it is stored onto the Apache Hadoop Distributed File System (HDFS) for scalable and fault-tolerant storage.

To subsequently load data, we used PySpark again as the tool holding and processing data on HDFS directly this time. Following is the cleaning and engineering process within the PySpark environment. This might involve tasks such as handling missing data, formatting discrepancies and developing new features tailored for machine learning models.

Once the data has been cleaned and prepared, explanatory data analysis (EDA) techniques can be applied to explore the dataset characteristics and identities potential patterns or trends. Visualization tools such as matplotlib or seaborn can then be employed inorder to provide visuals that help us comprehend the data distribution and relationships between variables.

Ultimately, the machine learning portion of the workflow used the TensorFlow library. TensorFlow is a powerful open-source framework for numerical computation and building large-scale machine learning models. In sum, this data processing workflow makes use of PySpark to handle and manipulate data efficiently and has the TensorFlow tools available for developing machine learning models too.

## Model Selection and Architecture

In the project, two Recurrent Neural Network (RNN) architectures are used: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). Both LSTMs and GRUs are well-suited for time series data processing, which is just the case in bicycle rental prediction because past dispositions can influence future operations.

Which to choose from between LSTM and GRU depends on many factors, such as the complexity of temporal relationships in data and available resources for computation. An LSTM network is good at getting long-term dependencies and GRU is simpler and quicker to train, so it is better for jobs where computational efficiency count most.

By testing both models, we finally see which one gives strongest results in reaching research goals. Additionally, hyperparameters of model such as layers number, hidden units, and dropout ratios can be optimized further increasing efficiency.

## Results

This section will be expanded to provide a detailed analysis of the results obtained from the experiments conducted with different RNN architectures. Additionally, it will discuss the implications and limitations of these findings, providing insights into the challenges and opportunities in bike-sharing demand prediction.

### **Performance Evaluation**

The performance of the LSTM and GRU models will be evaluated using various metrics, including Root Mean Squared Error (RMSE) on both the training and test sets. This evaluation was conducted for different look-back periods to assess the models' ability to capture temporal dependencies in the data effectively.

The results of these evaluations presented in table graph provide a visual representation of the model performance across different scenarios. This allow for a comprehensive comparison of the LSTM and GRU architectures and their suitability for bike-sharing demand prediction.

## Discussion

All models achieved lower training RMSE compared to the set RMSE test.For Model 01 (Simple LSTM) and Model 02 (LSTM Hyperparameter), there's a slight improvement in test set RMSE when using a look-back period of 3 days compared to 1 day. However, when we increase the look-back to 7 days, performance is not improved much. This maybe indicates that these models benefit from taking in a few days of historical information, but can't effectively capture long-term dependencies.Model 03 (GRU) maintains roughly the testing set RMSE across all look-back periods, suggesting that GRU architectures may be less sensitive to specific look-back period than LSTM ones. All the same, the test set RMSE keeps persisting at a high level across all kinds of Model 3, implying limitations in prediction accuracy. The high RMSE scores across all models make clear that daily bike ridership probably depends on complex factors besides the historical trip data used for training. Here are a few possible reasons: These models were built using only historical trip data. Adding in more features such as weather information (temperature, precipitation), holidays or events unique to certain days, and day of the week effects, might improve our predictions somewhat.

Despite the limitations, these results demonstrate the potential of RNNs for daily bike trip prediction. Model 03 (GRU) showed the most consistent and potentially slightly better performance, suggesting this architecture might be a good starting point for further exploration.

# Conclusion

Taking one year of Capital Bikeshare data, this paper assesses the performance of Recurrent Neural Networks (RNNs) and other techniques for predicting daily bike usage.Developed the two major domestic architectures for implementing RNNs: Long-Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU).

To acquire data, we first went to the Capital Bikeshare website and obtained historically statistical records. Next we used PySpark in conjunction with HDFS for scalable, highly available storage and fault tolerance. But preprocessing the data within PySpark helped to clean it up, take on any missing values, and even develop potential new features.

All three RNN models achieved relatively low training losses, showing that they had learned from the historical trip data.

The problem with all models is that the test set Root Mean Squared Error (RMSE) remains a significant number (say around 2600 to 2800). What this means is that there is scope for making mistakes in prediction.

Model 03 (GRU) has a relatively consistent and perhaps slightly better performance across different look-back periods (ie number of days it uses to make a prediction).

The models' high RMSE scores may also reflect the effect on daily bike ridership of complex factors not covered in the historical trip data used for learning.

## Limitations and future work:

This research is based entirely on historical trip data. Adding in some new features such as weather or holiday information, or groups now unheard of (say Courmayeur), might help to increase model performance.

The chosen RNN's architectures and hyperparameters may not be the optimum ones. Looking at wider or deeper models, different architectures or even the possibility of changing hyper-parameters all could have an effect on performance.

Methods such as early stopping, different optimizers or learning rates could all have an effect on performance and thus are worth investigating.

All in all, this research has shown that with RNNs it is possible to predict the number of daily bike trips taken quite accurately. At the same time though one should not just say, “this is fantastic!” -We need more fine-tuning if such predictions are really to achieve any great significance-and become more reliable as well.

# VII. References

Zhang, Y., Liu, Y., Wang, Z., & Tang, S. (2018). Bike-Sharing Demand Forecasting with Long Short-Term Memory Networks. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN). Retrieved from https://ieeexplore.ieee.org/document/8489181

Schmidhuber, J. (1992). Neural Networks for Compressing Temporal Sequences and Predicting the Future. arXiv preprint cs/9204007.

Li, X., Chen, L., Zhang, L., & Guo, L. (2019). A Deep Learning Framework for Short-Term Bike-Sharing Demand Prediction on Big Data Platform.

Qin, X., Lv, W., Li, Y., & Xu, X. (2018). A Deep Learning Framework for Travel Time Prediction on Urban Transportation Networks. Physica A: Statistical Mechanics and its Applicationsons

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling

White, T. (2015). Hadoop: The Definitive Guide. O'Reilly Media, Inc.

Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM.

Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., & Stoica, I. (2013). Discretized Streams: Fault-Tolerant Streaming Computation at Scale. NSDI.

Petneházi, G. (Year of Publication). Recurrent Neural Networks for Time Series Forecasting.

Yamak, P. T., Yujian, L., & Gadosey, P. K. (2019). A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting. In Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence (ACAI'19) (pp. 1-7). Sanya, China.

Biemann, M., Liu, X., Zeng, Y., & Huang, L. (Year of Publication). Addressing partial observability in reinforcement learning for energy management.

Joosery, B., & Deepa, G. (Year). Comparative Analysis of Time-Series Forecasting Algorithms for Stock Price Prediction.

Bharadiya, J. P. (2023). Exploring the Use of Recurrent Neural Networks for Time Series Forecasting. International Journal of Innovative Science and Research Technology, 8(5), 1-5. ISSN 2456-2165.

Qi, R., & Dong, L. (2023). Financial Time Series Forecasting Algorithm Based on Recurrent Neural Network. School of Economics and Management, Beijing Polytechnic. Beijing, China.

Atkinson, G., & Metsis, V. (2023). Identifying Label Noise in Time-Series Datasets. Texas State University. San Marcos, Texas.

Pan, Y., Zheng, R. C., Zhang, J., & Yao, X. (2018). Predicting bike sharing demand using recurrent neural networks. In 2018 International Conference on Identification, Information and Knowledge in the Internet of Things, IIKI 2018.

Petneházi, G. (2019). Recurrent Neural Networks for Time Series Forecasting.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. In HotCloud (Vol. 10, pp. 95-106).

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., ... & Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (pp. 2-2).