TILBURG UNIVERSITY

PREDICTING LAP TIMES IN A FORMULA 1 RACE USING DEEP LEARNING ALGORITHMS

A COMPARISON OF UNIVARIATE AND MULTIVARIATE TIME SERIES MODELS

FLEUR BRUSIK

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
AT THE SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
OF TILBURG UNIVERSITY

STUDENT NUMBER
2039129

COMMITTEE
dr. Boris Čule
Federico Zamberlan

LOCATION
Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science & Artificial Intelligence
Tilburg, The Netherlands

DATE
June 17th, 2024

WORD COUNT
8794

PREDICTING LAP TIMES IN A FORMULA 1 RACE USING DEEP LEARNING ALGORITHMS

A COMPARISON OF UNIVARIATE AND MULTIVARIATE TIME SERIES MODELS

FLEUR BRUSIK

Abstract

Predictive modeling for professional racing has proved to be a challenge which can be overcome only when domain knowledge of racing is integrated in the modeling techniques. In particular, predicting lap times has proved to be challenging due to substantial external factors. Previous work has suggested that complex time series forecasting scenarios like these require multivariate forecasting rather than univariate forecasting. Therefore, this study proposes to compare multivariate and univariate time series models in predicting lap times in a Formula 1 race. To this end, two sets of univariate and multivariate deep neural architectures are compared to each other when tested on the official Formula 1 timing data from the seasons 2018-2023. Ultimately, the dataset is divided into low complexity and high complexity races, allowing us to examine how univariate and multivariate models compare to each other when dealing with different levels of complexity in time series. This study shows that complex forecasting scenarios like predicted lap times in a Formula 1 race are most accurately tackled by multivariate models. Nonetheless, dealing with the external factors associated with these complex forecasting scenarios remains a challenge for these models. In addition, this study advocates the manual selection of features by means of domain knowledge of the forecasting scenario at hand. Furthermore, this study contributes to the field of feature extraction mechanisms in deep neural networks by offering some premises for future research.

1. Problem Statement and Research Goals

Formula 1 is a data driven and strategic sport, in which every strategic call is crucial for the optimal end result. The most important ingredient for winning a race is setting the fastest lap times compared to your competitors. Therefore, a model that is able to predict lap times is a huge strategic asset in a race. Furthermore, the models and methods that are derived from this research can be applied in various other sectors that benefit society. For example, it could be used for predicting times in public transportation whereby schedules can be optimized. In addition, it can be utilized to predict energy consumption patterns, potentially leading to an improved energy consumption and enhanced sustainability.

However, predicting lap times in Formula 1 is not straightforward, as a race can be very dynamic, with many external factors that impact the lap times. For example, Tulabandhula and Rudin (2014) concluded that the challenges that are faced in designing a prediction and decisions system for professional racing can be overcome only when domain knowledge of racing is integrated in the modeling techniques. The track and race characteristics are race dependent, which severely affect the lap times (Tulabandhula & Rudin, 2014).

Predicting lap times in a Formula 1 race is an application of time series forecasting. Traditional methods that are generally used for time series are univariate forecasting techniques. In a purely univariate forecasting problem, the future values of a time series are predicted only on its own past values. So, no exogenous variables are used in predicting future values (Hewamalage et al., 2021). Hewamalage et al. (2021) state in their empirical study on Recurrent Neural Networks (RNN's) for time series forecasting that univariate forecasting may not always be the best method in any forecasting scenario. They state that complex forecasting scenarios where the time series units may be interdependent require multivariate forecasting rather than univariate forecasting. Complex time series are those time series that are affected by conflicting causal forces (Armstrong et al., 2005).

Given the challenges in predictive analysis for professional racing that were outlined above, it can be assumed that predicting lap times in a Formula 1 race is such a complex forecasting scenario which would potentially be most appropriately tackled by means of multivariate forecasting methods. Following these insights in literature, the following main research question can be formulated:

How do multivariate time series models compare to univariate time series models in predicting lap times in a Formula 1 race?

The main research question can be divided into several sub-questions. Recurrent Neural Networks have proven to be competitive forecast methods for time series analysis. Specifically, Hewmalage et al. (2021) conclude in their extensive empirical study of existing RNN architectures that a stacked architecture combined with Long Short-Term Memory (LSTM) cells fed with data in a moving window format is a generally competitive model across many different types of datasets. Hewmalage et al. (2021) have only tested pure univariate forecasting scenarios. Therefore, it would be beneficial to know if any improvement is achieved in predicting lap times in a Formula 1 race if a Multivariate LSTM is implemented.

SQ1: *To what extent can a Multivariate LSTM with domain specific exogenous variables outperform a Univariate LSTM in predicting the lap times in a Formula 1 race?*

A time series model extracts features that represent time series patterns. In an attempt to facilitate this procedure, Karim et al. (2017) propose to augment a Fully Convolutional Network (FCN) with LSTM RNN for time series classification, where the FCN functions as a feature extractor and a temporal relationship detector. In this light, Karim et al. (2019) propose to transform the univariate Long Short-Term Memory Fully Convolutional Network (LSTM-FCN), into a multivariate time series classification model by expanding the fully convolutional block with a squeeze-and-excitation block. They found that the multivariate models perform significantly better than univariate models. This result is ascribed to the squeeze-and-excitation block which is able to model the interdependencies between the variables in a multivariate time series model (Karim et al., 2019). It would be useful to explore whether these models that were used for time series classification perform well when applied to this study's time series forecasting problem.

SQ2: *To what extent can a Multivariate LSTM-FCN with domain specific exogenous variables outperform a Univariate LSTM-FCN in predicting lap times in a Formula 1 race?*

This study found that a multivariate model is able to more accurately predict the lap times in a Formula 1 race, and therefore is assumed to be better suited to tackle complex

forecasting scenarios than univariate models. Nonetheless, dealing with the external factors associated with these complex forecasting scenarios remains a challenge for these models.

2. Literature Review

2.1 Recurrent Neural Networks for Time Series Forecasting

Hewamalage et al. (2021) find that a RNN with LSTM cells is generally competitive across many times series forecasting scenarios. Testing the LSTM on datasets taken from several forecasting competitions demonstrated that it was able to outperform the benchmark techniques for almost all datasets. More details on the used datasets and performance in this work can be found in Appendix A, Table 1. Furthermore, Hewamalage et al. (2021) conclude that RNN's are able to capture seasonality without deseasonalization of the data when all series have homogeneous seasonal patterns, with sufficient lengths that cover the same duration in time.

It can be argued that lap times in a Formula 1 race exhibit a mix of both heterogeneous and homogeneous seasonal patterns. On the one hand, lap times have homogeneous seasonal patterns, as they follow predictable and regular patterns across races. For example, lap times tend to be faster after pitstops, or they increase during stints due to tire degradation. On the other hand, Formula 1 races and lap times exhibit heterogeneous seasonal patterns due to exogenous factors that affect the lap times, like changing weather conditions and incidents.

Therefore, in predicting lap times in a Formula 1 race it is crucial to take both the heterogeneous and homogeneous seasonal patterns into account. As outlined above, the homogeneous patterns in Formula 1 race time series can be accounted for by a univariate LSTM. To capture the heterogeneous patterns, other modeling techniques should be implemented that can incorporate domain knowledge of racing (Tulabandhula & Rudin, 2014). This can potentially be accomplished by feeding the model some important exogenous variables in addition to the lap times, such that it becomes a multivariate LSTM.

Previous work has been done on the comparison between univariate and multivariate RNNs. Cao et al. (2012) find in their comparative analysis of the wind speed forecasting accuracy of univariate and multivariate models that the multivariate RNN models perform better than the univariate models. Moreover, their results indicate that incorporating other measurable covariates yields significant improvement in RNN models. Similarly, Zhang et al. (2004) find in their comparative analysis of univariate and multivariate models for earnings

per share forecasting that the incorporation of fundamental accounting variables improves forecasting accuracy.

2.2 Feature extraction in Recurrent Neural Networks for Time Series Forecasting

In order to leverage its full potential, a time series model has to be able to extract features that represent time series patterns. Multiple studies have experimented with feature-based approaches that can help RNNs and other deep learning models to achieve this. However, many of these methods require heavy feature engineering and feature extraction. In this light, Fully Convolutional Networks (FCNs) have been demonstrated to provide state-of-the-art performance for time series modeling. For example, Wang et al. (2017) find that the FCN achieves premium performance in time series classification compared to other state-of-the-art approaches. They demonstrate that the FCN is performed as a feature extractor, without requiring heavy data preprocessing or feature engineering.

Similarly, Bai et al. (2018) demonstrate in their systematic evaluation of generic convolutional and recurrent architectures for sequence modeling that a simple convolutional architecture outperforms recurrent networks such as LSTMs across many sequence modeling tasks. They propose a generic Temporal Convolutional Networks (TCN) designed for sequential data, which is based on two principles: the network produces an output of the same length as the input, and there is no leakage from the future into the past. Bai et al. (2018) ascribe the superior performance of TCNs over RNNs to its substantially longer memory, allowing the TCN to capture longer histories.

Karim et al. (2017) propose the Long Short-Term Memory Fully Convolutional Network (LSTM-FCN) to overcome the above mentioned problems associated with feature-based approaches in time series classification. The LSTM-FCN is the product of an augmentation of a FCN with LSTM RNN, where the TCN functions as a feature extractor and a temporal relationship detector in a FCN branch. The LSTM-FCN significantly enhances the performance of FCNs, while requiring minimal preprocessing of the data (Karim et al., 2017). Testing the model on all 85 UCR time series datasets (Chen et al., 2015), demonstrated that the LSTM-FCN at least achieves state-of-the-art performance (Appendix A, Table 1).

However, Karim et al. (2019) acknowledge that while the LSTM-FCN has high performance for univariate datasets, its performance is not optimal when applied directly to multivariate datasets. Therefore, Karim et al. (2019) introduce the Multivariate LSTM-FCN (MLSTM-FCN) for multivariate time series. They propose to transform the LSTM-FCN into

a multivariate time series classification model by expanding the fully convolutional block with a squeeze-and-excitation block. The squeeze-and-excite block models the interdependencies between variables in a multivariate time series model, leading to significant better performance of the multivariate models as compared to the univariate models (Karim et al., 2019). It was demonstrated by means of testing the MLSTM-FCN on 35 benchmark datasets that it is able to attain state-of-the-art performance (Appendix A, Table 1).

3. Methodology & Experimental Setup

Following the research gaps pointed out in the previous sections, an experiment was set up to test how multivariate time series models compare to univariate time series models in predicting lap times in a Formula 1 race. This section describes the entire experimental procedure in detail. The complete experimental procedure is visualized in a flowchart in Figure 1.

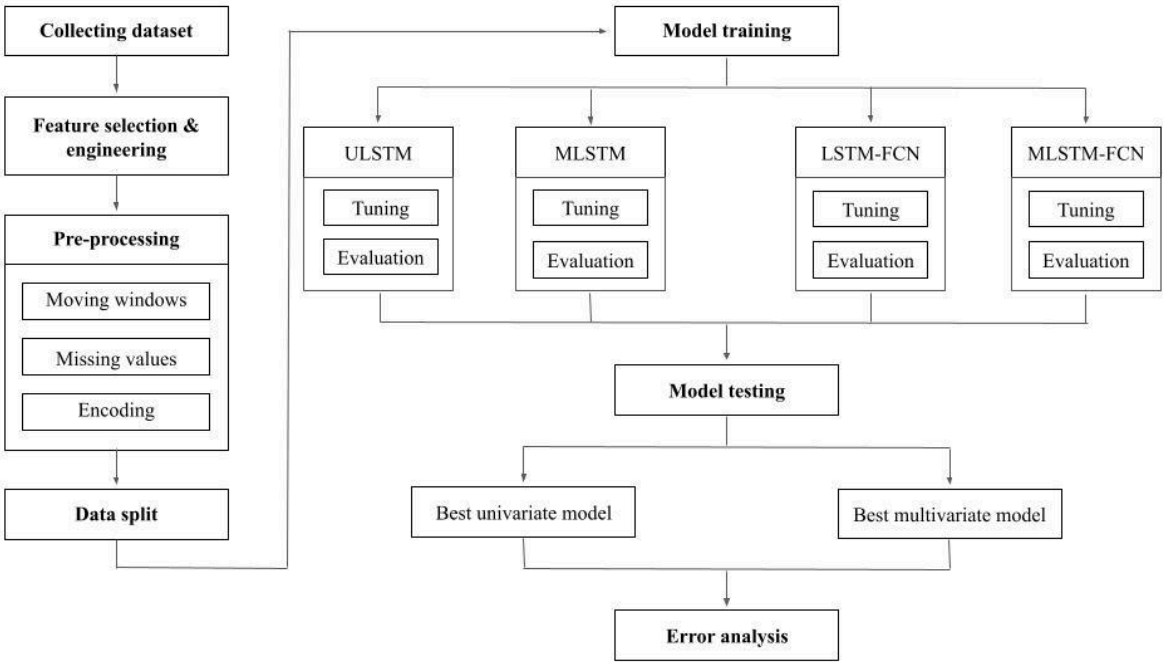

Figure 1: Flowchart of the experimental procedure.

3.1 Dataset Description

For this study, the necessary data was accessed by means of the FastF1 package (FastF1, 2024). FastF1 gives access to F1 lap timing, position data, tyre data, track data, and weather data. Amongst other, timing data and session information is only available from the 2018 season onwards (Fastf1, 2024). Therefore, the time series data will only originate from

the 2018 season onwards. The collected dataset consists of 136,122 lap times with 39 features, collected over 125 races.

Thus, the data consists of lap timing data and weather data. During a race, weather data is updated once per minute. This means that there are generally one or two data points per lap. Therefore, the weather features for a specific lap are either the first value within the duration of the lap, or the last known value before the end of the lap if there are no values within the duration of the lap (FastF1, 2024).

3.2 Feature Selection and Engineering

Regarding the multivariate models, additional features from the created dataset were carefully selected to capture the dynamics of racing in the models. An overview of all features with their characteristics can be found in Appendix B, Table 2. The choice for and the characteristics of the exogenous features are described in the following paragraphs.

Compound. Compound refers to the type of tyre that was used during the lap. This feature is known to be extremely important in modeling professional racing, as the compound type has immediate effects on lap times (Tulabandhula & Rudin, 2014). Namely, some compounds are optimal for durability while others are optimal for peak performance (Pirelli, n.d.).

Tyre Life. Tyre Life can be defined as the number of laps that were driven on this specific set of tyres. Tyre Life is a crucial feature for predicting lap times, because lap times increase throughout an outing as a result of tyre wear down (Tulabandhula & Rudin, 2014).

Stint. A stint refers to the set of laps that are performed between two pitstops, or the set of laps between a pitstop and the start or end of the race. This feature was selected because generally drivers try to minimize tyre degradation by managing and stabilizing their lap times more during stints early in the race as compared to stints later in the race.

Lap-type. This feature was not readily available from the dataset, and was instead created based on the features PitOutTime and PitInTime, which has resulted in three categories. Inlap can be defined as a lap in which the driver enters the pits, so when PitInTime is not missing. Outlap can be defined as a lap in which the driver exited the pit, so when the value for PitOutTime is not missing. When a lap is neither an Inlap or Outlap, it is categorised as a regular lap. This feature was included because a pitstop has severe effects on lap times. For example, drivers tend to drive faster in an Outlap.

Driver. The skills among drivers can vary, which in turn affects their lap times. Empirical research indeed indicates that driver characteristics play an important role in

modeling applied to professional races, as Allender (2011) found that driver years of experience play a significant role in predicting NASCAR races outcomes.

Team. Not only the drivers' skills affect lap times. Every constructor builds their own car, which makes the car's performance constructor specific. Generally, the performance of the car is heavily dependent on the constructors' expertise and other resources, which in turn impact the lap times.

Trackstatus. Irregularities occur very regularly in Formula 1, such as mechanical failures or accidents. These irregularities have to be accounted for in our predictions by including the feature Trackstatus in the models, which contains information on all types of deployed flags and safety cars in a race. Every time the track status changes, a new value is sent. This entails that this feature can have multiple values, resulting in status codes.

Position. The position in which a car is driving has substantial effects on its lap times. Namely, the fresh air effect entails that lap times are generally lower for cars in front of the group as compared to cars in the middle or back of the group (Tulabandhula & Rudin, 2014).

Lastly, a few important weather features were selected that directly affect lap times or indirectly affect them by increasing tyre wear and degradation. See Table 2 in Appendix B for an overview of all weather features.

3.3 Pre-processing

3.3.1 Input and Output Strategy

For the creation of input and output windows, a multiple-input multiple-output (MIMO) strategy was implemented, where a moving window approach is used to feed the inputs and create the outputs as in the work of Hewmalage et al. (2021) and Bandara et al. (2020). The advantage of using a MIMO strategy over a single-output strategy is twofold. First, forecasting the whole output window in one go allows incorporation of the interdependencies between the time steps, instead of predicting each time step individually (Hewmalage et al., 2021). Secondly, a single input would require the recurrent units to remember the whole sequence's history, while the use of input windows relaxes this task (Hewmalage et al., 2021).

3.3.1.1 Moving Window Approach

In a moving window approach, the model takes a window of inputs and creates a window of outputs respective to the time steps subsequent to the fed inputs. In the next time step, the model takes an input window of the same size, shifted forward by one.

3.3.1.2 Window Sizes

In this study, a sequence can be defined as the sequence of laps for one driver in one race. Regarding the choice of the window sizes, it was decided to align them with the average stint length in races. In practice, this means that the input and output windows will both be half the length of an average stint. This choice was founded on two important concerns. First, from a Formula 1 team's perspective, it is extremely valuable to get insights into how the current stint will develop. In this way, the team will know when lap times are expected to increase in the current stint, such that they can plan their next pit stop. However, from a model performance perspective, the windows need to be of sufficient length to allow the model to perform well. Concretely, this means that both the input and output windows in this study will be of length 10, given the fact that the average stint length in the dataset is 20.25 laps. As a result, the sequences with less than 20 data points were excluded from analysis.

3.3.2 Processing Missing values

The dataset contained a number of missing values that had to be dealt with appropriately. Pratama et al. (2016) state in their review of missing values handling methods in time-series data that estimation techniques are overall the best option for missing values handling, especially with large amounts of missing values. However, it can be argued that methods like mean or mode imputation are not a sensible option in this study, as all lap times and variables are heavily sequence specific, which would make imputation based on the variable in the whole data unreliable. Furthermore, the percentage missing in this dataset is only about 2% to 3%.

Given these considerations, it can be argued that deletion is an appropriate missing values handling method in this study. Pratama et al. (2016) state that basic methods such as deletion are simplistic and effective for low percentage of missingness. Only with larger percentages of missing (>15%) will these methods affect the results of analysis.

Table 3 in Appendix C gives an overview of the number of missing values per variable in the dataset. The missing values in this dataset can be categorized into three types of missing values; first laps missing, consecutive laps missing, and individual values missing. It was observed that the first lap time in a race is almost always unknown. Therefore, it was decided to delete all first laps. Furthermore, it appeared that a missing lap time in a sequence is commonly followed by one or more consecutive missing lap times. This is most likely due to technical issues with measurements in the car. Therefore, it was decided to delete the

windows that contained two or more consecutive missing values in a row, because any imputation method would incur too much unreliability.

The last category, individual values missing, required a different missing values handling method, as listwise deletion would make too many windows unusable. Therefore, the Next Observation Carried Backward (NOCB) technique was used, which uses the first valid observation after the missing value and carries it backward (Ahn et al., 2022). This technique was chosen because as mentioned in section 3.2, lap times evolve gradually by increasing throughout an outing. Therefore, NOCB is a more appropriate technique than its opposite technique Last Observation Carried Forward (LOCF), which carries forward the last valid observation before the missing value. In this work, NOCB was used first to impute the missing values for all features that had missing values. However, it might be the case that the last value in the sequence is missing, therefore after NOCB, LOCF was used to handle these last missing values.

3.3.3 Feature Encoding and Normalisation

All proposed models in this work require categorical features to be encoded. As illustrated in section 3.2, the values of feature Trackstatus consist of a code, which is the result of all status changes in that lap. This means that an observation can have multiple values for this feature. Therefore, Trackstatus was encoded by means of multi-label encoding. All other categorical features were encoded by means of one-hot encoding. Furthermore, the lap times were converted to milliseconds. Ultimately, all numerical input values were standardized to enable stable and efficient model training.

3.3.4 Splitting the Data and Predictions

The data has been splitted according to a traditional time series split which follows strict chronological order. This means that the model is trained and developed on the earliest partition of races and tested on a small partition of the latest races. The models are trained on all windows that are generated from the sequences in the train set, and the models will predict all windows in the test set that follow the first window of a sequence. Thus, the first actual prediction in a race will be done once the first window has been fed to the model.

3.4 Experimental Design and Algorithms

This subsection gives an outline of the time series forecasting techniques employed in this study. An overview of all tested models can be found in Table 4. These models are

evaluated against a baseline, which uses the last known lap time in the input window as an estimate for all lap times in the predicted window. The models were trained using the Keras library (Keras) with the TensorFlow backend (TensorFlow).

Model	Origin	Architecture
Univariate Long Short-Term Memory (ULSTM)	Hewmalage et al. (2021)	Stacked architecture with LSTM cells
Multivariate Long Short-Term Memory (MLSTM)	Hewmalage et al. (2021)	Stacked architecture with LSTM cells
Long Short-Term Memory Fully Convolutional Network (LSTM-FCN)	Karim et al. (2017)	FCN augmented with LSTM-RNN
Multivariate Long Short-Term Memory Fully Convolutional Network (MLSTM-FCN)	Karim et al. (2019)	FCN with squeeze-and-excitation blocks, augmented with LSTM-RNN

Table 4: Overview of the tested models.

3.4.1 ULSTM and MLSTM

The LSTM cell, initially introduced by Hochreiter and Schmidhuber (1997), is praised for its ability to capture long-term dependencies in a sequence while mitigating vanishing gradient issues. The LSTM cell has two components to its state that makes it stand out from the basic RNN cell. Namely, the hidden state, which provides short-term memory, and the cell state, which corresponds to the long-term memory. In addition, a gating scheme that consists of input, forget and output gates is introduced. The input and forget gate jointly determine the amount of the historical information that is kept in the current cell state and how much of the current context is propagated to the next time steps.

In this study, the LSTM will be implemented as proposed by Hewamalage et al. (2021), with a stacked architecture combined with LSTM cells. First, this architecture will be

implemented such that it is an univariate model of lap times. Subsequently, it will be implemented as a multivariate model, by additionally feeding it the exogenous features.

3.4.1.1 Hyper Parameters for ULSTM and MLSTM

The hyper parameters for the LSTM models were carefully selected based on the work by Hewamalage et al. (2021), adapted to this specific forecasting problem. The motivation for all parameter choices are outlined below.

Regarding the optimiser, the Adam optimiser was selected with the initial learning rate set to 0.001, as Hewamalage et al. (2021) found that the Adam optimizer usually required a small range (0.001-0.1) to converge. Concerning the number of hidden layers in the RNN, Hewamalage et al. (2021) strengthen the existing convention that a low value usually performs better. Therefore, in this work two layers with both 64 cells were used. With regard to the activation functions, Hewamalage et al. (2021) used the sigmoid activation function for the hidden layers. However, in this study it was decided to select the Rectified Linear Unit (ReLU) activation function for the hidden layers, because sigmoid and tanh functions are more likely to cause vanishing gradient problems (Sharma et al., 2020). Sharma et al. (2020) conclude in their review on activation functions in neural networks that ReLU is usually the preferred choice as activation function in hidden layers. Regarding the activation of the output, the linear activation function was considered to be the appropriate choice. Namely, predicting lap times is an application of a regression problem, which requires a linear activation function. An overview of all initial and tuned hyper parameters for the LSTM models can be found in Appendix D, Table 5.

3.4.1.1.1 Hyper Parameter Tuning for ULSTM

The ULSTM was first run with the above mentioned initial parameter settings. The hyperparameters were tuned one at a time, while keeping all others constant. According to Hewamalage et al. (2021), the batch size needs to be chosen proportional to the size of the dataset, such that the lower bound of the initial hyperparameter range is around a tenth of the dataset size. Given the fact that this dataset is quite large (136,122 lap times), larger batches of the following sizes were tested: 64, 128, 256. However, it appeared that increasing the batch size reduced performance, such that the initial batch size of 32 leads to the best performance. Additionally, tuning the number of epochs proved that the initial value of 10 epochs yielded the best performance as well. Lastly, experimenting with the number of layers

proved that a model with three layers with 64 cells each performed best as compared to a higher number of layers (4 or more) or the initial setting of two layers.

3.4.1.1.2 Hyper Parameter Tuning for MLSTM

For the MLSTM, the tuned hyperparameters from the ULSTM were used (Appendix D, Table 5). First, the MLSTM was run with all features (see section 3.2 and Appendix B, Table 2). However, inspecting the training and validation loss indicated that the model is overfitting, as the training loss continued to decrease, while the validation loss quickly stopped decreasing. A plausible cause for this overfitting is the high model complexity arising from the high dimensionality. Therefore, the hyperparameter tuning experiments for the MLSTM consisted of combating the overfitting by searching for the optimal number and combination of features.

As outlined earlier, Tulabandhula and Rudin (2014) found that the challenges that are faced in designing a prediction system for professional racing can be overcome only when domain knowledge of racing is integrated in the modeling techniques. They explain that domain knowledge can be practically integrated into the modeling techniques by infusing it into the feature generation and selection. Therefore, it was decided to search for the best combination of features in this study by manually selecting the features by means of domain knowledge of Formula 1. This domain knowledge has been gathered by consistently watching Formula 1 races, and reading commentaries. The search was done manually with multiple combinations of those features that were considered most important based on the domain knowledge of Formula 1 racing: Trackstatus, Position, Lap-type, Tyre Life, and all weather variables. An overview of why these features are important was given in section 3.2. The search proved that an MLSTM with features Track Status and Position resulted in the best performance.

3.4.2 LSTM-FCN and MLSTM-FCN

The LSTM-FCN is composed of a Fully Convolutional Network (FCN) and an LSTM-RNN (Karim et al., 2017). Karim et al. (2017) use TCNs as a feature extractor in a FCN branch. One dimensional filters are applied on the convolutional layers to capture how the input evolves over the course of a time series. The FCN is then augmented with an LSTM-RNN.

Karim et al. (2019) propose to transform the univariate LSTM-FCN into a multivariate time series classification model by expanding the fully convolutional block with

a squeeze-and-excitation block to make the model applicable to multivariate time series. The squeeze and excite mechanism adaptively rescales feature maps, acknowledging that not all features contribute equally to subsequent layers. This can be seen as a form of learned self attention, where the importance of the feature maps is adjusted to model feature interdependencies at each time step (Karim et al., 2019).

3.4.2.1 Hyperparameters for LSTM-FCN and MLSTM-FCN

The hyper parameters for the LSTM-FCN models were carefully selected based on the work by Karim et al. (2017) and Karim et al. (2019), adapted to this specific forecasting problem. The motivation for all parameter choices are outlined below.

Regarding the model architecture, the fully convolutional block is comprised of three stacked temporal convolutional layers accompanied by batch normalization. The convolutional layers have filter sizes of 128, 256, and 128 respectively. The LSTM block is comprised of one LSTM layer with 64 cells, followed by a dropout layer. Karim et al. (2017) propose to use a high dropout rate of 80% to combat overfitting. Concerning the activation functions, ReLU activation function was used for the convolutional layers as well as for the LSTM layer. The concatenated output of the LSTM block and global pooling layer is passed onto a linear activation function in the output layer instead of a softmax activation function used by Karim et al. (2017), considering the fact that they use the LSTM-FCN for time series classification, as opposed to time series forecasting in this study. Both models were trained by means of the Adam optimizer, with an initial learning rate of 0.001. The number of epochs was initially set at 10, following the ULSTM and MLSTM models. Simultaneously, an initial batch size of 32 was used.

The only adjustment that had to be made to the architecture to transform the LSTM-FCN into the MLSTM-FCN, was to conclude the first two convolutional blocks with a squeeze-and-excite block. The only hyperparameter for the squeeze-and-excite block is the reduction ratio, which was set to and kept at 16 following Karim et al. (2019). An overview of all initial and tuned hyper parameters for the LSTM-FCN models can be found in Appendix D, Table 6.

3.4.2.1.1 Hyper Parameter Tuning for LSTM-FCN

The LSTM-FCN was first run with the above mentioned initial parameter settings. The hyperparameters were tuned one at a time, while keeping all others constant. First, as the model showed no signs of overfitting, the high dropout rate of 80% was tuned. The optimal

rate was found to be 20%. With regard to the number of epochs, it was observed that training loss as well as the validation loss continued to decrease while approaching the 10th epoch. Tuning the number of epochs therefore proved that 20 epochs yielded the best performance. Additionally, experiments with larger batch sizes indicated that the initial batch size of 32 led to the best performance. Lastly, tuning the learning rate did not lead to any improvement.

3.4.2.1.2 Hyper Parameter Tuning for MLSTM-FCN

Considering the MLSTM's inability to deal with the high dimensionality incurred from using all features (see section 3.4.1.1.2), it was decided to run the MLSTM-FCN with the same combination of features as in the MLSTM. However, for experimental purposes, an attempt was made to run the MLSTM-FCN with all features. As expected, this model was unable to learn and converge due to the high model complexity.

The MLSTM-FCN was run with the tuned hyperparameters from the LSTM-FCN. Nonetheless, some hyperparameter tuning experiments were conducted to make sure that these hyperparameter settings are optimal for this model as well. As such, the optimal number of LSTM cells proved to be 128, instead of the initial value of 64. However, experiments with the batch size, learning rate, and number of epochs demonstrated that the tuned values for the LSTM-FCN were the optimal values for the MLSTM-FCN as well.

3.5 Evaluation methods

The models were evaluated with three different evaluation metrics to ensure robust model comparison. First, mean squared error (MSE) was chosen, as it ensures that opposite signed errors do not offset each other, which offers an overall view of the forecasting error (Adhikari et al., 2013). Unfortunately, MSE penalises extreme errors. For this reason, mean absolute percentage error (MAPE) was selected as an additional evaluation metric, which represents the percentage of the average absolute error. This metric is especially suitable for predicting lap times as it does not penalise extreme deviations, something which is highly expected to happen in lap time sequences (Adhikari et al., 2013). Lastly, mean absolute error (MAE) is selected in order to gain insights in the magnitude of the overall error (Adhikari et al., 2013).

3.6 Error analysis: high and low complexity races

With the aim of offering a conclusive answer to the research question, the data set has been splitted into two groups; high and low complexity races, where high complexity races

are races in which rainfall was present, and low complexity races are those races in which no rainfall was present. Both the best performing univariate model and multivariate model were used to train and test on these separate groups. In this way, conclusions can be drawn on how univariate and multivariate models compare to each other when dealing with different levels of complexity in the data.

The sequences from a race were added to the high complexity group when rainfall was present in at least one lap during the race. This has resulted in 18 high complexity races and 107 low complexity races. Considering the fact that the high complexity group is substantially smaller than the low complexity group, it was decided to decrease the batch size for the high complexity group. All other hyper parameters except for the number of epochs were the same as in the models that were used for the whole dataset, to ensure unbiased comparison between the groups and models. Similarly, the same evaluation metrics were used.

4. Results

In order to answer this study's research question, the evaluation metrics for the univariate and multivariate models will be compared, which can be found in Table 7. A helpful remark for interpretation of these metrics is that the lap times are measured in milliseconds, where the average lap time in the test set amounts to 90407 milliseconds.

Model Name	Train MSE	Train MAE	Train MAPE	Test MSE	Test MAE	Test MAPE
Baseline	-	-	-	130676431	4774	4,9%
ULSTM	66656808	4223	4,5%	91758211	4684	4,7%
MLSTM	54747748	3774	3.9%	80647734	3958	3.9%
LSTM-FCN	73390136	5353	5,8%	84052706	4005	3,9%
MLSTM-FCN	49250432	3709	3,9%	77825885	4104	4,0%

Table 7: Overview of results for the compared models in the experiment.

4.1 Results ULSTM and MLSTM

The MLSTM was able to outperform the baseline, while the ULSTM performs nearly similarly. In addition, it can be observed that both models perform almost equally on the

training and test set. When comparing the ULSTM and MLSTM, it can be observed that the MLSTM (MSE = 80647734, MAE = 3958, MAPE = 3,9%) performs better than the ULSTM (MSE = 91758211, MAE = 4684, MAPE = 4,7%). This suggests that adding domain specific exogenous variables enables an LSTM to more accurately predict lap times in a Formula 1 race.

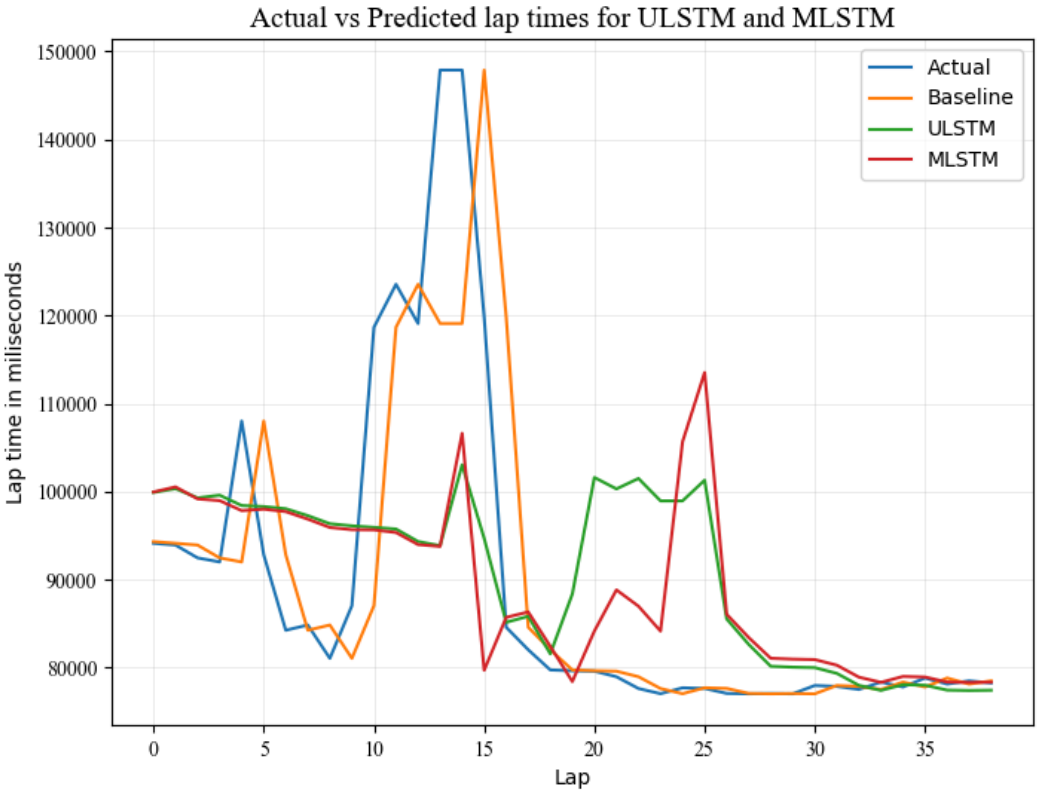

Figure 2: The plot of the actual lap times compared to the predicted lap times by the ULSTM and MLSTM, for one sequence.

Figure 2 shows the plot of the actual lap times compared to the predicted lap times by the ULSTM and MLSTM, for a randomly selected sequence. As mentioned earlier, a sequence in this study can be defined as the sequence of laps for one driver in one race. Figure 3, in Appendix E shows four additional plots from randomly selected sequences. The plots demonstrate that there is a lag in the predicted values, which is approximately of the same size as a window. This suggests that the ULSTM and MLSTM mimic the former seen window as a prediction for the current window. Interestingly, in line with the above observation that the MLSTM performs better than the ULSTM, it can be observed that the MLSTM trends are a more accurate mimicry of the actual trends than the ULTSM trends. However, both models highly underestimate peaks in the lap times. At a first glance, it might seem that the baseline performs better than the models, as it seems to follow the actual trend

better. However, in doing so it tends to make huge errors, as it is always one lap behind. This will particularly produce huge errors in lap time peaks such as in Figure 2 around lap 14.

4.2 Results LSTM-FCN AND MLSTM-FCN

Both the LSTM-FCN and MLSTM-FCN were able to slightly outperform the baseline model. In addition, it is slightly remarkable that the LSTM-FCN performs better on the test set than the training set. When comparing the LSTM-FCN (MSE = 84052706, MAE = 4005, MAPE = 3,9%) and MLSTM-FCN (MSE = 77825885, MAE = 4104, MAPE = 4,0%), it can be observed that there is no strong evidence that any of the two models performs substantially better. Moreover, they do not perform better than the MLSTM. However, the LSTM-FCN does substantially outperform the ULSTM.

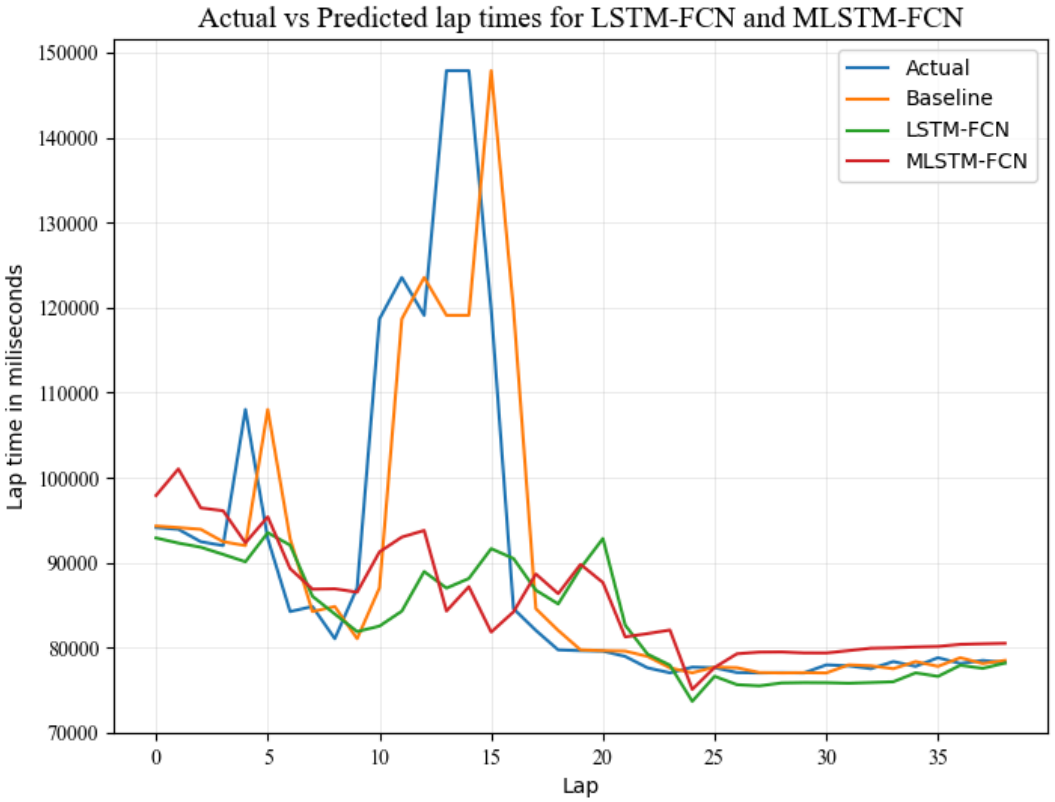

Figure 4: The plot of the actual lap times compared to the predicted lap times by the LSTM-FCM and MLSTM-FCN, for one sequence.

Figure 4 shows the plot of the actual lap times compared to the predicted lap times by the LSTM-FCN and MLSTM-FCN, for the same sequence as in Figure 2. Figure 5, in Appendix E shows four additional plots. The plots do not demonstrate the same apparent prediction lag as the ULSTM and MLSTM did. This suggests that the LSTM-FCN and MLSTM-FCN do not merely mimic the former seen windows as a prediction for the current

window. In line with the results described above, the MLSTM-FCN trends seem to be overall more accurate than the LSTM-FCN trends.

4.3 Error analysis: delving into a sequence

Thus far, the results suggest that all models have difficulties with accurately predicting the lap times, especially when there is a peak in the lap times. An example of such a peak can be found in Figure 2 around laps 4 and 14. Figuring out what conditions cause these peaks in the lap times can uncover the conditions that the models struggle with most. Appendix F, Table 7 offers the explanatory variables per lap for the sequence in Figure 2, resulting from the multivariate data used in this study.

When looking at laps 3 and 4 in Table 7, it can be observed that all features stay stable, except for TrackStatus which goes from 'Track clear' (1) to 'Yellow flag' (2)[1]. Another peak starts around lap 10. It can again be observed that when moving from lap 10 to lap 11, the track status changes from 'Track clear' to 'Yellow flag'. The same goes for the peak around lap 14, as the track status shows the same change moving from lap 13 to 14. These observations suggest that a peak tends to be caused by a change in track status. Thus, a change in track status can be seen as a condition that the models struggle with. When looking at laps 24 and 25 in Table 7, it can be observed that the driver goes for a pitstop, based on the feature 'LapType'. Interestingly, the lap times stay consistent during this event. This suggests that a pit stop is not a direct cause of a peak in lap times, and therefore is not a condition which underlies the models' weak performances.

Model Name	**Train MSE**	**Train MAE**	**Train MAPE**	**Test MSE**	**Test MAE**	**Test MAPE**
LSTM-FCN no rain	67892480	5389	5,8%	55024178	3479	3,4%
MLSTM-FCN no rain	16668492	2179	2,3%	26537436	2575	2,6%
LSTM-FCN rain	121087960	7975	8,6%	78678865	5657	6,2%
MLSTM-FCN rain	37122796	3255	3,3%	56659053	4000	4,3%

Table 8: Overview of results for the error analysis.

[1] The yellow flag is a signal of danger (Seymour, 2023). Additionally, the overview of all feature categories in Appendix B Table 2 is helpful in interpreting these features.

4.4 Results error analysis: high and low complexity races

As outlined in section 3.6, in order to analyse the errors, the data set was divided into high complexity races in which rain was present and low complexity races in which rain was not present. The LSTM-FCN and MLSTM-FCN have been separately trained and tested on these groups. The results can be found in Table 8.

4.4.1 Results group no rain

Again, it appears that the LSTM-FCN performs better on the test data than on the training data. When looking at the no rain group, it can be observed that the MLSTM-FCN (MSE = 26537436, MAE = 2575, MAPE = 2,6%) performs substantially better than the LSTM-FCN (MSE = 55024178, MAE = 3479, MAPE = 3,4%). This finding is interesting, because the results in section 4.2 demonstrated that the MLSTM-FCN does not confidently outperform the LSTM-FCN. This suggests that there is greater difference between the LSTM-FCN and MLSTM-FCN in favor of the latter when used on specific groups in the data, which is in this case less complex data.

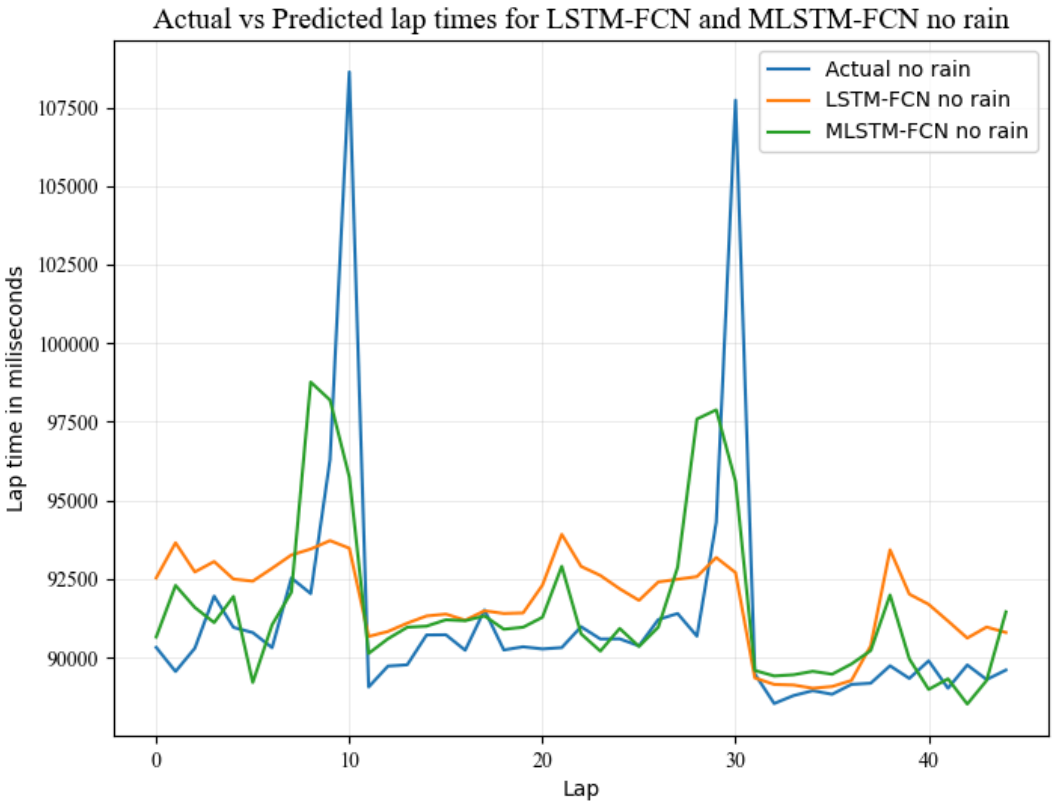

Figure 6: The plot of the actual lap times compared to the predicted lap times by the LSTM-FCM and MLSTM-FCN, for one sequence in the no rain group.

Figure 6 shows the plot of the actual lap times compared to the predicted lap times by the LSTM-FCN and MLSTM-FCN, for one sequence in the no rain group. Figure 7, in Appendix E shows four additional plots. Interestingly, in line with the above mentioned result, it can be observed in the plots that the MLSTM-FCN trends follow the actual trends more accurately than the LSTM-FCN trends do. This finding strengthens the indication that the MLSTM-FCN can outperform the LSTM-FCN under some conditions. However, it is still apparent that the models have difficulties with estimating peaks accurately.

4.4.2 Results group rain

First of all, it can be observed that there are some differences in the performances of the models for the training and test sets, however these differences are not outstanding. When looking at the rain group, it can again be observed that the MLSTM-FCN (MSE = 56659053, MAE = 4000, MAPE = 4,3%) performs better than the LSTM-FCN (MSE = 78678865, MAE = 5657, MAPE = 6,2%). This strengthens the conviction that the MLSTM-FCN outperforms the LSTM-FCN. When comparing the results for the rain group to the no rain group, it is proved that the models have more difficulties with predicting more complex sequences. However, in any situation, the MLSTM-FCN is superior to its univariate counterpart.

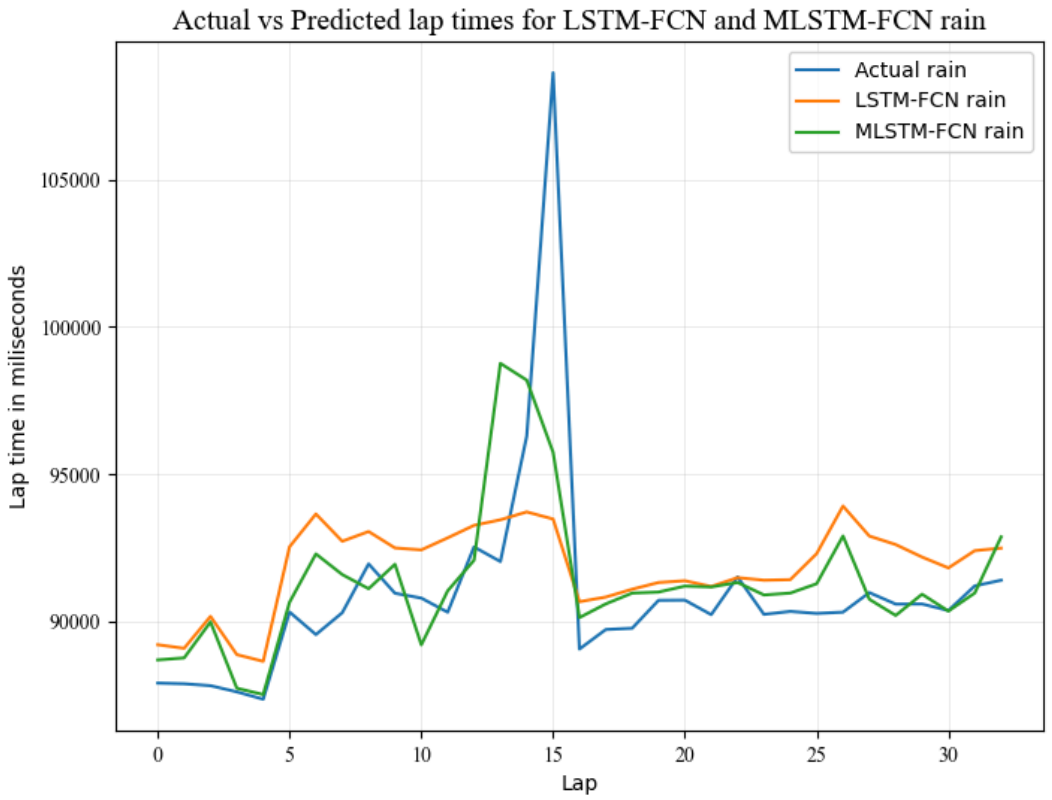

Figure 8: The plot of the actual lap times compared to the predicted lap times by the LSTM-FCM and MLSTM-FCN, for one sequence in the rain group.

Figure 8 shows the plot of the actual lap times compared to the predicted lap times by the LSTM-FCN and MLSTM-FCN, for one sequence in the rain group. Figure 9, in Appendix E shows four additional plots. The plots demonstrate that the MLSTM-FCN trends follow the actual trends more accurately than the LSTM-FCN trends do. Once again, this strengthens the support of choosing the multivariate model over the univariate model.

5. Discussion

The research goal of this study was to determine how multivariate and univariate times series models deal with complex forecasting scenarios and specifically how they compare to each other in predicting lap times in a Formula 1 race.

5.1 Discussion of results

A first interesting finding is that the MLSTM performed better than the ULSTM, which means that in predicting lap times in a Formula 1 race it is beneficial to add domain specific exogenous features in addition to the lap times in an LSTM. This finding is in line with existing work on the comparison between univariate and multivariate RNNs (Zhang et al., 2004; Cao et al., 2012).

However, when adding those features it is required that one is critical in choosing them, as the MLSTM suffered from high model complexity and dimensionality when run with the whole set of initially selected features. This instance of the curse of dimensionality is not surprising, as this a broad problem in data analysis and in time series forecasting (Verleysen et al., 2005). In this study, the features for the multivariate models were manually selected by using domain knowledge in the choice for the features, based on the work of Tulabandhula and Rudin (2014).

When comparing the LSTM-FCN and MLSTM-FCN it was initially found that the difference in performance was not large enough to strongly prove that the MLSTM-FCN leads to more accurate predictions. However, it was found that even though the LSTM-FCN models did not outperform the MLSTM, the LSTM-FCN did perform substantially better than the ULSTM. This finding suggests that a univariate forecasting scenario benefits from integrated feature extraction mechanisms in a model, while in this study, such a large benefit was not found in a multivariate forecasting scenario. One potential explanation for this contradicting observation is that a simple univariate model needs some facilitated feature extraction mechanisms, as time series patterns are hidden in a single value. While these

mechanisms are less needed in multivariate models as these contain exogenous variables that already represent and explain some of the underlying patterns in the time series.

In the error analysis, the MLSTM-FCN outperformed the LSTM-FCN in both the low and high complexity group. This is an interesting finding when compared to the discussion above, because for some reason the MLSTM-FCN is only superior to its univariate counterpart in this study when used on separate groups. One potential explanation for this is that a division in distinct groups based on some criterion removes one major source of noise in the data. This removal of noise enables the squeeze-and-excite block to better model the interdependencies between the variables in the MLSTM-FCN. This finding and reasoning offer an interesting perspective on the work of Karim et al. (2017) and Karim et al. (2019).

In order to provide more interpretability in the model errors, an effort was made to uncover the conditions that cause the models to struggle with accurately predicting the lap times. It was found that while the models perform reasonably when the lap times are stable and external factors are minimal, none of the models are capable of accurately predicting peaks in the lap times that are caused by external factors, such as on track incidents. This finding confirms that predictive analysis for professional racing is challenging, due to many external factors that impact a race, which was contended by Tulabandhula and Rudin (2014).

5.2 Limitations

The primary limitation of this study is the limited consideration with respect to inconsistencies in the data set. Namely, Formula 1 regulations change slightly every year, and every few years there is even a large change in regulations. This implies for example that there were different categories of tyres in 2018 as compared to 2023. This study used all sequences from all races, and did not take into account any inconsistencies that may exist between them. This may have led to decreased reliability of the data.

What is more, this study did not experiment with different window sizes and strategies due to limited resources. The window sizes were held constant at ten, which is relatively short. Experimenting with larger windows could have potentially enabled the models to better capture long-term dependencies. In addition, more exploration could have been done on different types of windows, like for example expanding windows. These restrictions reduce the robustness of the study and limit its generalizability.

Lastly, although this study did analyze the model errors and behavior which has provided a certain extent of model interpretability, it would have been interesting to conduct

this analysis for more sequences. Unfortunately, this was not done due to resource constraints.

5.3 Relevance and future research

For the broader domain of time series forecasting, this study has found evidence that complex forecasting scenarios benefit from multivariate rather than univariate models, which was already proposed by Hewamalage et al. (2021). This insight has the potential to improve other complex forecasting scenarios in other sectors that benefit broader society. Future research could gain more insights into how univariate and multivariate models compare to each other when dealing with different levels of complexity in the data by comparing model performance on complex forecasting scenarios and less complex forecasting scenarios.

Furthermore, this study has demonstrated that without automated techniques and methods for feature selection, the features can be selected manually by means of domain knowledge of the complex forecasting scenario at hand. This study has proved that this selection method offers a robust and effective starting point in the case of constraints of resources. This finding has significant societal relevance, as it implies that even in resource-limited settings, such as small businesses or underfunded research initiatives, effective forecasting of complex scenarios can still be achieved.

Despite infusing domain knowledge of racing into the modeling techniques based on Tulabandhula and Rudin (2014), this study did not succeed in overcoming the challenges in predictive analysis for professional racing, as the models are still not able to cope with the external factors that impact a Formula 1 race. In this light, this study sees opportunities in predicting lap times in Formula 1 by choosing the set of features race dependent. This means that different multivariate models with different selections of features could be trained for distinct types of races. For example, track status may be the most important feature in a street race, as these races are known to have an increased risk of incidents. By creating these race specific models, the domain knowledge of racing is even more integrated into the modeling techniques, which hopefully allows the models to better cope with external factors.

However, there are other opportunities for incorporating domain knowledge into the modeling techniques which can facilitate the models' ability to cope with these conditions. Namely, in order to tackle the forecasting of complex time series, Armstrong et al. (2005) propose decomposition of complex time series by causal forces. Domain knowledge is used to identify different forces for component series, after which forecasts are obtained for each

component instead of for the global series. It would be interesting to apply this technique on Formula 1 races and this specific dataset.

Finally, future work could focus on the mixed results regarding the LSTM-FCN and MLSTM-FCN. One suggestion that was offered in this study stated that multivariate models might benefit less from feature extraction mechanisms because the exogenous features already explain and represent the underlying patterns in the time series. Another suggestion offered in this study was that an MLSTM-FCN is only able to outperform an LSTM-FCN when the noise in the data is reduced. Future studies could dive into these reasoning and empirically test them.

6. Conclusion

This study contributes to the field of time series forecasting and to the sub-domain of comparing univariate and multivariate time series models in complex forecasting scenarios like Formula 1 racing by answering the following research question:

How do multivariate time series models compare to univariate time series models in predicting lap times in a Formula 1 race?

This study has demonstrated that multivariate models perform better on a complex forecasting scenario than univariate models. Feature selection for the multivariate models was done by manually selecting the features using domain knowledge. Despite its minor drawbacks, feature selection by means of domain knowledge proves to be efficient and offers further opportunities in the field of multivariate time series models. Furthermore, potential solutions were given for the models' limited ability to deal with the external factors associated with complex forecasting scenarios. Finally, this study has offered valuable insights in the domain of feature extraction of patterns within time series, and offers some interesting reasoning and recommendations that can be further examined in future studies.

7. Source/code/ethics/technology statement

The data has been acquired from the FastF1 API through the FastF1 python package. Work on this thesis did not involve collecting data from human participants or animals. The original owner of the data and code used in this thesis retains ownership of the data and code during and after the completion of this thesis. The thesis code can be accessed through the following GitHub repository: https://github.com/fbrusik/MasterThesis_DS_FleurBrusik.

The code used for the LSTM-FCN model used the implementation proposed by Karim et al. (2017) acquired from a GitHub repository https://github.com/titu1994/LSTM-FCN. The code used for the LSTM-FCN model used the implementation proposed by Karim et al. (2019) acquired from a GitHub repository https://github.com/titu1994/MLSTM-FCN. All figures and tables in this study are made by the author.

References

Ahn, H., Sun, K., & Kim, K. P. (2022). Comparison of missing data imputation methods in time series forecasting. *Computers, Materials & Continua, 70*(1), 767-779. https://doi.org/10.32604/cmc.2022.019369

Allender, M. (2008). Predicting the outcome of NASCAR races: The role of driver experience. *Journal of Business & Economics Research (JBER)*, 6(3). https://doi.org/10.19030/jber.v6i3.2403

Armstrong, J. S., Collopy, F., & Yokum, J. T. (2005). Decomposition by causal forces: a procedure for forecasting complex time series. International Journal of forecasting, 21(1), 25-36. https://doi.org/10.1016/j.ijforecast.2004.05.001

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint*. https://doi.org/10.48550/arXiv.1803.01271

Bandara, K., Bergmeir, C., & Smyl, S. (2020). Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert systems with applications, 140*, 112896. https://doi.org/10.1016/j.eswa.2019.112896

Cao, Q., Ewing, B. T., & Thompson, M. A. (2012). Forecasting wind speed with recurrent neural networks. *European Journal of Operational Research, 221*(1), 148-154. https://doi.org/10.1016/j.ejor.2012.02.042

Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., & Batista, G. (2015). *The UCR Time Series Classification Archive.* Retrieved from www.cs.ucr.edu/~eamonn/time_series_data/

FastF1. (2024). https://docs.fastf1.dev/

Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting, 37*(1), 388-427. https://doi.org/10.1016/j.ijforecast.2020.06.008

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Karim, F., Majumdar, S., Darabi, H., & Chen, S. (2017). LSTM fully convolutional networks for time series classification. *IEEE access*, 6, 1662-1669. https://doi.org/10.1109/access.2017.2779939

Karim, F., Majumdar, S., Darabi, H., & Harford, S. (2019). Multivariate LSTM-FCNs for time series classification. *Neural networks,* 116, 237-245. https://doi.org/10.1016/j.neunet.2019.04.014

Keras. (n.d.). *Keras: The Python deep learning API*. Retrieved from https://keras.io/

Pei, W., Dibeklioğlu, H., Tax, D. M., & van der Maaten, L. (2017). Multivariate time-series classification using the hidden-unit logistic model. *IEEE transactions on neural networks and learning systems, 29* (4), 920-931. https://doi.org/10.1109/tnnls.2017.2651018

Pirelli. (n.d.). *F1 Tyres.* https://www.pirelli.com/tyres/en-ww/motorsport/f1/tyres.

Pratama, I., Permanasari, A. E., Ardiyanto, I., & Indrayani, R. (2016, October). A review of missing values handling methods on time-series data. *In 2016 international conference on information technology systems and innovation (ICITSI),* 1-6. IEEE. https://doi.org/10.1109/icitsi.2016.7858189

Schäfer, P., & Leser, U. (2017). Multivariate time series classification with WEASEL+ MUSE. *arXiv preprint arXiv*. https://doi.org/10.48550/arXiv.1711.11343

Seymour, M. (2023). The beginner's guide to Formula 1 flags. *Formula 1*. Retrieved from: https://www.formula1.com/en/latest/article/the-beginners-guide-to-formula-1-flags.T5DqOqbWI6S4Va8Y5yMld

Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. *International Journal of Engineering Applied Sciences and Technology 4*(12), 310-316.

TensorFlow. (n.d.) *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Retrieved from https://www.tensorflow.org/

Tulabandhula, T., & Rudin, C. (2014). Tire changes, fresh air, and yellow flags: challenges in predictive analytics for professional racing. *Big data*, *2*(2), 97-112. http://dx.doi.org/10.1089/big.2014.0018

Verleysen, M., & François, D. (2005). The Curse of Dimensionality in Data Mining and Time Series Prediction. *In Computational Intelligence and Bioinspired Systems,* 758–770. Springer Berlin Heidelberg. https://doi.org/10.1007/11494669_93

Wang, Z., Yan, W., & Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. *International joint conference on neural networks (IJCNN),* 1578-1585. IEEE. https://doi.org/10.1109/ijcnn.2017.7966039

Zhang, W., Cao, Q., & Schniederjans, M. J. (2004). Neural network earnings per share forecasting models: A comparative analysis of alternative methods. *Decision Sciences, 35*(2), 205-237. https://doi.org/10.1111/j.00117315.2004.02674.x

Appendix A - Literature

Origin	Model	Datasets	Performance
Hewamalage et al. (2021)	LSTM	Several datasets from the following forecasting competitions: • CIF 2016 Forecasting Competition Dataset • NN5 Forecasting Competition Dataset • M3 Forecasting Competition Dataset • M4 Forecasting Competition Dataset • Wikipedia Web Traffic Time Series Forecasting Competition Dataset • Tourism Forecasting Competition Dataset	The model outperforms the benchmark techniques on all the datasets except for the M4 monthly dataset, in terms of both the SMAPE and MASE error metrics.
Karim et al. (2017)	LSTM-FCN	The model has been tested on all 85 UCR time series datasets (Chen et al., 2015).	Model evaluation was done by means of accuracy, arithmetic rank, geometric rank, the Wilcoxon signed-rank test, and mean per class error. The LSTM-FCN

			overall achieves state-of-the-art performance, and it is able to outperform state-of-the art models in at least 43 datasets.
Karim et al. (2019)	MLSTM-FCN	The model has been tested on 35 benchmark datasets from various fields, of which five were formerly used by Pei et al. (2018), and 20 were used by Schäfer et al. (2017).	Model evaluation was done by means of accu-racy, arithmetic rank, geometric rank, the Wilcoxon signed-rank test, and mean per class error. The MLSTM-FCN attains state-of-the-art results for most of the datasets, namely 28 out of 35.
Wang et al. (2017)	FCN	The model has been tested on a subset of the UCR time series datasets (Chen et al., 2015), which includes 44 distinct time series datasets.	The FCN achieves premium performance to other state-of-the-art approaches.

Bai et al. (2018)	LSTM, GRU, RNN, TCN	The models are evaluated on tasks that have been commonly used to benchmark the performance of different RNN sequence modeling architectures: • The adding problem • Sequential MNIST and P-MNIST • Copy memory • JSB Chorales and Nottingham • PennTreebank • Wikitext-103 • LAMBADA • text8	A simple convolutional architecture outperforms canonical recurrent networks such as LSTMs across a diverse range of tasks and datasets

Table 1: Details on the used most important literature works.

Appendix B - Features

Feature Name	Feature Type	Categories
Laptime	Numerical	-
Compound	Categorical	Hard, Hypersoft, Intermediate, Medium, Soft, Supersoft, Ultrasoft, Wet
Tyre Life	Numerical	-
Stint	Numerical	-
Lap-type	Categorical	Inlap, Outlap, Lap
Driver	Categorical	AIT, ALB, ALO, BOT, DEV, ERI, FIT, GAS, GIO, GRO, HAM, HAR, HUL, KUB, KVY, LAT, LAW, LEC, MAG, MAZ, MSC, NOR, OCO, PER, PIA, RAI, RIC, RUS, SAI, SAR, SIR, STR, TSU, VAN, VER, VET, ZHO
Team	Categorical	Alfa Romeo, Alfa Romeo Racing, AlphaTauri, Alpine, Aston Martin, Ferrari, Force India, Haas F1 Team, McLaren, Mercedes, Racing Point, Red Bull Racing, Renault, Sauber, Toro Rosso, Williams
Trackstatus	Categorical	‘1’: Track clear, ‘2’: Yellow flag, ‘4’: Safety Car, ‘5’: Red Flag, ‘6’: Virtual Safety Car deployed, ‘7’: Virtual Safety Car ending
Position	Numerical	-
Weather - Air Temperature	Numerical	-

Weather - Humidity	Numerical	-
Weather - Pressure	Numerical	-
Weather - Rainfall	Categorical	True, False
Weather - Track Temperature	Numerical	-
Weather - Wind Direction	Numerical	-
Weather - Wind Speed	Numerical	-

Table 2: An overview of all features for the multivariate models.

Appendix C - Missing values in the data

Feature Name	Missing Sum - Including First laps	Missing Sum - Excluding First laps
Laptime	2646	2533
Stint	343	18
Compound	343	18
Tyre Life	343	18
Trackstatus	177	177
Position	177	177
Total Missing Values	4029	2941
Percentage Missing Dataset	2.95%	2.16%

Table 3: An overview of the number of missing values per variable in the dataset. Shown before and after the deletion of all first laps. The features that had no missing values were excluded from this table.

Appendix D - Hyper parameter settings

Hyper Parameters	Initial Value	Tuned Value
Optimizer	Adam	Adam
Learning rate	0.001	0.001
Batch size	32	32
Hidden Layers	Two layers with 64 cells	Three layers with 64 cells
Activation Function - Hidden State	ReLU	ReLU
Activation Function - Output	Linear	Linear
Loss Function	MSE	MSE
Epochs	10	10

Table 5: Overview of the initial and tuned hyper parameter settings for the LSTM models.

Hyper Parameters	Initial Value	Tuned Value: LSTM-FCN	Tuned Value: MLSTM-FCN
Optimizer	Adam	Adam	Adam
Learning rate	0.001	0.001	0.001
Batch size	32	32	32
Convolutional Layers	Three layers with filter sizes of 128, 256, and 128	Three layers with filter sizes of 128, 256, and 128	Three layers with filter sizes of 128, 256, and 128
LSTM Layer	One layer with 64 cells	One layer with 64 cells	One layer with 128 cells
Dropout rate	80	20	20
Reduction rate	-	-	16
Activation Function - Layers	ReLU	ReLU	ReLU
Activation Function - Output	Linear	Linear	Linear
Loss Function	MSE	MSE	MSE
Epochs	10	20	20

Table 6: Overview of the initial and tuned hyper parameter settings for the LSTM-FCN models.

Appendix E - Actual versus. Predicted plots

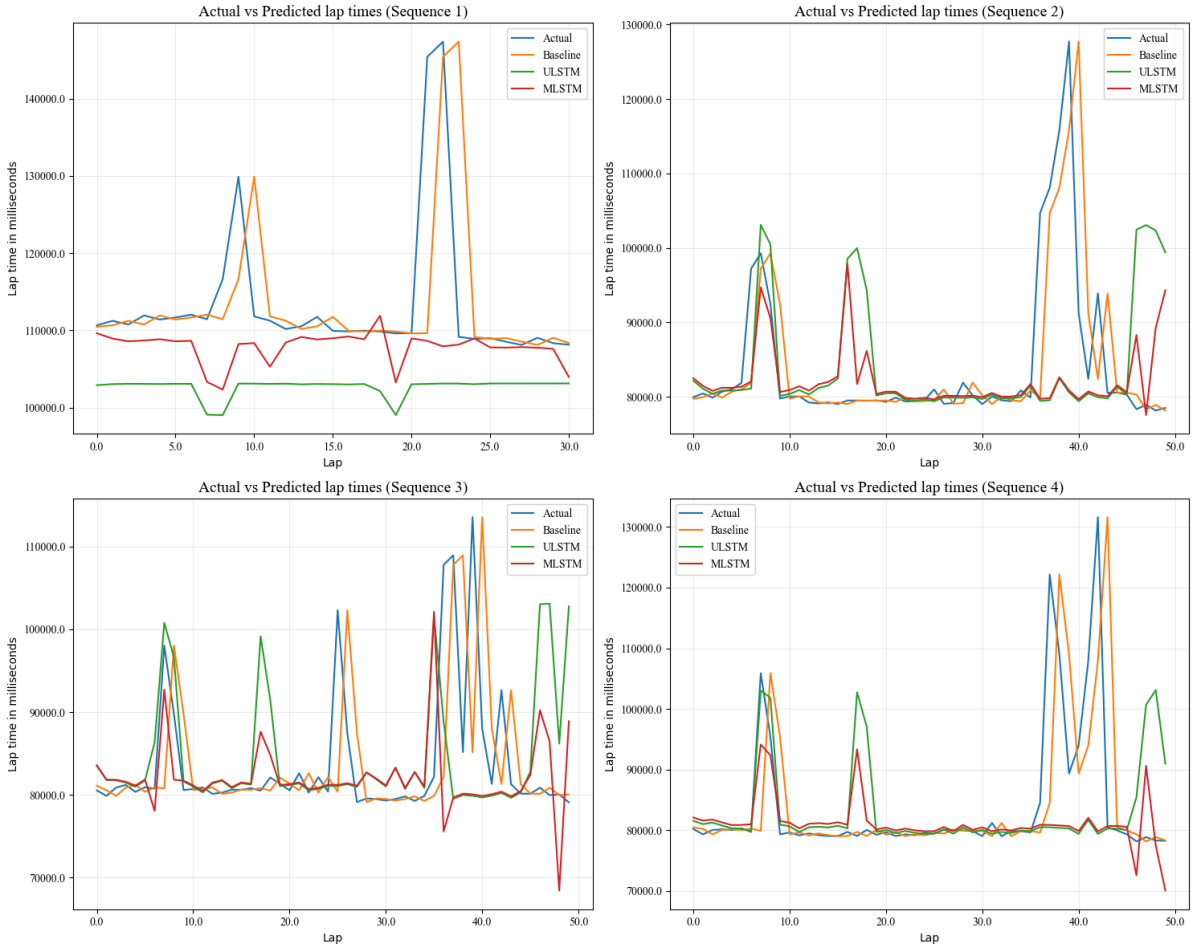

Figure 3: Plots of the actual lap times compared to the predicted lap times by the ULSTM and MLSTM, for four sequences.

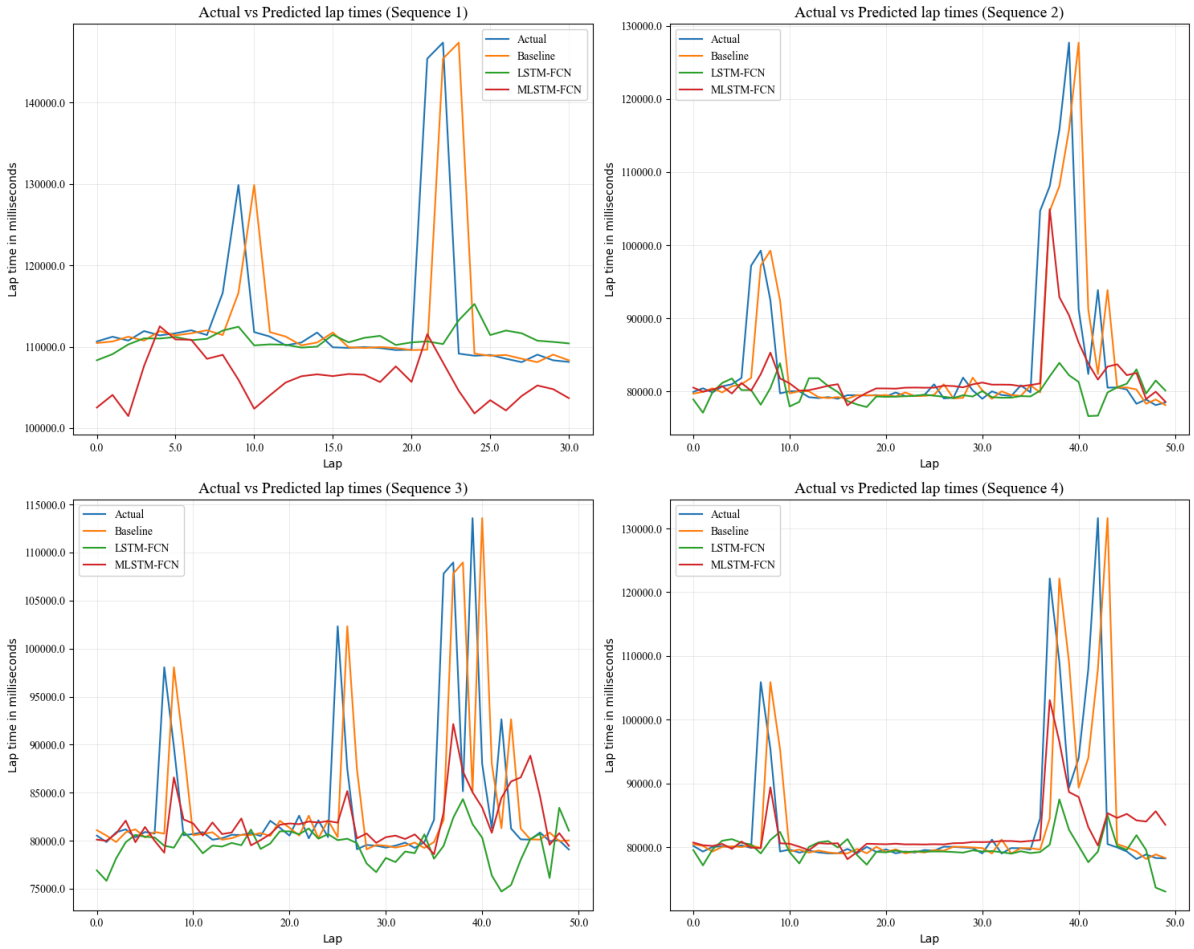

Figure 5: Plots of the actual lap times compared to the predicted lap times by the LSTM-FCN and MLSTM-FCN, for four sequences.

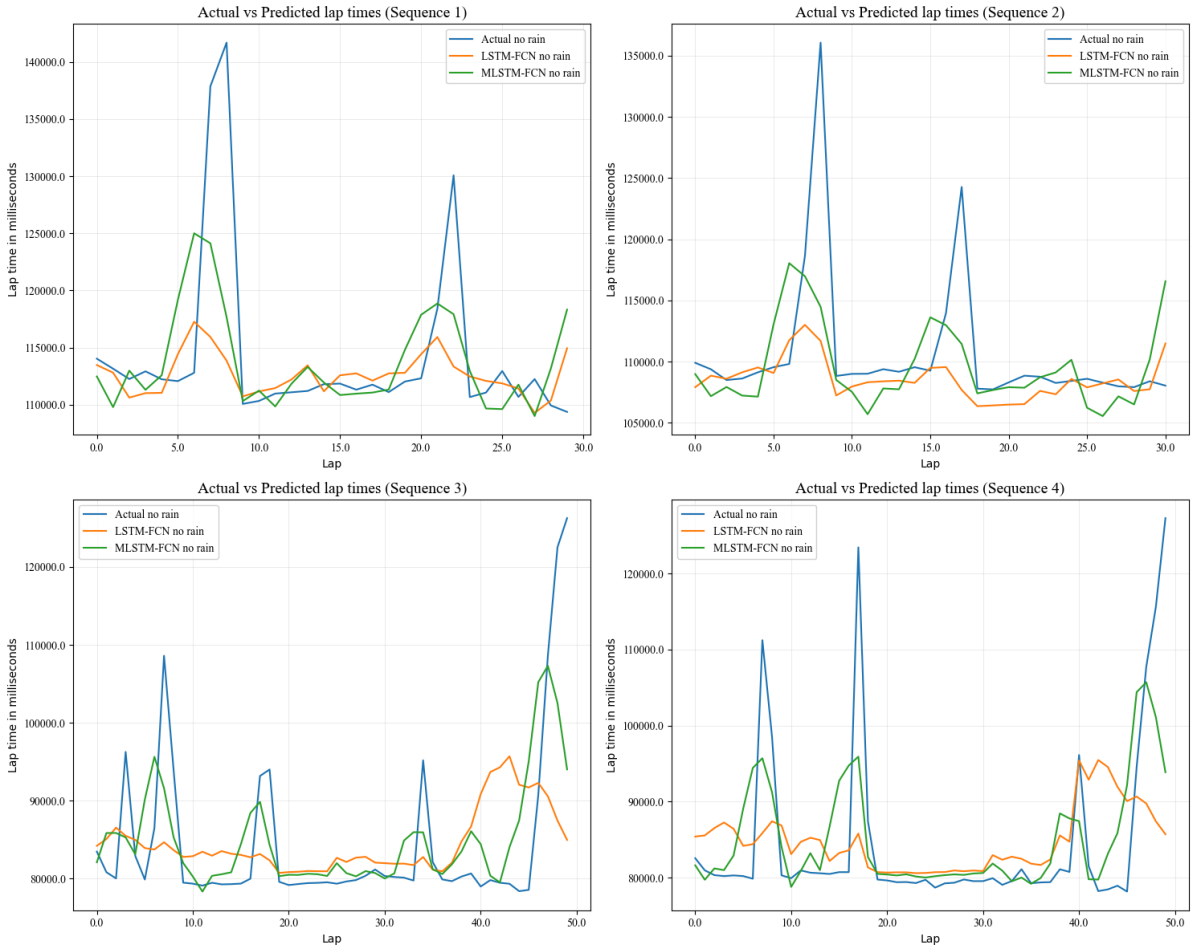

Figure 7: Plots of the actual lap times compared to the predicted lap times by the LSTM-FCN and MLSTM-FCN, for four sequences in the no rain group.

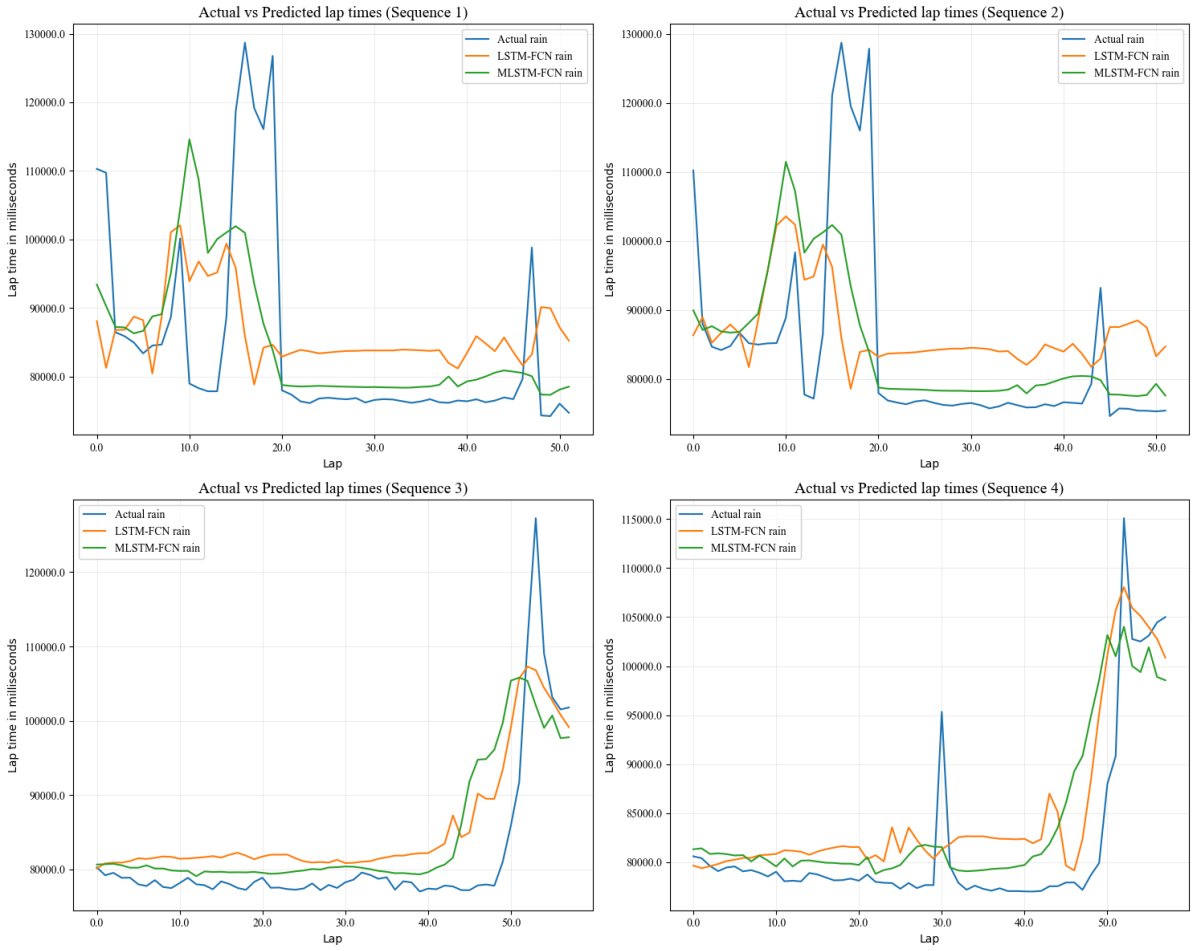

Figure 9: Plots of the actual lap times compared to the predicted lap times by the LSTM-FCN and MLSTM-FCN, for four sequences in the rain group.

Appendix F - Error analysis of a sequence

Lap	Driver	Stint	Compound	TyreLife	TrackStatus	Position	Rainfall	LapType
1	RUS	2.0	HARD	7.0	1.0	6.0	TRUE	Lap
2	RUS	2.0	HARD	8.0	1.0	6.0	TRUE	Lap
3	RUS	2.0	HARD	9.0	1.0	6.0	TRUE	Lap
4	RUS	2.0	HARD	10.0	2.0	6.0	TRUE	Lap
5	RUS	2.0	HARD	11.0	2.0	6.0	TRUE	Lap
6	RUS	2.0	HARD	12.0	1.0	6.0	TRUE	Lap
7	RUS	2.0	HARD	13.0	1.0	6.0	TRUE	Lap
8	RUS	2.0	HARD	14.0	1.0	6.0	TRUE	Lap
9	RUS	2.0	HARD	15.0	1.0	6.0	TRUE	Lap
10	RUS	2.0	HARD	16.0	1.0	6.0	TRUE	Lap
11	RUS	2.0	HARD	17.0	2.0	5.0	TRUE	Lap
12	RUS	2.0	HARD	18.0	1.0	5.0	TRUE	Lap
13	RUS	2.0	HARD	19.0	1.0	5.0	TRUE	Lap
14	RUS	2.0	HARD	20.0	2.0	5.0	TRUE	Lap
15	RUS	2.0	HARD	21.0	1.0	6.0	TRUE	InLap
16	RUS	2.0	HARD	22.0	1.0	6.0	TRUE	Outlap
17	RUS	2.0	HARD	23.0	1.0	5.0	TRUE	Lap
18	RUS	2.0	HARD	24.0	1.0	5.0	TRUE	Lap
19	RUS	2.0	HARD	25.0	1.0	5.0	TRUE	Lap
20	RUS	2.0	HARD	26.0	26.0	5.0	TRUE	Lap
21	RUS	2.0	HARD	27.0	64.0	5.0	TRUE	Lap
22	RUS	2.0	HARD	28.0	4.0	5.0	TRUE	Lap
23	RUS	2.0	HARD	29.0	4.0	5.0	TRUE	Lap
24	RUS	2.0	HARD	30.0	45.0	5.0	TRUE	InLap

25	RUS	3.0	MEDIUM	1.0	1.0	5.0	FALSE	Outlap
26	RUS	3.0	MEDIUM	2.0	1.0	5.0	FALSE	Lap
27	RUS	3.0	MEDIUM	3.0	1.0	5.0	FALSE	Lap
28	RUS	3.0	MEDIUM	4.0	1.0	5.0	FALSE	Lap
29	RUS	3.0	MEDIUM	5.0	1.0	5.0	FALSE	Lap
30	RUS	3.0	MEDIUM	6.0	1.0	5.0	FALSE	Lap
31	RUS	3.0	MEDIUM	7.0	1.0	5.0	FALSE	Lap
32	RUS	3.0	MEDIUM	8.0	1.0	5.0	TRUE	Lap
33	RUS	3.0	MEDIUM	9.0	1.0	5.0	FALSE	Lap
34	RUS	3.0	MEDIUM	10.0	1.0	5.0	FALSE	Lap
35	RUS	3.0	MEDIUM	11.0	1.0	5.0	FALSE	Lap
36	RUS	3.0	MEDIUM	12.0	1.0	5.0	FALSE	Lap
37	RUS	3.0	MEDIUM	13.0	1.0	5.0	FALSE	Lap
38	RUS	3.0	MEDIUM	14.0	1.0	5.0	FALSE	Lap
39	RUS	3.0	MEDIUM	15.0	1.0	5.0	FALSE	Lap

Table 7: Error analysis; the explanatory variables for the sequence in Figures 2 and 4.