2020-11-15

Deep Learning in the Maintenance Industry

Paulo Cesar Ribeiro Silva
Technological University Dublin

Recommended Citation

Silva, P. (2020). *Deep learning in the maintenance industry*. Masters dissertation. Technological University Dublin. doi: 10.21427/q6yr-et27

Deep learning in the maintenance industry

Workload forecasting and task duration classification

Paulo Cesar Ribeiro Silva

Supervisor: Dr. Fernando Perez-Tellez

Dr. John Cardiff

TU Dublin

This dissertation is submitted for the degree of

Masters by Research

November 2020

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other University.

Paulo Cesar Ribeiro Silva
November 2020

Acknowledgements

I would like to acknowledge Neil from Civic and the computing department at TU Dublin Tallaght, in particularly Barry for bringing this project to light. I would also like to thank my supervisors Fernando and John for their help and guidance during the whole process. And I'm forever grateful to my partner Anabela, for your patience, understanding and support throughout the entire development of this project.

Abstract

The work hereby presented was born from a desire to find how modern machine learning methods can be applied to improve scheduling in the maintenance industry.

Efficient scheduling is a daily challenge in the maintenance industry. This industry deals with planned maintenance and daily requests for repairs. Commercial building maintenance work is regularly needed due to deterioration, accidents on the premises or legal requirements for building compliance. Efficient scheduling is a daily challenge in this industry. The scheduling must account for variable workload, variable workforce and variable task duration. In our research, we leverage recent developments in Machine Learning to provide additional information that can aid human operators to make informed decisions when scheduling. We take the approach of using AI to improve human lives instead of replacing human workers. To this end, we propose two methods:

- Forecasting workload.
- Task classification into short/long duration based on the task's text description.

Solving these tasks would aid the human operator to create a more efficient work plan, optimizing human resources and reducing overwork due to poor scheduling. Our results show that we were able to train skilful models for forecasting workload, which greatly improved on the baselines. We employ recent innovations in Natural Language Processing such as word vectors and attention mechanisms to achieve notable results in classifying tasks, creating models capable to outperform human performance at the task.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

This study was motivated by a collaboration between the industry and the academic community. Creating an efficient schedule is a challenge in the maintenance industry. The industry has a variable daily workload due to daily requests for repairs from customers. The workforce capacity is variable as well, with the company needing to ensure that appropriate human resources are allocated each day.

Each maintenance engineer works on a variable number of tasks each day and his remuneration is based on the total work hours completed. The work may involve hazards (i.e. heights, machinery, electricity), so each engineer needs to be allocated enough time to assess risks and complete the task safely.

Planning and allocating tasks correctly is of major importance for efficiency and safety. Making sure that employees have an appropriate number of tasks for each day and enough time for each task benefits both the organization and its employees.

In this study, we approach the challenge of helping human operators to make more informed decisions when scheduling. We take the approach of using recent developments in artificial intelligence and deep learning to build models that collaborate with the human operator. Instead of trying to replace human decision making, we aim to use artificial intelligence to inform human decision making.

1.1 Background

The maintenance industry operates the maintenance of commercial facilities. It provides services of repairs and maintenance of various nature. Common maintenance tasks can be from different areas such as electrical, plumbing, mechanical, drainage, fabric. To complete the maintenance work the company need employees with a diverse skill set.

The customer can request these services in a planned maintenance contract, quoted work, reactive tasks and emergency tasks. In a planned maintenance contract the work is scheduled to take place on a regular basis. This contract agrees for a number of monthly visits to the facilities for regular maintenance. Quoted work refers to work requested by customer initiative or engineer advice. This work can pertain repairs or renovations that require budgeting and customer approval. Reactive and Emergency work is the classification given when a repair of some urgency is required (ie. lights are out). Emergency tasks are reactive tasks that require immediate attention and take precedence over others.

The general workflow of this industry consists of an operations team based on an office and a mobile workforce of engineers and subcontractors of flexible size.
The operations team is responsible for the creation and assignment of tasks. The engineer's team is responsible for travelling to each store to execute the repairs.

1.2 Research questions

The purpose of this study is to provide information that can aid the decision-making of operations teams in charge of scheduling.

When it comes to scheduling tasks the human operator must consider a variety of aspects both objective and subjective. The maintenance engineer must have the right skill set for the task, be within reasonable travel distance, be available. The customer may take priority due to history with the company or a task requires urgency. The customer can have a preference for a particular person due to familiarity or personality traits. The maintenance engineer may be particularly skilled at the task required, have the necessary materials at hand, or a particularly friendly demeanour that helps in stressful situations.

Due to the need for a human touch for this job, a lot of subjective decisions and a lot of information that is hard to capture, an automatic scheduling system would be a poor substitute for it. We tackled the research from the point of view of aiding the human operators, not replacing them with automatic scheduling. To this end, we identified two objectives that would aid the human operator on his scheduling.

- Providing a forecast for workload in the following day, so that appropriate resources are allocated.
- Identify and group tasks that take less time to complete and longer to complete as to facilitate scheduling.

Based on these objectives we formulated our research questions. The aim of this study is to answer the research questions:

How can historical data be used to forecast the workload for the next day?

How can a task text description be used to identify short tasks and long tasks?

These questions present a set of challenges. There is a natural random behaviour when it comes to human activity. A text description of a task is often ambiguous, since it is based on natural language and at times incomplete instructions. Predicting the duration based on a text description will likely not be completely deterministic. The description of a task is not a precise account of every step needed to complete a task. Complications can arise during the execution of a task. And humans are not precise machines that perform the same task in the same time. The same task can take more or less time depending on the person performing it or even the same person at different times.

1.3 Significance

Forecasting is a task that helps organizations with resource allocation, improving efficiency and setting goals. Producing accurate forecasts is a challenge even for personnel with years of experience working in the company. Predicting the future is not an easy task. Customer's demands vary daily and unexpected accidents will drive up the demand for maintenance. Having a reliable forecasting model helps organizations prepare for the future. It improves planning for capacity, ensuring that the necessary resources are allocated to complete the tasks in a safe and timely way.

When managing projects and activities with a variable workload an accurate task duration estimation is essential. Failing to do so can easily lead a project to fail, to miss its deadline or to go over budget. Even experienced project managers struggle at this task and complex methods of estimation have been developed (Institute, 2013) (Karner, 2010).

In our study, we propose a method to estimate a task duration using machine learning models given a text description. This method has the potential of helping other organizations to improve their planning process by employing similar models.

1.4 Scope and limitations

This study was based on the data provided by the Civic Group, a company operating on the maintenance area. The forecasting of workload is based on the maintenance company data, not on the customer's data on maintenance needs. The data collected for this study refers to a period of operation of 22 months.

The text description of the tasks and time to complete them are limited to this company's data. Other organisations may describe tasks in a different way or operate in different areas.

1.5 Chapter summary

This dissertation is organized in the following sections:

Chapter 2: Literature Review

In this chapter, we go over previous work that relates to the objectives of this study. The chapter is divided into two sections that mirror our two research questions. One section covers the task of time series forecasting and how previous related work has approached similar challenges. The second section discusses related work in the area of text classification and recent development on the area of natural language processing.

Chapter 3: Forecasting Workload

The experiment created for the first research question, forecasting the daily workload. We present the methodology to create and evaluate the dataset. We explain the test methodology used, evaluation metric and the results obtained, followed by a brief discussion.

Chapter 4: Task Duration Classification

The second research question is covered here. Task classification based on text description. We present the data analysis, the metric chosen and discuss the methodology employed for the experiment. Finally we present the results obtained, with a brief discussion of the results obtained.

Chapter 5: Conclusion

A brief final chapter outlining the results of this study, their significance and potential future work.

Chapter 2

Literature Review

2.1 Introduction

This chapter is divided into three sections. The first section reviews the field of Machine Learning and Deep Learning. In the second section, we will review the task of time series forecasting and the current state of the art. The third section is dedicated to the task of text classification as part of the field of natural language processing. We will explore previous work in the field and the techniques that have achieved state of the art results in studies that tackle similar problems.

2.2 Machine Learning and Deep Learning

Machine learning (ML) is a field that concerns itself with algorithms capable of learning automatically from data. A simplified explanation would be that if in traditional programming we combine data and a programmed mathematical model to create a wanted output, in machine learning we combine data and the output we want to create a mathematical model. Machine learning algorithms are able to given data and an objective to construct a model capable of achieving that objective to the best of its ability.

Each algorithm takes a different approach to learning. For example, Naive Bayes (NB) calculates the probability of a result given a set of variables. Support Vector

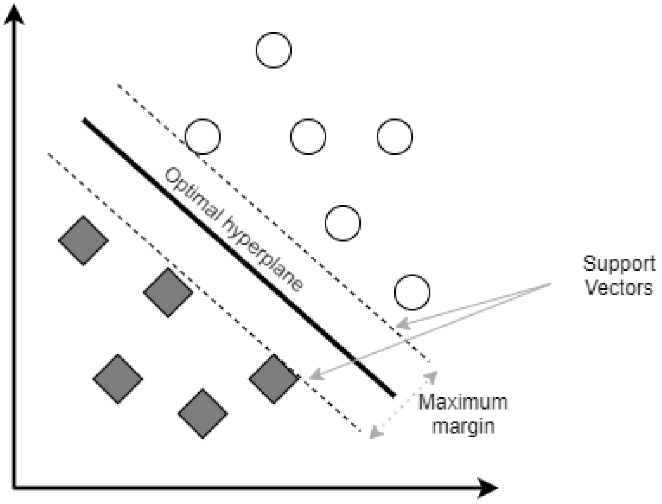


Fig. 2.1 SVM hyperplane in a two dimensional space example

Machines (SVM) builds a hyperplane so that the margin between classes is maximised (fig 2.1). Artificial neural networks (ANN) are built of artificial neurons, loosely based on the human brain. Artificial neural networks learn by means of function approximation using backpropagation (Hecht-Nielsen, 1989) to progressively minimize the error value.

Deep learning (DL) is the field that studies deep neural networks. The word *deep* comes form the way neural networks can stack simple elements to learn complex

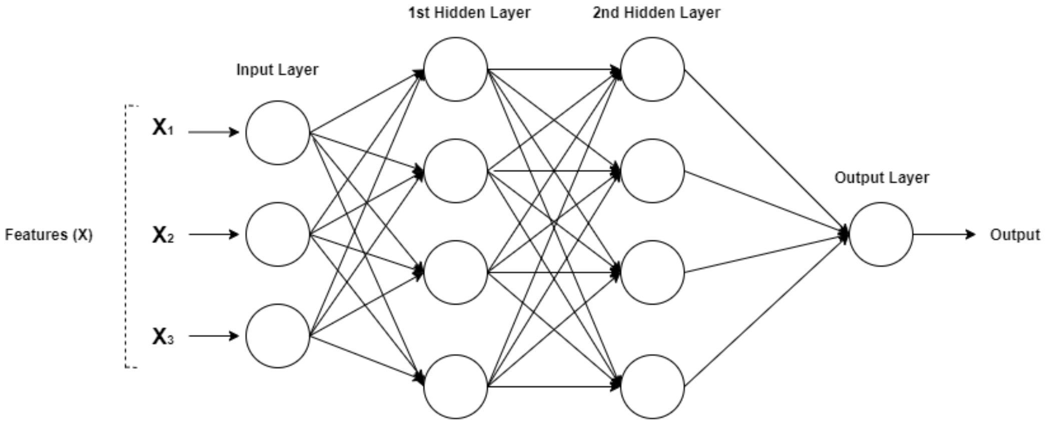


Fig. 2.2 Multi-layer perceptron

concepts (Goodfellow *et al.*, 2016). The idea of stacking simple elements to learn complex relationships is inspired by the structure of neurons on the brain.

A neural network can stack layers of neurons to build new representations of the inputs as is illustrated in fig 2.2. This stacking allows an artificial neural network to create deep networks and learn hierarchical concepts and representations from the inputs. The driving idea that each additional hidden layer can learn a higher abstraction from the previous layers.

In recent years, Deep Learning has gained a lot of popularity. The growth of computing power enabled researchers to explore deeper neural networks and create new architectures that have proven capable of achieving state of the art results in areas such as computer vision, natural language processing and Time Series Forecasting (Chollet, 2017). The quintessential example of a deep learning model is the multi-layer perceptron (MLP) (fig 2.2). The MLP maps a set of input values to a set of outputs values. Each node in an MLP is a neuron (a simple function) and the MLP can form complex mappings by building on this simple functions.

The way that neural networks learn is through a method called backpropagation and stochastic gradient descent (Hecht-Nielsen, 1989), (Amari, 1993). While training a neural network, this method uses the error value to update the parameters of our model so that a better mapping of inputs to outputs is found. Over time more complex architectures of neural networks were introduced, such as convolutional neural networks (CNN) (Fukushima, 1980) and recurrent neural networks (RNN) (Rumelhart *et al.*, 1988).

Convolutional neural networks revolutionized the field of image classification in 2012 when AlexNet (Krizhevsky *et al.*, 2012), a convolutional neural network created by Alex Krizhevsky competed in the ImageNet challenge and won the challenge with an error 10% lower than the second place contestant. Convolutional neural networks have since become a popular architecture for processing images and natural language (Brownlee, 2017). In a CNN convolution layers are used to apply filters to the inputs and generate internal representations that are influenced by neighbour inputs. For the case of image processing, this means neighbour pixels. In natural language (text data) this means neighbour words. Convolution layers are typically followed by pooling layers that select a subsample of the inputs to pass to the following layers.

Recurrent neural networks are a class of neural networks that explore the sequential nature of the input. They are a popular approach to model data that is sequential in nature such as time series and text Goodfellow *et al.* (2016). RNN networks have a hidden state that is learnt based on the previous time steps of the data. These networks are trained with the backpropagation algorithm, but in the case of RNNs the gradient depends not only on the current step but also on the previous time steps. Due to this backpropagation through time recurrent neural networks face two main challenges when training. The problems caused by vanishing gradients and exploding gradients (Gulli and Pal, 2017). If the value of the hidden state is small, as it backpropagates it becomes smaller and smaller meaning that the contribution of previous time steps gets closer and closer to zero and the network learns nothing from the previous time steps. This problem is known as the vanishing gradient. If the value of the hidden state is large, as it backpropagates it becomes larger and larger and become so large that the computer has no representation for it, causing errors. This problem is known as the exploding gradient. These problems make recurrent neural networks harder to train than multi-layer perceptrons or convolutional neural networks (Pascanu *et al.*, 2013). Over the years several approaches have been attempted to overcome these problems. The most popular implementations of RNNs in use today are the Long short-term memory (LSTM) neural networks first proposed by Hochreiter and Schmidhuber (1997) and Gated recurrent units (GRUs) introduced in Cho *et al.* (2014). Both approaches implement mechanisms to deal with the vanishing gradient problem.

When creating a neural network model it is possible to combine different architectures in the same network. By using different architectures it becomes possible to explore different characteristics of the dataset. This method provides improvements in performance when tackling certain datasets (Sainath *et al.*, 2015).

2.3 Time Series forecasting

Capturing the daily workload data from a maintenance company forms what is considered a time series dataset. Time Series refers to data that is captured in sequence at different time intervals. It has long been studied and it has applications in multiple and diverse areas. Measuring a value every hour or every day, for example, generates sequential data that is a time series. The call centre industry faces similar challenges to the maintenance industry, having to plan for capacity based on workload (Aldor-Noiman *et al.*, 2009). We can find studies in the medical field, with methods to forecast patient workload (Olya *et al.*, 2018). Even more traditional organizations such as the manufacturing industry benefit from workload forecasting (Albertetti and Ghorbel, 2020). The usefulness of accurate forecasting is not merely limited to organizations dealing with human activities. In computing, planning for cloud computing capacity based on workload forecasting is an important topic (Fliess *et al.*, 2019). Even when it comes to lower level problems such as application parallelism (Laberge *et al.*, 2019) workload forecast to find the optimal multithreading schedule plays an important role. In our research, we found a gap of studies dealing with forecasting workload in the maintenance area.

The task of time series forecasting is quite general on its applications. This study focus on workload forecasting but the methods used are not limited to workload forecasting. With similar methods, we find applications in familiar areas such as weather forecasting (Karevan and Suykens, 2018), pandemic spread forecast (Perone, 2020) and stock price prediction (Mehtab and Sen, 2020). There is a wide range of applications for time series forecasting.

There is no universal model or method to forecast every problem (Wolpert and Macready, 1997). In the literature, we found two major schools of thought for modelling time series. Models created specifically for time series data (Box and Jenkins, 1970), (Hyndman and Athanasopoulos, 2018) and general purpose models that can be applied to time series (Brownlee, 2018a).

Classical literature on time series like Box and Jenkins (1970) and Winters (1960) proposed the methods Auto Regressive Integrated Moving Average (ARIMA) and Exponential Smoothing (ES) as time series forecasting methods. Despite their age, ARIMA (and its variants) and ES are still widely used today (Hyndman and Athanasopoulos, 2018), Perone (2020). Both ARIMA and ES methods are focused on time series data. A more recent model created for time series data is the Prophet model, created and released by the Facebook Core Data Science team in Taylor and Letham (2017). The Prophet model aim is to provide a fast and accurate forecasting method able to scale with large datasets. These three methods approach time series forecasting in different ways, but they have certain expectations on the behaviour of the data. They tend to favour recent observations and variations as predictors over older ones and expecting features such as trend and seasonality.

A second approach to time series forecasting is to transform the time series forecasting problem into a supervised learning problem and then apply general purpose regression methods. In this approach, the model applied can come from classic machine learning or from the most recent developments in deep learning. There are numerous studies that take this approach. For example in Mei *et al.* (2014) the ensemble method Random Forest is employed to forecast the price in New York electricity market or Support vector machines in the work of Sansom *et al.* (2003). The recent popularity of Deep Learning has created an interest in exploring Deep Learning for time series forecasting. A lot of recent publications such as Brownlee (2018a), Lai *et al.* (2017) and Gamboa (2017) explore the use of Deep Learning and neural networks such as Multi-layer Perceptrons (MLP), Convolutional Neural Networks (CNN), and Recurrent neural networks (RNN). Recurrent neural networks have been particularly explored for their capability to handle sequences of data due to their design (Fu *et al.*, 2016). Specifically, Long Short-Term Memory Networks (LSTM) and Gated recurrent units (GRU) (Goodfellow *et al.*, 2016) are popular choices for sequence modelling.

From simple methods like linear regression to advanced recurrent neural networks, time series forecasting is a fertile field being explored by researchers and industry alike.

2.4 Text Classification

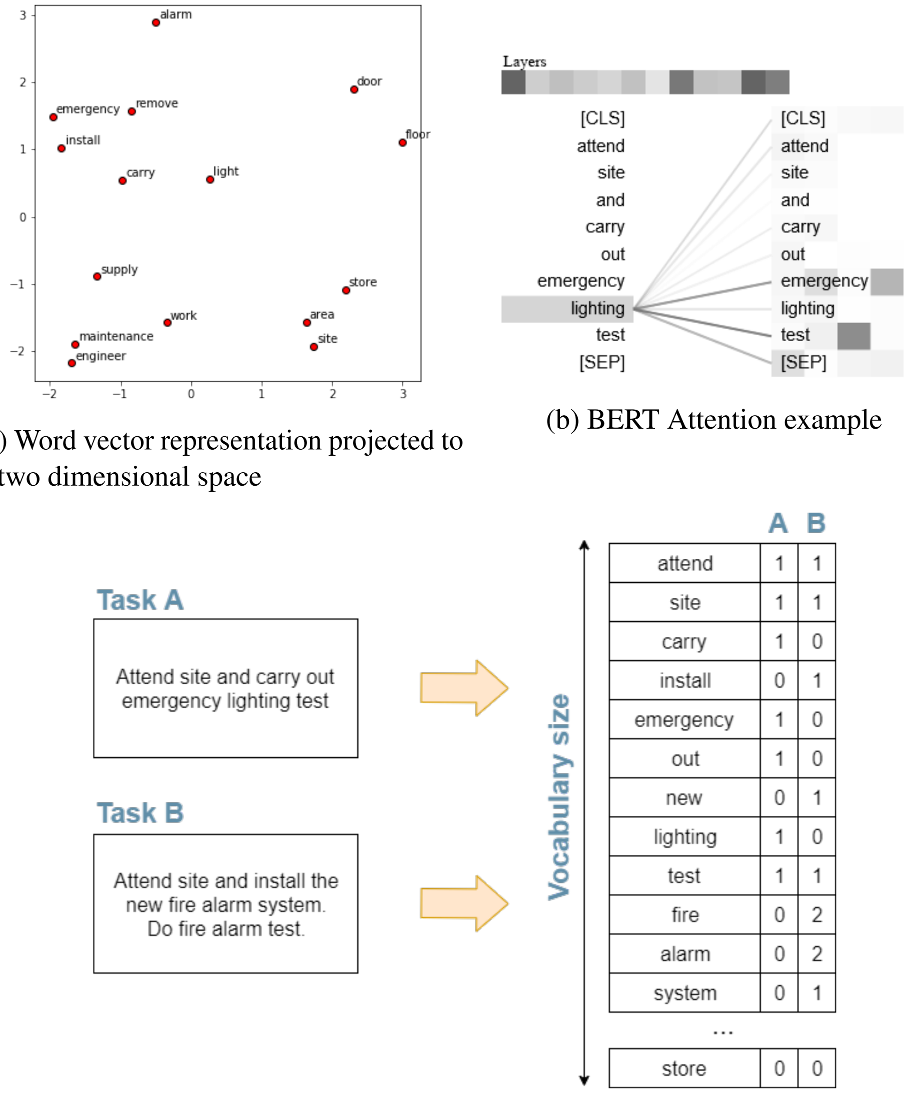
Natural language processing (NLP) is the field of automatic analysis and processing of human language. It is a vast field that draws from knowledge linguistics, computer science, statistics and artificial intelligence. Natural language processing tasks range from text classification to relationship extraction, language translation, speech recognition and many more.

For our research question, we want to develop a method that given a task description is able to classify it as a short duration task or a long duration task. This task falls in the text classification task of natural language processing. Text classification concerns itself with the classification of text samples into classes or categories. In recent years this area has seen numerous innovations both in methods and applications (Young *et al.*, 2017). The growing amount of data available motivates the research, with much of the generated data being in text form.

Natural language processing has a long history behind it. Its beginnings can be traced to at least the 1950s. Alan Turing famously proposed the Turing test in Turing (1950). Of the first experiments, the most famous is the Georgetown experiment by Georgetown University and IBM. At this time, researchers searched for models capable of representing knowledge in computers such as knowledge graphs and ontologies. Researchers tried to map real world objects and their relations in a computer with complex structures. Roger Schank introduced the Conceptual dependency theory in Schank and Tesler (1969) with the goal of representing the meaning of a phrase independent of the words used. There were also efforts made to decompose phrases by syntactic rules and parts of speech (Woods, 1970). In these early years, most of the research was based on expert systems, complex sets of handwritten rules and decision trees to emulate intelligent behaviour.

In the 1980s and 1990s we start to see the use of machine learning methods and statistical text representation in natural language applications (Biebricher *et al.*, 1988), Crawford *et al.* (1991). Text representations such as bag of words (BOW) and term frequency inverse document frequency (tf–idf) represent individual words as a long vector and the occurrence for each word in a text see fig 2.3c. Some of the first machine learning methods used for text classification use the Naive Bayes model with the bag of words representation (Lewis *et al.*, 1996), (Heckerman *et al.*, 1998). It is still common to use this method as a baseline today.

Support Vector Machines (SVM) became a widely used text classification model achieving good results even in recent work (Van-Tu and Anh-Cuong, 2016).



(a) Word vector representation projected to a two dimensional space

(b) BERT Attention example

Attend site and carry out emergency lighting test

Attend site and install the new fire alarm system. Do fire alarm test.

	A	B
attend	1	1
site	1	1
carry	1	0
install	0	1
emergency	1	0
out	1	0
new	0	1
lighting	1	0
test	1	1
fire	0	2
alarm	0	2
system	0	1
...		
store	0	0

(c) Bag of Words example for two short text documents

Fig. 2.3 Different Word Representations

In more recent years Deep Learning has become the dominant approach in text classification Young *et al.* (2017). One of the innovations brought by Deep Learning were neural representations of words. Deep learning enabled multi-level representation learning. Word2Vec (Mikolov *et al.*, 2013b) was one of the first methods that made neural representations popular. Mikolov et al proposed CBOW and skip-gram models for representation learning. CBOW calculates the probability of a word given a context and skip-gram calculates the probability of a context given a word. The resulting word embedding represents words as multi-dimensional word vectors (fig 2.3a).

While simple in concept Word2Vec proved extremely powerful (Young *et al.*, 2017) achieving state of the art performance in multiple problems. Word2Vec representations also exhibit interesting proprieties such a distributional semantics and meaningful syntactic and semantic relations between vectors as explained in Mikolov *et al.* (2013c). Over the years other word vector representations have been proposed and became popular such as GloVe (Pennington *et al.*, 2014). Transfer learning of word vectors became widely used. Word vectors such as Word2Vec and GloVe were pre-trained over a large corpus of unlabeled text. These trained word vectors became publicly available and could then be incorporated into problem specific models, transferring the learned representations and giving the models a head start. Neural network architectures such as CNN and LSTM combined with word vectors became popular for text classification (fig 2.4).

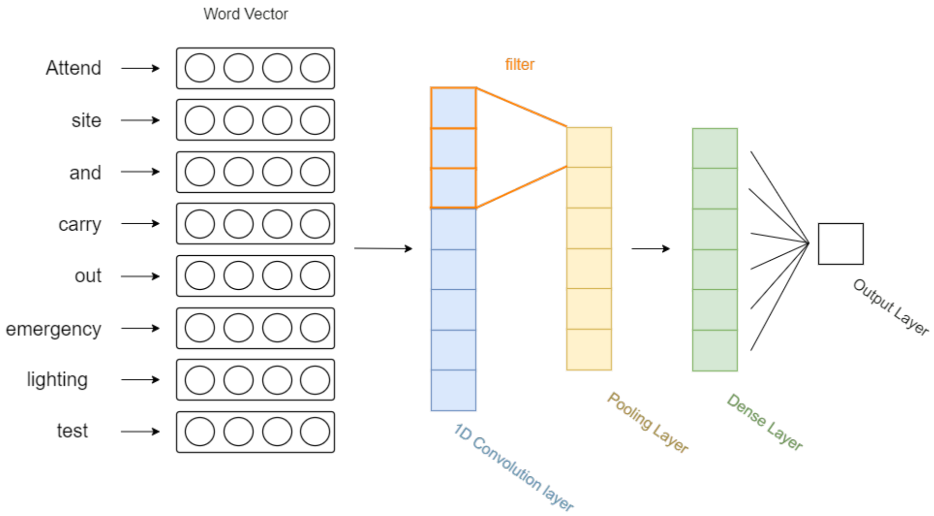


Fig. 2.4 CNN text classification example

In more recent works we saw the rise of attention mechanisms and the transformer architecture (Vaswani *et al.*, 2017). Attention mechanisms can use multiple layers to let a word pay a weighed "attention" to surrounding words in its representation (see 2.3b). The models BERT (Devlin *et al.*, 2019) and XLNet (Yang *et al.*, 2019) have been introduced based on these innovations. They have achieved state of the art performance in text classification tasks as can be seen in Ruder (2020) and Paperswithcode (2020). The model fastText (Joulin *et al.*, 2017) incorporated attention mechanisms combined with CBOW and skip-grams and achieved great results.

Text classification sees practical application in the fields of spam filtering, sentiment analysis, opinion mining, contextual search and topic modelling (Dalal and Zaveri, 2011). One of the most popular tasks in text classification is sentiment analysis, often framed as a binary classification problem (Thongtan and Phienthrakul, 2019). The objective of sentiment analysis is to classify a given text into positive or negative sentiment. This has wide applications in marketing, where customer's reviews and opinions in relation to products and services can be automatically analysed (Indurkhya and Damerau, 2010) (Samuels and Mcgonical, 2020). Document classification is another common task. In this case, the objective is to identify the topics or subjects when given a text document. Practical application can be seen in classifying news articles into topics or research papers into subjects as can be seen in Yang *et al.* (2018).

In our research, we found no previous works that applied machine learning models to estimate a task duration from a text description. This is likely to the unavailability of this type of dataset since it typically involves sensitive company data. And also likely due to the relation between a text description and a task duration not being as clearly identifiable as a topic or sentiment. It is the objective of this study to establish that such a relationship can exist and that machine learning models can be used to predict task duration from text description.

Chapter 3

Forecasting Workload

3.1 Introduction

In this chapter we will look at the task of forecasting the workload for the next day based on historical data. Using a mathematical model to describe real-world behaviour is a widely used practice as it is well put by Box and Jenkins in their classical book *Time series analysis, forecasting and control.*

"The idea of using a mathematical model to describe the behaviour of a physical phenomenon is well established. In particular, it is sometimes possible to derive a model based on physical laws, which enables us to calculate the value of some time-dependent quantity nearly exactly at any instant of time."

- Box and Jenkins (1970)

The model that we propose to create is about modelling workload, an aspect of human activity. Which is safe to assume that is not completely deterministic. There is more uncertainly involved with human activity than with physical laws. Humans often present complex and non-deterministic behaviour. But by using a rigorous methodology to test and evaluate models, it is possible to measure and identify the ones that provide the best predictions.

"All models are wrong, but some are useful"

- George Box

3.2 Methodology

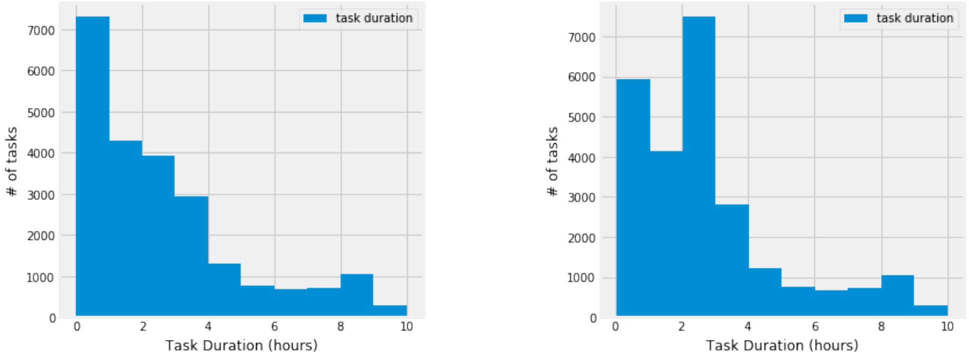
3.2.1 Dataset creation

The data that we are analysing in this study comes from a production SQL database of a company operating in the maintenance industry. From the database, we obtain a list of all completed tasks. For the purpose of this study, we need to transform this data into a fixed step time series. After retrieving the list of tasks we observe that some tasks contain null dates. When dealing with missing data, there are two options to approach the problem, to remove the data or to impute the data (Hyndman and Athanasopoulos, 2018). We opted to remove those tasks that have null dates since it would be impossible to know when they took place. The final dataset covers the period of February 2018 to the end of November 2019, for a total of 668 days.

We then calculated each task duration based on the start date and completion date timestamps. Examining the calculated task's duration we see that there are some oddities in the dataset. The minimum task duration is under -153 days and the maximum over 365 days. Given the nature of the data and knowledge of the industry, we know these outliers to be impossible. Every task was completed on the same day, for a maximum length of 10 hours (8 normal hours and 2 hours of overtime). These values are likely due to user input error. For every task with zero duration, negative duration, and duration over 10 hours, we assumed that the duration is invalid.

Then we imputed the tasks with invalid task duration with the mean value for task duration (fig 3.1). This is an important step, because removing the tasks would cause a drop in the corresponding day's workload value. The tasks were completed, merely reported with an invalid duration. So imputing it with the mean gives them a representation on the dataset without changing the overhall distribution dramatically (fig 3.1).

A total of 25,092 tasks constitute this dataset, of which 3,486 were imputed (13.9%). Finally, we sum all the task duration's per day using a rolling window of one



(a) original

(b) after imputing

Fig. 3.1 Task duration histogram

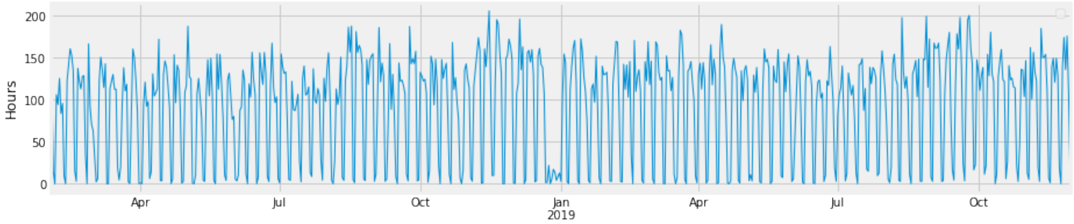


Fig. 3.2 Workload hours per day

day to obtain the total daily workload. The plot of the final dataset, representing the workload per day can be seen in fig 3.2. This graph represents the total number of hours of work completed per day, for all maintenance engineers.

3.2.2 Data Analysis

This section will present the data analysis done to the dataset to better understand it, the statistical characteristics it presents and detectable patterns.

Looking at the plot of the workload time series (fig 3.2) it is possible to detect some patterns. There seems to be a distinct cycle of the workload, with a steep dip on the weekends that see a much lower workload than weekdays. The workload seems to have some monthly variation, with a dip after July and an increased workload in the weeks leading to Christmas. This matched the maintenance industry expectation where the stores request more maintenance due to the Christmas shopping season. And there is a noticeable gap just before January, this corresponds to the Christmas week. This week is celebrated in Ireland with businesses slowing down work. All of

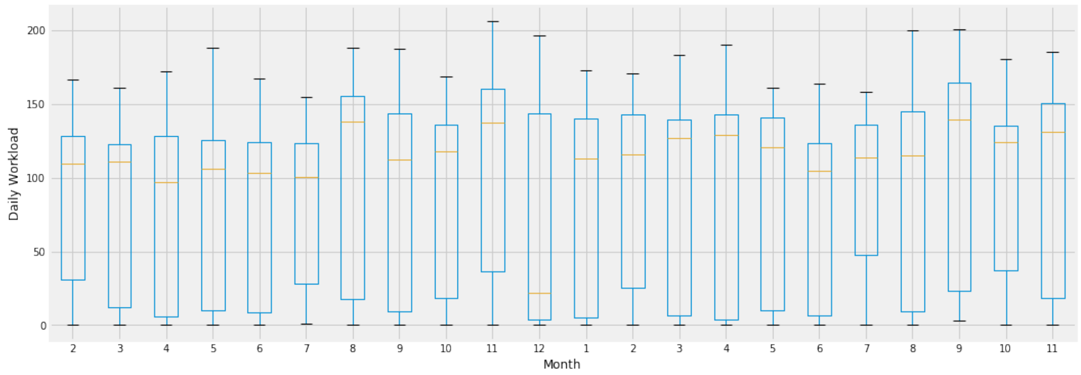


Fig. 3.3 Box-plot of daily workload distribution on each month

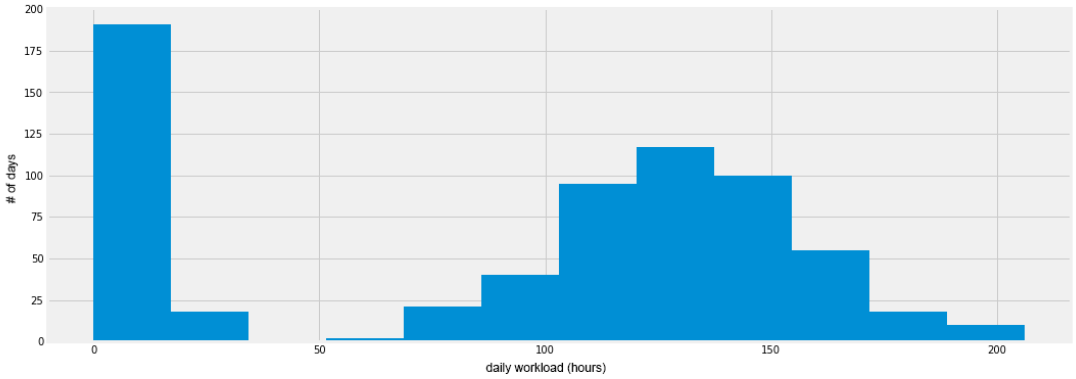


Fig. 3.4 Daily workload histogram

this is great news. They indicate that the workload time series does not have a random behaviour, so there is the potential for skilful forecasting.

In fig 3.3, we present the box-plot of the distribution of the daily workload per month. In this plot, we see a variation of the daily workload distribution from month to month. Industry knowledge can explain some of the monthly variance, such as a higher workload is expected in the month of November due to the stores needing more maintenance work to prepare for the holiday season and a dip of foot traffic in the first months of the year.

In fig 3.4, we show a histogram of the overall distribution on the daily workload. The plot seems to follow a normal distribution on the right side. The left group of bars correspond to the workload on the weekends and holidays. The company still operates on those days, but only performs emergency works and has limited employees on duty, resulting in a much lower workload values.

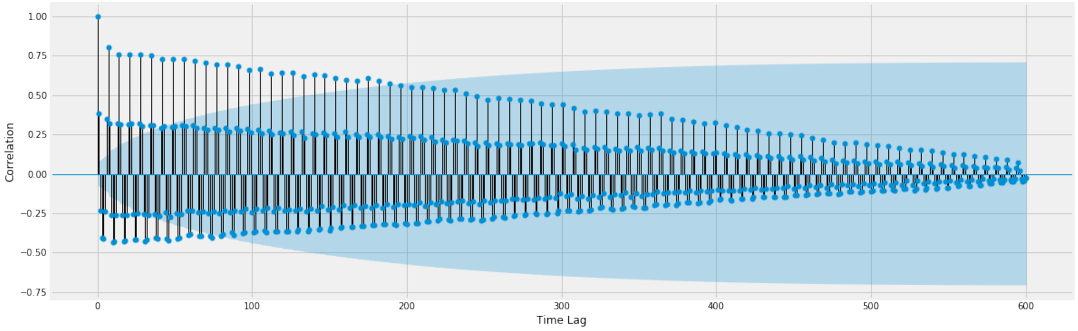


Fig. 3.5 Auto-correlation plot

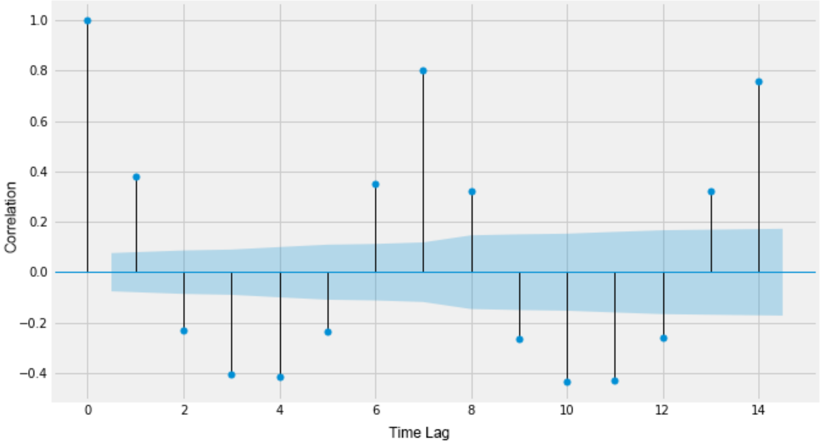


Fig. 3.6 Auto-correlation plot (detail)

A common practice in time series forecasting is to draw an auto-correlation plot (Brownlee, 2018b). This is a plot where a time step t is tested for correlation with the previous time steps ($t - timelag$). In our dataset, we can think of it as testing for correlation of the current day's observation against yesterday's observation, the day before yesterday, and so on.

In fig 3.5, we show the auto-correlation plot tested over 600 days and in fig 3.6 we have a more detailed view of the correlation of the previous 14 days. We can see that there is a strong correlation in time lag 7, of approximately 0.8 and a similar but slightly lower correlation in time lag 14. What this means is that the workload of the current time step t is correlated to the workload of the time step $t-7$. In a more practical example, this means that the workload on this Monday is strongly correlated to the workload of the previous Monday (last week) and a somewhat lower correlation with the Mondays in the preceding weeks.

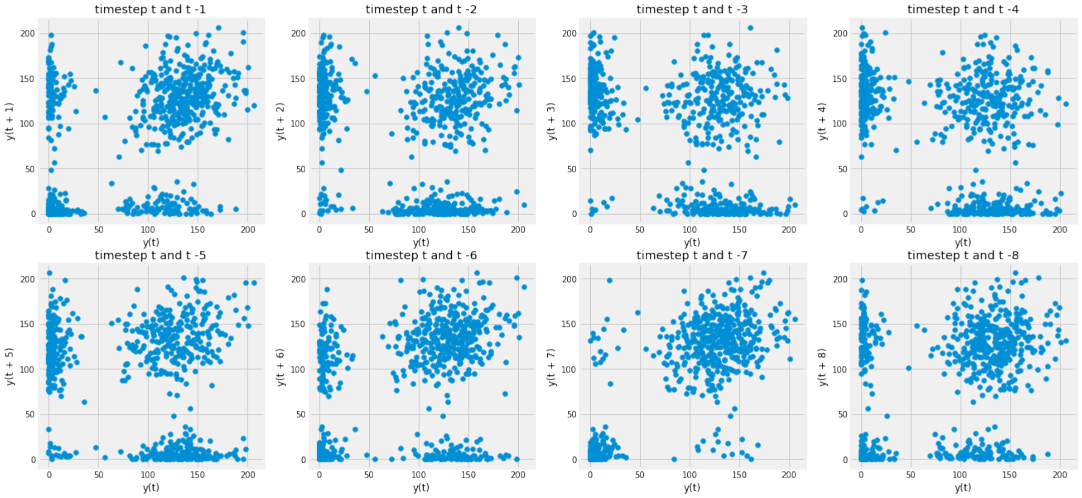


Fig. 3.7 Correlation between time step *t* and previous time steps (days)

In the auto-correlation plot, we see that there also seems to exist some correlation between other previous days but to a much lower degree. This is expected due to the dip in workload on the weekends. The value on Monday will be much higher than the value on Sunday. We plot the correlation for time steps -1 to -8 in fig 3.7. While not quite as obvious as in the auto-correlation plot, time step t and $t-7$ present the strongest correlation.

In more classical works on time series like in Hyndman and Athanasopoulos (2018), it is common to see time series as a combination of components. The components are the trend (T_t), the seasonal component (S_t) and the residual component or remainder (R_t). The components are then combined as an additive model or a multiplicative model based on the dataset characteristics.

additive model: $y_t = S_t + T_t + R_t$
multiplicative model: $y_t = S_t \times T_t \times R_t$

It is possible to perform classical time series decomposition based on an additive or multiplicative model. For our dataset, the additive model was the better fit. The relation between trend, seasonality and residuals in our dataset are better represented by adding them together. The multiplicative model is better suited to datasets where the effect of trend and seasonality get exaggerated over time. We performed a classical

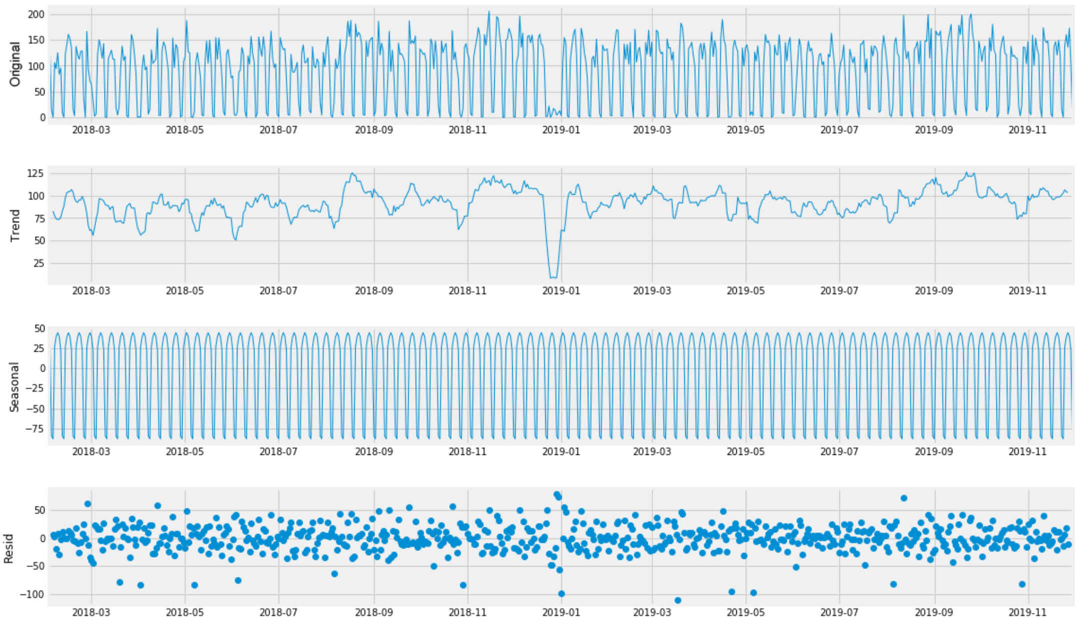


Fig. 3.8 Decomposed time series using additive model and frequency of 7

decomposition based on the additive model and present it in fig 3.8. We can see that the additive decomposition captured the weekly seasonality in the seasonal component and the general trend of the dataset. While this method is not perfect, it still manages to capture a pattern in the dataset. This is a good indication that there is information to be captured by our forecasting models. The data presents some recognizable patterns and not merely random variation.

3.2.3 Evaluation metric

To evaluate the quality of the models in this experiment we must first decide on a metric. We are interested in how close the predicted value is to the real value, so we will need to measure the difference between the predicted and actual value and minimize the error. The most commonly used metrics are MAE (Mean Absolute Error), MSE (Mean Square Error), RMSE (Root Mean Square Error) and MAPE (Mean Absolute Percentage Error) (Hyndman and Athanasopoulos, 2018). The different metrics have different advantages and drawbacks.

MAE has the advantage of presenting the error value on the same metric as the dataset. As a drawback MAE treats small error and big error values equally, not penalizing models that on occasion produce wildly inaccurate predictions. MSE has the advantage of penalizing high error values more than low errors. MSE has the disadvantage of presenting the error as a value that does not have a frame of reference for interpretation. RMSE has the advantage of bringing the error value to the same units as the values, making it intuitive to interpret the results. Additionally, it keeps the advantage of benefiting consistently low error models. MAPE has the advantage of presenting its error value as a percentage value in relation to dataset. MAPE has the drawback that in data sets where the values reach zero encounters a division by zero error. In our dataset the workload value reaches zero in some weekends, making MAPE an invalid choice of metric.

After looking at the advantages and disadvantages of each metric Root Mean Square Error stands out as the best choice for a metric. RMSE is widely used as a metric to evaluate forecasting problems (Hyndman and Athanasopoulos, 2018) and we can see why. It penalises models where the forecast deviates too far from the actual value and presents the result in the same unit of measure as the data. The target of the experiment is to find a consistent, accurate model so we chose RMSE as the metric as our evaluation metric. Since RMSE gives us the error in hours we will try to find the model that produces the lowest error.

3.2.4 Experiment design

For this experiment, we are interested in forecasting the workload for the next day based on workload history.

A typical method of evaluation of time series models is using a training-test set split. In this evaluation, the data is divided into a training set and into a test set. The model is then trained on the training set and evaluated on the test set. No part of the test set is used to train the model. This has the disadvantage of reducing the size of the data to be trained. Since a lot of time series problems have a relatively short sample size (because they are dependent on time) this may make the training set too small to capture more complex patterns. Ideally, the model would be trained in as much data as possible for a better estimate of real-world performance.

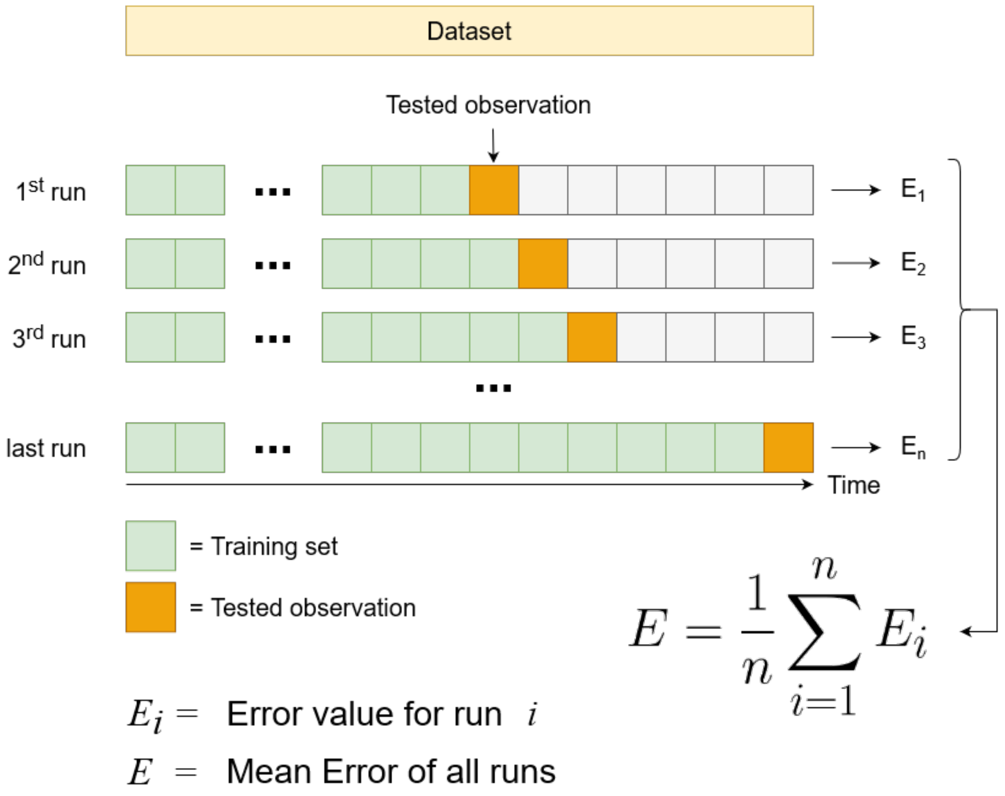


Fig. 3.9 Walk-forward validation diagram

Traditional cross-validation use for testing time series is controversial since we would be using data from the future to test the past. In the real world, we don't have access to information from the future.

A more sophisticated method is the method of walk-forward validation. This method can also be called time series cross-validation in some sources like in Hyndman and Athanasopoulos (2018). To avoid confusion with the k-fold cross-validation method, we will refer to it as walk-forward validation in this study as is referenced in Brownlee (2018b). Both sources credit this method as the most sophisticated and reliable estimation of the performance of a time series forecasting model. It is also the method that more closely approximates a real-world application, where the model would be updated as more data is created thought the passage of time.

Walk-forward validation works by reserving the last entries of a dataset for testing purposes and progressively training, testing in one observation, and adding the observation to the training set. So in a dataset of size n and a test size t the first test iteration would be trained on samples 0 to $n-t$ and tested on the observation $n-t+1$

and then observation $n-t+1$ is added to the test set, the model is retrained and tested on observation $n-t+2$ until the final observation n and the average of the errors is calculated. A visual representation of this can be seen in fig 3.9. The final error value E is obtained by calculating the mean of the error value of each run (E_i).

We use the RMSE metric to calculate the error between the forecasts and the actual values for each forecasted day. For our experiments, we reserved the last 60 time steps to be tested with Walk-forward validation. We present the results as RMSE error values since this is the metric that better fits our objective and dataset. The lower the RMSE value, the better the model performance.

3.3 Results

The results section is divided into three main experiments. On the first experiment we tested the models using only the historical data of workload in the *Univarible* section. We performed a second experiment where we tested models using the historical data of workload and date information in the *Multivariable with date features*. We performed one last experiment where we added weather features in addition to the date information in the section *Multivariable with date and weather features*. In the last section, we discuss the results of the experiment.

3.3.1 Models tested

Naive models

Naive models are meant to be used to establish a baseline on a time series forecast problem. They are simple in nature but serve as a reference for comparison with more complex models to better evaluate their skill.

Persistence: The persistence model is a naive model that merely uses the previous observation t_{i-1} as the forecast for the time step t_i being predicted. It is a naive method that in the case of our dataset assumes that what happened today is going to happen tomorrow.

Persistence -7: From the data analysis performed on the dataset we saw a strong correlation on the time lag -7. We decided then to create a second persistence model with a time lag of -7 instead of -1. The Persistence -7 predict the previous observation t_{i-7} as the forecast for the time step t_i. In practical terms for our dataset, we can say that if the day to be predicted is a Monday, the model we created predicts the workload value of the last week's Monday.

Mean: In datasets that do not show a strong growing or declining trend, the mean value can provide a naive but useful forecasting method. We include a naive model that uses the mean value of the training data as the prediction.

Time Series models

These are models that have been created for the purpose of forecasting time series data. They are based on a statistical background and tend to do some assumptions on the behaviour of the time series.

Autoregressive (AR): The AR model uses the values from previous time steps as input to a linear regression model to predict the next value (Box and Jenkins, 1970).

Moving Average (MA): The moving average model uses the current error and previous errors as predictors for the next time step to smooth the data points (King, 1912).

Autoregressive integrated moving average (ARIMA): ARIMA combines the Autoregressive (AR) model with the Moving Average (MA) model and a differencing factor (I) for a more complex model for time series forecasting. ARIMA and Exponential smoothing are the two most widely used forecasting model for time series (Hyndman and Athanasopoulos, 2018).

Seasonal Autoregressive integrated moving average (SARIMA): This model is similar to the ARIMA but with the inclusion of a seasonal component. SARIMA is commonly used to model seasonal time series (Hyndman and Athanasopoulos, 2018), (Manayaga and Ceballos, 2019).

Exponential Smoothing (ES): Exponential smoothing models time series by performing weighted averages os previous time steps. The weights are higher for recent time steps and lower for older time steps. It is a widely use method for time series forecasting (Hyndman and Athanasopoulos, 2018), De Gooijer and Hyndman (2006)).

prophet: This model was introduced by a Facebook researcher team in Taylor and Letham (2017). It was designed with performance in mind, to forecast time series at scale. The prophet model is based on an additive model with support of yearly, weekly, and daily seasonality.

Machine learning models

It is possible to use regression models from the area of machine learning to produce a forecast by framing the problem as a supervised learning problem. We can transform the historical time series data into supervised learning dataset where the inputs are previous time lags and the output is the value to be forecasted.

Linear Regression LR: A classical statistical model that finds the best fitting line.

Polynomial Regression: A regression method that fits a polynomial of the nth degree to the dataset (Gergonne, 1974).

Support Vector Regression (SVR): Support Vector Machine based regression model first introduced in Drucker *et al.* (2003).

XGBoost: Optimized distributed gradient boosting decision tree ensemble (Chen and Guestrin, 2016).

Random Forest: Regression ensemble model that calculates the mean predictions of the individual trees (Breiman, 2001).

K Nearest Neighbors Regression (KNN): find the k closest training samples and uses them for prediction (Altman, 1992).

Neural network models

Neural networks can similarly be used to forecasts time series by framing the problem as a supervised learning problem. They have seen a growing interest and popularity in recent years to tackle a multitude of problems, time series among them. For this experiment, we created and tested different neural network architectures. We created and performed a grid search for network topology initialization parameters and optimization algorithm.

MLP: Multi-Layer Perceptron.

CNN: One dimensional Convolutional Neural Network.

LSTM: Long short-term memory recurrent neural network (RNN).

GRU: Gated recurrent unit recurrent neural network (RNN).

Bi-LSTM: Bi-directional Long short-term memory neural network.

CNN LSTM: Convolutional LSTM Network implementation as per Shi *et al.* (2015).

3.3.2 Univariable

For the univariable experiment, we tested a wide range of models from classical time series models like ARIMA to more recent models like the CNN LSTM neural network. Due to the technical differences between models, the data preparation varied slightly between models.

Models based on neural networks showed better results when the data was normalized (converting the data to a 0 to 1 range). This was achieved by dividing the workload by a fixed value *n* to reduce the scale to 0 to 1 and the forecast was then multiplied by *n* before testing.

The models we present on table 3.1 were optimized with a hyperparameter grid search and the best model is presented.

Table 3.1 Univariable models results

Model	RMSE
Persistence	70.006
MA	60.064
Mean	60.012
LR	59.810
Persistence -7	32.721
AR	28.431
ARIMA	27.810
SVR	24.566
LSTM	23.969
ES	23.067
XGBoost	22.953
Polynomial Regression	22.596
SARIMA	22.451
Random Forest	22.208
prophet	21.811
GRU	21.131
KNN	20.476
Bi-LSTM	20.244
CNN	20.105
MLP	19.451
CNN LSTM	**18.757**

The best model of this experiment was the CNN LSTM model with an RMSE of 18.757. This value improved greatly on the naive models. Neural network models performed remarkably well in the univariable experiment. The best four models found were all based on neural networks.

3.3.3 Multivariable with date features

In this experiment, we added date features to the original dataset. The date features added were:

- day of the week
- day of the month

- day of the year
- week of the year
- month
- year
- yearly quarter
- is a bank holiday

For this experiment, we selected the models that performed well in the previous experiment and support multivariable input. Some models like Auto Regression have the limitation of only supporting a one variable time series as input. A note on the multivariable prophet model. The prophet model does not support multivariable inputs. But it supports the addition of a calendar of holidays, yearly seasonality and daily seasonality (Taylor and Letham, 2017). Because of this, the data preparation for the prophet model was different from the other models. It still includes date features, but technically the input data is not the same as the other models. The prophet model shown here included the original dataset and a calendar of Irish holidays.

The results are presented on table 3.2. The best model of the experiment was the Random Forest forest model with an RMSE of 15.810.

Table 3.2 Multivariable models with date results

Model	RMSE
CNN LSTM	18.784
KNN	18.432
CNN	18.202
LSTM	16.964
prophet	16.804
GRU	16.540
XGBOOST	16.215
Bi-LSTM	16.096
MLP	15.810
Random Forest	**15.215**

3.3.4 Multivariable with date and weather features

In this final experiment we tested if the best models from the previous experiment showed improvements when trained with the addition of weather features. Weather is a potential exogenous variable for the need for maintenance work so it was worth investigating. We obtained historical acquired weather data from Met Éireann (the Irish National Meteorological Service). We processed the weather data and added the following features to the dataset:

- Precipitation Amount
- Maximum Air Temperature
- Minimum Air Temperature
- 09 utc Grass Minimum Temperature
- Mean 10cm Soil Temperature
- Mean Wind Speed
- Highest 10 minute mean wind speed
- Highest Gust
- Sunshine duration

Table 3.3 Multivariable models with date and weather features results

Model	RMSE
Bi-LSTM	19.574
GRU	16.641
XGBOOST	16.576
MLP	16.071
Random Forest	**15.487**

The results of this experiment can be seen in table 3.3. Most of the models tested performed slightly worse with the addition of weather features. This suggests that the weather did not provide useful information for the forecast.

3.3.5 Results Compiled

This section compiles the results for the experiment of workload forecasting. We present all the results in the table 3.4 and fig 3.10 for easy comparison. The table 3.4 presents all the models tested. The models are grouped into 3 sections. The top section of the table contains the results for the univariable models, that only use the workload historical data as training. The best model in this section was the CNN LSTM with an RMSE of 18.757. A neural network introduced by Shi *et al.* (2015) that uses Convolutions in combination with Long short-term memory for forecasting.

The middle section of the table shows the results for the models that use workload historical data and date features as training data. Here the best model was the Random Forest introduced in Breiman (2001) with an RMSE of 15.215, improving on the univariable models.

The last section of the table presents the results for the models trained using workload historical data, date features and weather features. Once again, the best model was Random Forest with an RMSE of 15.487. Obtaining slightly worse score than in the previous section. The additional weather data did not provide an improvement on any of the models tested when compared to the models trained with historical data and date features. In fig 3.10 we show the results in graphical form.

Table 3.4 Workload forecasting models results compiled

Model	RMSE
Univariable Models (Historical data)	
Persistence	70.006
MA	60.064
Mean	60.012
LR	59.810
Persistence -7	32.721
AR	28.431
ARIMA	27.810
SVR	24.566
LSTM	23.969
ES	23.067
XGBoost	22.953
Polynomial Regression	22.596
SARIMA	22.451
Random Forest	22.208
prophet	21.811
GRU	21.131
KNN	20.476
Bi-LSTM	20.244
CNN	20.105
MLP	19.451
CNN LSTM	18.757
Multivariable Models (Historical and Date data)	
CNN LSTM	18.784
KNN	18.432
CNN	18.202
LSTM	16.964
prophet	16.804
GRU	16.540
XGBOOST	16.215
Bi-LSTM	16.096
MLP	15.810
Random Forest	**15.215**
Multivariable Models (Historical, Date and Weather data)	
Bi-LSTM	19.574
GRU	16.641
XGBOOST	16.576
MLP	16.071
Random Forest	15.487

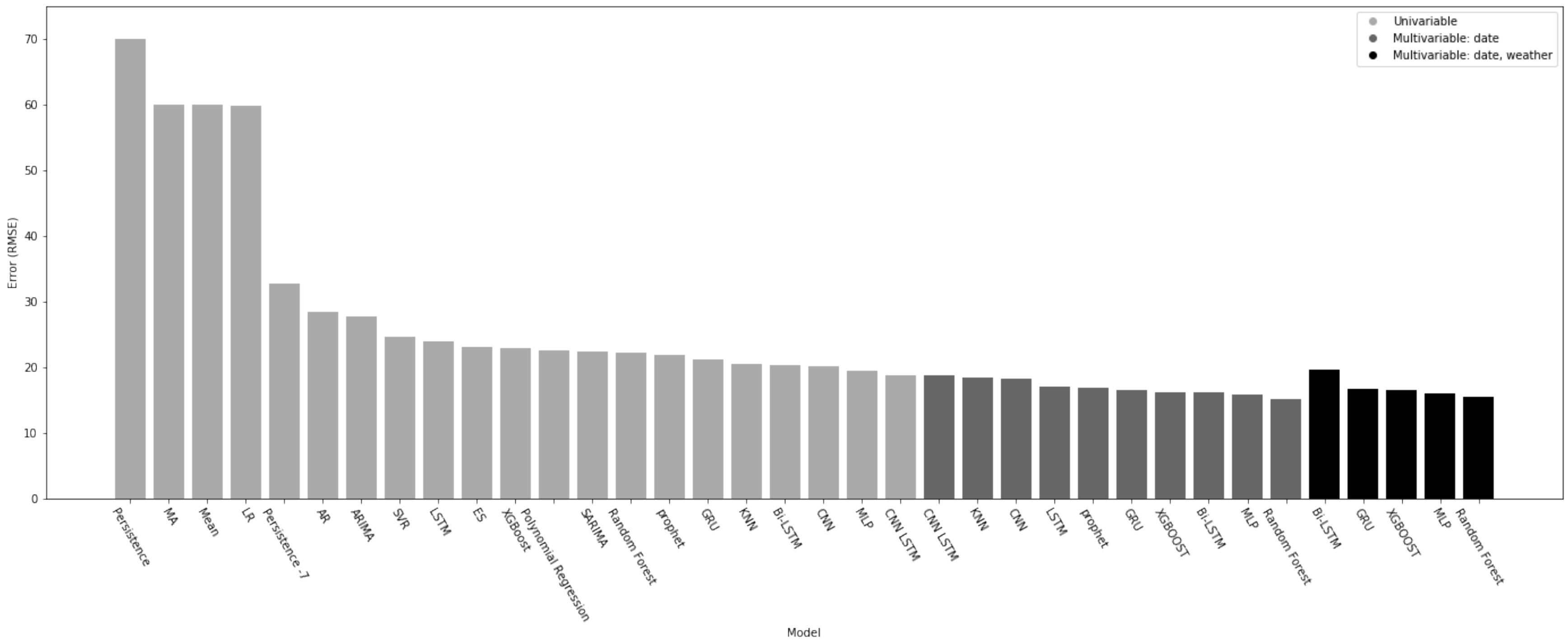


Fig. 3.10 Workload Forecasting Models Performance

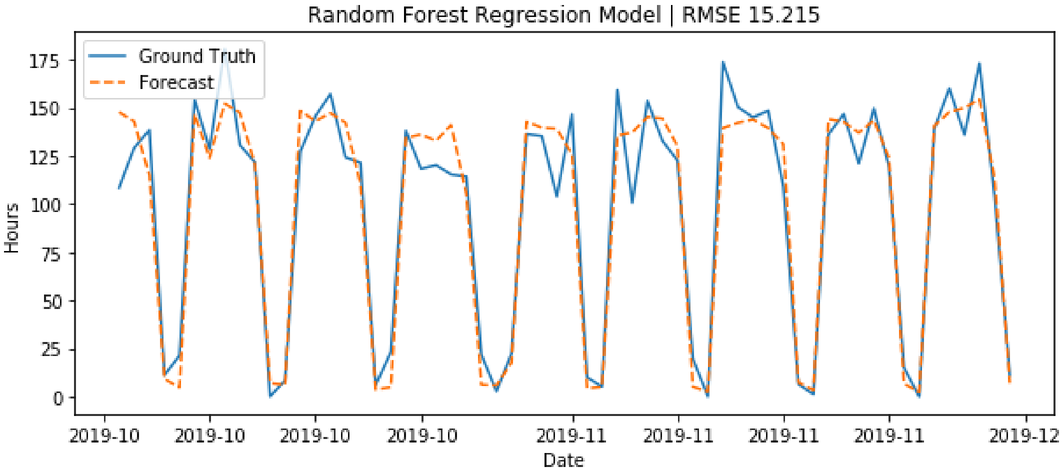


Fig. 3.11 Random Forest forecast

3.3.6 Random Forest forecast

The Random Forest model that used historical data and date features as input was the most accurate found in the experiment. In fig 3.11 we plot the forecast values obtained by the random forest model (dashed line) to the actual workload value (solid line). We can see that this model provided a forecast that closely matches the ground truth and does not deviate far.

We can also analyse the residuals to check if there are indications of information that was not captured. The residual values are the values not captured by the model and can be obtained by subtracting the forecast from the ground truth.

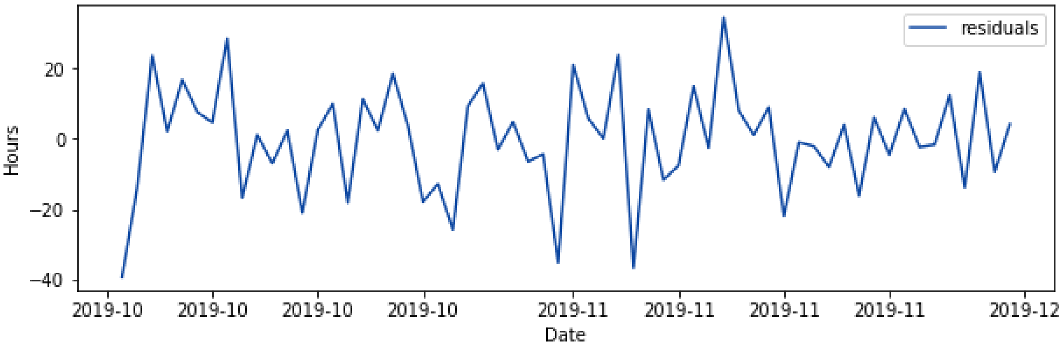


Fig. 3.12 Random Forest residuals

The residuals plot (fig 3.12) does not suggest a particular pattern to the residuals. By plotting their distribution (fig 3.13) we see that they seem to follow a fairly normal

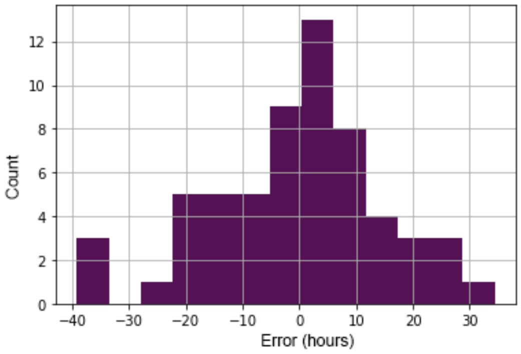


Fig. 3.13 Random Forest forecast error distribution

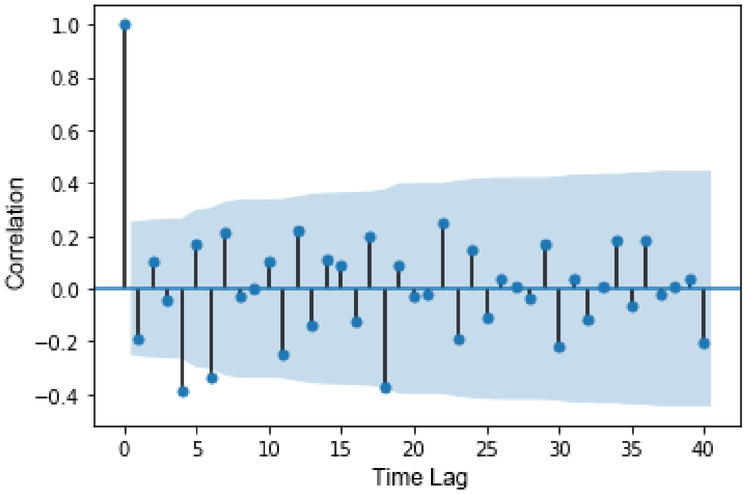


Fig. 3.14 Random Forest forecast residuals auto correlation plot

distribution. Most values are close to zero and no particular pattern emerges. This is an indication of a good fit of the model as per Brownlee (2018b). Performing an auto-correlation test on the residuals we see that it shows very small correlation values (fig 3.14). This is another good indication of a good fit (Brownlee, 2018b) since we want any correlation to be captured by the model.

Given that we know the residual distribution, it is possible to calculate a confidence interval on the predictions as per Reilly (2019). We calculate a confidence interval of 80% and present the resulting plot in fig 3.15. The orange area represents the confidence interval. We can observe that the forecast is well behaved, rarely falling outside the confidence interval.

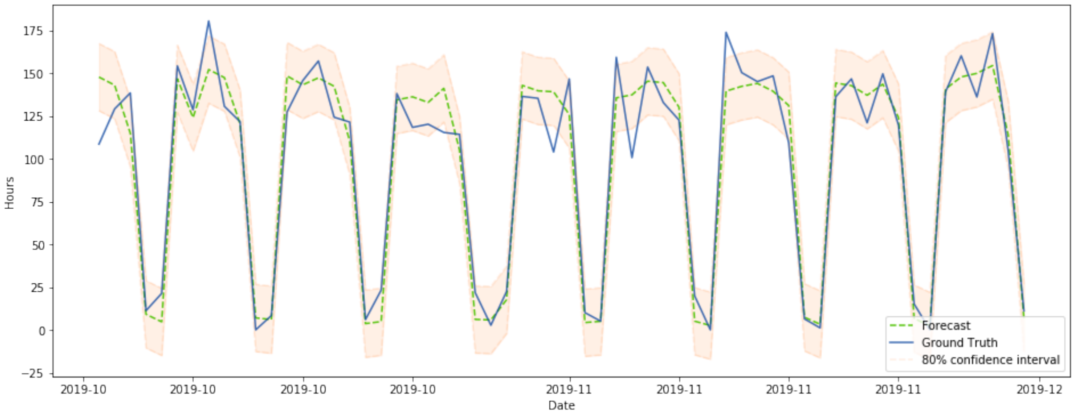


Fig. 3.15 Random Forest forecast with a 80% confidence interval

3.4 Discussion

In this experiment, we found that the most accurate model tested was the Random Forest when used as a multivariable model using past data and date features. It is not unusual to see an ensemble model to perform so well Mei *et al.* (2014). But it was still a bit of a surprise to see it outperforming the models based on neural networks. Recent studies in time series tend to focus on neural network approaches. These studies tend to ignore older methods of machine learning. But the results of this study suggest that models such as Random Forest may outperform them in certain datasets. This is not to say that the neural models performed poorly. They consistently achieved a low error and showed as top performers.

The MLP model performed remarkably well, obtaining the second-best performance in all three experiments. MLP was also more a consistently high performer unlike the CNN LSTM for example, that obtained top performance in the univariable experiment and the worse performance in the multivariable experiment. In the multivariable experiment, the CNN LSTM model obtained roughly the same RMSE as the univariable model, failing to take advantage of the additional data. This is likely due to recurrent neural networks such as LSTMs being difficult to train as explained in Pascanu *et al.* (2013). LSTMs and other recurrent neural network models have to deal with the exploding and vanishing gradients problem which makes the training process more complex than MLP and CNN.

The ARIMA family of models and Exponential smoothing revealed some of their limitations in this experiment. These models make some strong assumptions on the time series behaviour while models like MLP make little assumptions on the data. MLP networks are considered universal approximators (González-Díaz *et al.*, 2019) making them capable of approximating any function. Furthermore, neural networks are notably flexible and can easily support additional data and outputs.

In our results, the models showed improved performance when date features were added to the historical. This suggests that the day when the work is carried appears to have an influence on the results. The influence of weekends seems fairly obvious when analysing the plot, but the difference between weekdays is not so obvious.

Based on this supposition, we can test a null hypothesis that the day of the week does not influence the workload. We test both in the case of the full week (Monday to Sunday) and weekdays (Monday to Friday) using a one way ANOVA test on the dataset. The p-value represents the probability of obtaining test results assuming that the null hypothesis is correct. The smaller the p-value, the stronger the evidence that we should reject the null hypothesis. Assuming a confidence level of 99% the p-value needs to be lower than 0.01 to reject the null hypothesis. In table 3.5 we test these two null hypothesis. That the day of the week has no influence on the workload value when considering the full week (Monday to Sunday). And that the day of the week has no influence on the workload value when considering the weekdays (Monday to Friday).

Table 3.5 One-way ANOVA day of the week

Model	F-value	p-value
full week	354.6336	0.0000
weekdays	6.6632	0.0003

We can see the results in table 3.5. The p-value is lower than 0.01 for both cases. We can reject the null hypothesis for both cases and conclude that the day of the week has an influence on the workload. The workload on Mondays shows a statistically significant difference from the workload on Tuesdays and the other days of the week.

In the results, we can also see that adding weather features to the models tested did not improve performance. All models tested with the additional weather data performed slightly worse when compared to the models using historical data and date features. Suggesting that the weather data was merely introducing noise.

Overall our models improved greatly over the naive baselines, taking advantage of the historical data and date features to provide more accurate forecasts. While not perfect, they are certainly useful. On the final chapter, we present our conclusions.

Chapter 4

Task Duration Classification

4.1 Introduction

In this chapter, we will look at the task of classifying tasks as long or short based on their text description. We explore a wide variety of machine learning models, classical and state of the art. Our models achieve strong performance with a few models being able to outperform human performance. Then we explain how the dataset was created and the data preparation in section 4.2.1. We analyse the data in section 4.2.2. We go over the evaluation metric in section 4.2.3 and the experiment design in section 4.2.4. Finally, we present the results of the experiment followed by a brief discussion of the results.

4.2 Methodology

4.2.1 Dataset Creation

The objective of the research question was to find if it was possible to identify a task duration given a task text description. To this end, we needed to first establish how to approach this task. Treating this problem as a binary classification problem between small tasks and long tasks allow us to test this hypothesis and present it in a familiar context.

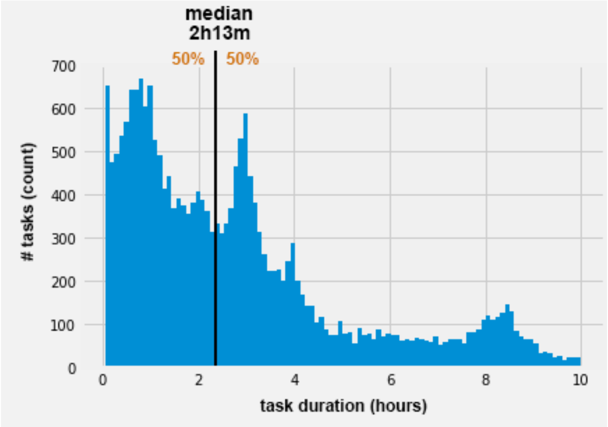


Fig. 4.1 Task duration distribution median point

We created two classes of tasks based on their duration. Short task and long tasks. We could then treat it as a classification problem and employ modern methods for text classification. To decide on the division point we looked at the data distribution of duration the tasks. Since the objective was to evaluate the skill on the models, a balanced distribution of the dataset was a natural choice. In a balanced distribution, each class has an equal number of samples, and no class is favoured. To this end, we found the median point of the task duration as can be seen in fig 4.1. The median point was of 2 hours and 13 minutes. Using this value as the division point the tasks were split into 10803 short tasks and 10803 long tasks.

Having now the base dataset, we cleaned the text to help the classification problem. All words were converted to lowercase, and multiple spaces were removed as is common practice as per Brownlee (2017) and Kowsari *et al.* (2019).

In our experiment, we tested additional text processing techniques that help certain models such as removing stop words and stemming and present the results.

4.2.2 Data Analysis

In order to better understand the data we performed data analysis on the dataset. The dataset is divided into two classes (short and long) with 10803 tasks each.

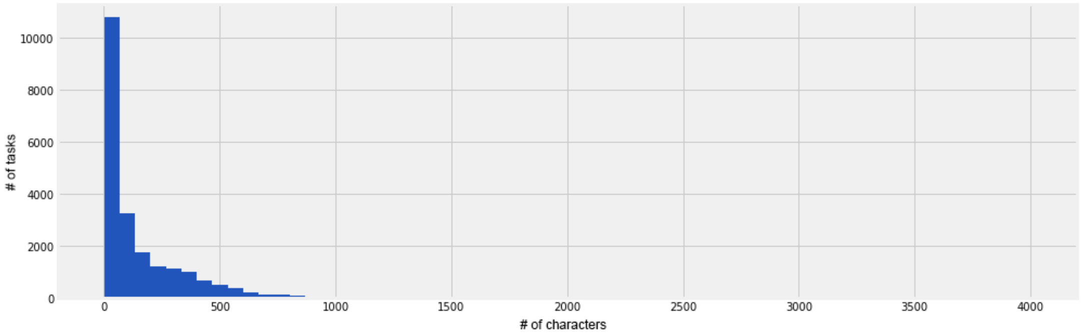


Fig. 4.2 Task text description length (characters)

In fig 4.2 we can see the text description length distribution. Most of the task have a short text description with 99% of the tasks having less than 1000 characters. We show a few samples of task description here:

- *Attend site and carry out emergency lighting test.*
- *Attend site and install new fire alarm system.*
- *Attend site: STORE NOT TRADING the shutter at the entrance to the store will not open, The key will not turn on the outside and the button is not responding on the inside of the store.*
- *Attend site :The main door is broken and not locking. Store has a shutter however someone could put fingers through and open doors etc. Also the threshold is coming up and requires attention. Someone on site till 7pm.*

We also found that in some cases the same description could correspond to short and long classes. Such as in the case of *Attend site - carry out PAT test.* that had 12 short tasks and 9 long tasks. This variation is not unexpected since there is a natural expectation of variation when it comes to human activity. The same task sometimes took longer to perform than others. This pointed to some of the natural ambiguity of the dataset, that a perfect accuracy would not be possible.

In the table 4.1, we analyse the most frequent words for each class. We can observe that there are a lot of common words that appear in both classes, with similar frequencies. But it is also possible to spot a few words that occur more frequently in one of the classes. Such as *ppm* appearing 5050 times in the description of long tasks versus 2074 in short tasks.

To better analyse the word frequency in relation to class we plotted the word frequency

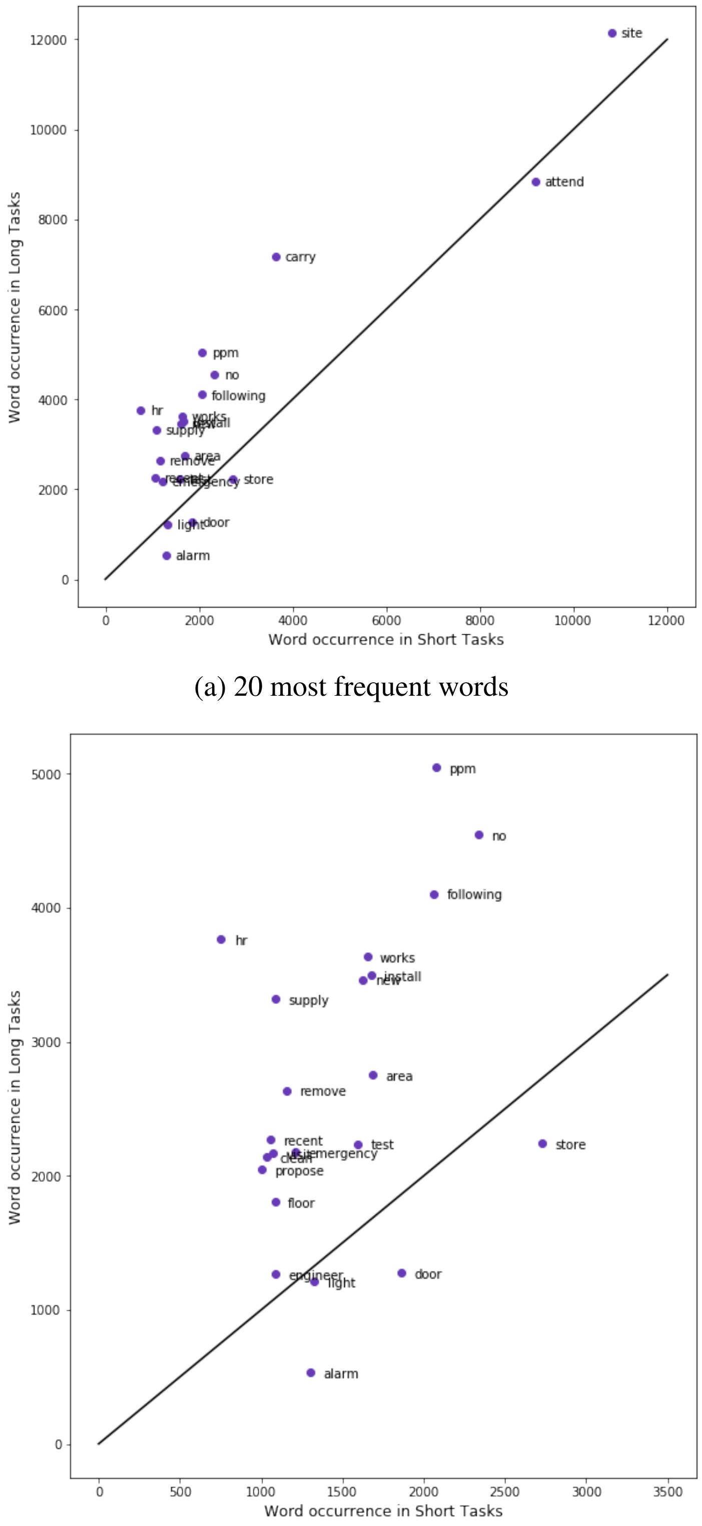
Table 4.1 Most frequent 20 words in each class, after removing stop words

Short		**Long**	
word	count	word	count
site	10804	site	12154
attend	9190	attend	8843
carry	3636	carry	7182
store	2729	ppm	5050
no	2338	no	4548
ppm	2074	following	4107
following	2060	hr	3765
door	1864	works	3636
area	1683	install	3500
install	1677	new	3465
works	1651	supply	3320
new	1625	area	2754
test	1594	remove	2637
light	1326	recent	2268
alarm	1303	store	2242
emergency	1210	test	2238
remove	1156	emergency	2177
supply	1091	visit	2167
engineer	1087	clean	2142
floor	1084	propose	2049

in short tasks relative to the same word occurring in long tasks in figure 4.3. With occurrence in short tasks in the x axis and occurrence in long tasks in the y axis it possible to analyse which word appear more frequently in one class versus the other. To facilitate this analysis we draw a 45°diagonal line. Words that appear closer to this line appear equally in each class. Words that appear far from this line appear more frequently in one class than the other. The further the distance from the line the more accentuated the occurrence difference. This means that a word that are far from the 45°diagonal line occurs more frequently one of the classes. Particularly in 4.3b, where we zoom in to the plot, it is possible to observe an occurrence difference of certain words. The words "ppm" and "hr" clearly appear more frequently in long tasks and the words "alarm" and "door" in short tasks. That suggests that if a word such as "ppm" is present on the task description, it is more likely that the description belongs to a long task than to a short task.

Some of the models implemented in this study represent the text on a word frequency level. For us to be able to confirm that some words are more frequent in one class than the other is a good indication that it will be possible to build skilful models.

Fig. 4.3 Word occurrence plot



(a) 20 most frequent words

(b) Detail view

4.2.3 Evaluation metric

In the evaluation of classification problem we evaluate the performance by comparing the predicted class by the models against the the actual class of the sample. It is common to build a confusion matrix like fig 4.2 (Manning *et al.*, 2008).

We can then calculate performance metrics such as accuracy, precision, recall and F1 score commonly used in text classification problems (Kowsari *et al.*, 2019).

Table 4.2 Confusion matrix

Actual class = short	True Positive (TP)	False Negative (FN)
Actual class = long	False Positive (FP)	True Negative (TN)
	Predicted class = short	**Predicted class = long**

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

$$precision = \frac{TP}{TP+FP}$$

$$recall = \frac{TP}{TP+FN}$$

$$F1 = 2.\frac{precision\,.\,recall}{precision+recall}$$

These metrics assume that we are searching for a specific class (TP). In cases such as our experiment where we are interested in the overall performance of the model it is common to aggregate results to a single value by Macro-Averaging or Micro-Averaging (Kowsari *et al.*, 2019). Macro averaging treats both class equally performing a harmonic mean of the metric for each class. Micro-averaging favours bigger classes (Sokolova and Lapalme, 2009). Since in our experiment both classes have an equal distribution, macro averaging was the method of choice for this dataset.

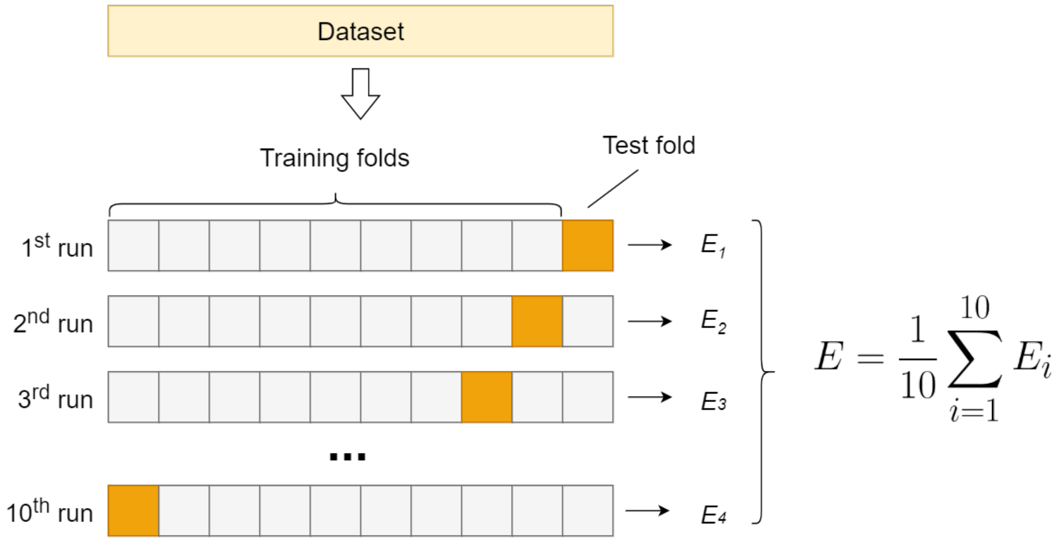


Fig. 4.4 k-fold cross-validation

4.2.4 Experiment design

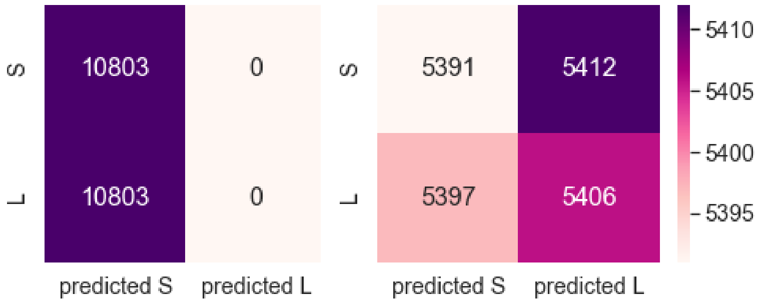
We evaluate the classification models on the dataset created in this study. We use the k-fold cross-validation which is considered the gold standard for text classification (Feldman and Sanger, 2006) , (Ingersoll *et al.*, 2013).

The tasks were shuffled as to avoid the influence of external factors related to the order that they were created. We performed a 10-fold cross-validation with stratified data. This means that we split the dataset into ten folds, each with the same distribution of data as the original dataset. Then we trained each model ten times on nine folds and tested on one fold, rotating the folds each loop. A sample of this can be seen in fig 4.4. Text classification can be a computationally expensive task. Particularly in cases where the samples can be in the thousands or more, as is the case of this dataset. So in addition to the performance metrics, we decided to measure the training times for each model and present them in table 4.18.
The results presented use Macro-averaged results for precision, recall and F1.

4.3 Results

4.3.1 1 Rule baseline

The 1 Rule (1R) model a naive model is commonly used to establish a baseline for a classification problem. This model works by predicting the most common class for every sample of the dataset, with no regards to each sample information. In our case it predicts every sample as a short task (see Fig. 4.5a). The 1 rule is useful to establish if more complex models show skill and are able to leverage the information of each sample or merely the statistical distribution of the dataset. It achieves a accuracy of 50.0% and an F1 macro score of 33.3%.



(a) 1R (b) Random

Fig. 4.5 Confusion matrices of naive models

4.3.2 Random class baseline

The random class model is another naive model. It predicts a random class for each sample. This is a model without skill only useful to establish a baseline for the dataset. It achieves an accuracy of 50.0% and an F1 score of 50.0%.

Table 4.3 Naive models results

Model	Accuracy	Precision	Recall	F1
1R	50.0%	25.0%	50.0%	33.3%
Random	50.0%	50.0%	50.0%	50.0%

The results for both the naive models can be seen in table 4.3. If any model trained shows similar results we can say it has no skill, since its predictions are as good as random. For this dataset, only models that achieve more than 50.0% accuracy and F1 score show skill at the task of classification and are better than a random guess.

4.3.3 Human Performance

To establish a human performance baseline, we utilized the scheduling data created by human operators. We obtained data for each task from the scheduling software and extracted the expected start time and expected end time for each task as estimated by the person scheduling. With this information, it was possible to calculate the expected duration for each task than could then be classified into the short and long classes. Comparing this to the actual duration class for each task, we can calculate the performance metrics. The results are shown in table 4.4. In essence, these results reflect the performance of the human experts in the work environment. How accurate was their estimation at the time that they were scheduling the tasks.

Table 4.4 Human Performance results

Model	Accuracy	Precision	Recall	F1
Human Performance	74.1%	74.2%	74.2%	74.1%

4.3.4 Naive Bayes

Naive Bayes is a classic probabilistic classifier based on Bayes theorem (Bayes, 1763). It is a commonly used model for baseline classification performance. We decided to create and train four variations of text representation with the Naive Bayes classifier. We test two text representations (bag-of-words and tf-idf) and the use of text pre-processing (removing stop words, stemming) compared to no pre-processing.
NB - BOW refers to the Naive Bayes classifier with the bag-of-words text model.
NB - BOW - P refers to the Naive Bayes classifier with the bag-of-words text model with pre-processing.
NB - TF-IDF refers to the Naive Bayes classifier with the tf-idf text model.

NB - TF-IDF - P refers to the Naive Bayes classifier with the tf-idf text model with pre-processing.
The results obtained can be seen in table 4.5.

Table 4.5 Naive Bayes models results

Model	Accuracy	Precision	Recall	F1
NB - BOW	71.8%	71.9%	71.8%	71.8%
NB - BOW - P	72.3%	72.3%	72.3%	72.3%
NB - TF-IDF	72.9%	73.2%	72.9%	72.9%
NB - TF-IDF - P	72.2%	72.6%	72.2%	72.1%

The best performance of the Naive Bayes models was the NB - TF-IDF model with an accuracy of 72.9% and an F1 score of 72.9%. This model uses tf-idf as text representation and no text pre-processing.

4.3.5 KNN

The k-nearest neighbors (KNN) algorithm was first introduced in Altman (1992). We created a model using KNN as a classifier in conjunction with the tf-idf text representation and no text pre-processing. It achieved an accuracy of 72.1% and an F1 score of 72.1%. Of note is how balanced the confusion matrix for KNN results was, as can be seen in fig 4.6.

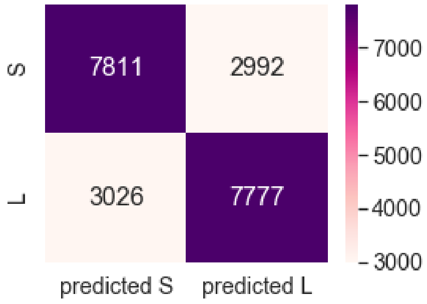


Fig. 4.6 KNN - confusion matrix

Table 4.6 KNN results

Model	Accuracy	Precision	Recall	F1
KNN	72.1%	72.1%	72.1%	72.1%

4.3.6 Gradient boosting

We created a model using Gradient boosting (GB) (Friedman, 2001) as the classifier and tf-idf as text representation with no text pre-processing. It obtained an accuracy of 73.8% and an F1 score of 73.9%.

Table 4.7 Gradient boosting results

Model	Accuracy	Precision	Recall	F1
GB	73.8%	73.9%	73.8%	73.8%

4.3.7 XGBoost

XGBoost (Chen and Guestrin, 2016) is a high-performance implementation of gradient boosted decision trees. For our experiment we created a model using XGBoost as the classifier and tf-idf as text representation with no text pre-processing.

Table 4.8 XGBoost results

Model	Accuracy	Precision	Recall	F1
XGBoost	73.1%	73.3%	73.1%	73.1%

4.3.8 SVM

The SVM (Boser *et al.*, 1992) model that we created for this experiment was trained with tf-idf as text representation and no text pre-processing. It achieved an accuracy of 74.6% and an F1 score of 74.5%. This was the first model tested to perform higher than human performance (F1 score 74.1%). Of note was the long training time in comparison to the other models in the experiment, being by far the slowest model tested. We show the comparison of training time in table 4.18.

Table 4.9 SVM results

Model	Accuracy	Precision	Recall	F1
SVM	74.6%	74.7%	74.6%	74.5%

4.3.9 CNN models and transfer learning

In this section we started testing the use of neural networks approaches to the task classification problem. The use of neural networks in NLP problems is a very active field.

One of the biggest innovations in recent years is the use of high dimensional, continuous vectors to represent the statistical probability of a word in a given context. This results in words with semantically similar value to be close, as well as other interesting proprieties such as the distance between words representing their relations. This method was first popularized by the method Word2Vec in the work of Mikolov *et al.* (2013a). Another popular word vector representation is GloVe introduced in Pennington *et al.* (2014). One of the advantages of these models is the availability of pre-trained models that have been trained in a large corpus of text to create a multi-dimensional representation of words. This trained models then can be then used by the method of transfer learning to augment neural networks models given them an improved starting point where the model merely has to learn the task at hand and not the relation between words.

While using pre-trained word vector models is the general advice, it is possible to begin with untrained word vectors and train the word vectors at the same time as the classification task. This has the disadvantage of the word vectors model being limited to the vocabulary in the training corpus and thus not having a good representation if it encounters new words when deployed after training. The advantage is that the word vector representation will be optimized for the given task since the error back-propagates from the classification loss to the word vectors. It is a balance between a more general model or one that is over-fitted to the task at hand.

Convolutions neural networks (CNN) are commonly used as a model for text classification (Kowsari *et al.*, 2019) often achieving state of the art results. We used a CNN neural network with a similar architecture with several variations for the word vector representation. With untrained word vectors, with a pre-trained Word2Vec model and with a GloVe pre-trained model. We then compare the results on table 4.10. For the implementation of the different CNN models, we used the Keras framework (Chollet *et al.*, 2015) in conjunction with Tensorflow (Abadi *et al.*, 2015).

Untrained Word Vectors

For the first CNN model, we used an untrained word embedding layer with 300 dimensions followed by a convolution neural network. The error will propagate from the CNN to the word vectors, training them at the same time as the CNN. This model is referenced as CNN - UWV. After optimizing hyper-parameters, the CNN - UWV model achieved an accuracy of 73.8% and an F1 score of 73.8%.

Word2Vec

For this classification model, we used the Word2Vec model previously trained as a 300 dimensional vectors representation. The Word2Vec used was trained on a *Google News* dataset containing approximately 100 billion words.
We refer to this model as CNN - Word2Vec. This model achieved an accuracy of 73.4% and an F1 score of 73.4%.

GloVe

The GloVe model used for this model was pre-trained on the *Wikipedia 2014* and *Gigaword 5* datasets, for a total of approximately 6 Billion words. It represents words as a 300 dimension vector.
We refer to this model as CNN - GloVe. This model achieved an accuracy of 74.0% and an F1 score of 74.0%.

4.3.10 Neural Networks Architectures and GloVe

The GloVe word representation model provided the best performance of word vectors in the previous section. We decided then to explore different Neural Networks

Table 4.10 CNN - Different Word Vector models

Model	Accuracy	Precision	Recall	F1
CNN - UWV	73.8%	73.8%	73.8%	73.8%
CNN - Word2Vec	73.4%	73.4%	73.4%	73.4%
CNN - GloVe	74.0%	74.1%	74.0%	74.0%

Architectures. In this section, we present the results for different neural networks architectures tested using the GloVe model.

Multilayer Perceptron (MLP)

Multilayer Perceptron (MLP), sometimes called Feed Forward Neural Network is one of the simpler neural network architectures. Using the GloVe language model it achieved an accuracy of 72.1% and an F1 score of 72.1%.

Multi-Headed CNN

Building on our CNN model, we built a Multi-Headed convolution neural network. By creating three convolution heads, the model is capable of exploring different convolution sizes, potentially exploring relations between words that are further apart. The diagram for this model can be seen in figure 4.7. This model slightly improved the accuracy of the simpler CNN model, achieving 74.1% and an F1 score of 74.0%.

LSTM

The LSTM model is commonly used to model sequences of data, such as words in a sentence. Our LSTM - GloVe achieved an accuracy if 73.9% and an F1 score of 73.9% a result close to the CNN model.

CNN LSTM

This model combines convolution layers with LSTM layers for this classification task . We refer to this model as CNN LSTM - GloVe. It achieved an accuracy of 73.7% and an F1 score of 73.7%.

Table 4.11 Different Neural Networks Architectures models and GloVe

Model	Accuracy	Precision	Recall	F1
CNN - GloVe	74.0%	74.1%	74.0%	74.0%
MLP - GloVe	72.1%	72.1%	72.1%	72.1%
CNN MH - GloVe	74.1%	74.2%	74.1%	74.0%
LSTM - GloVe	73.9%	74.1%	73.9%	73.9%
CNN LSTM - GloVe	73.7%	73.8%	73.7%	73.7%

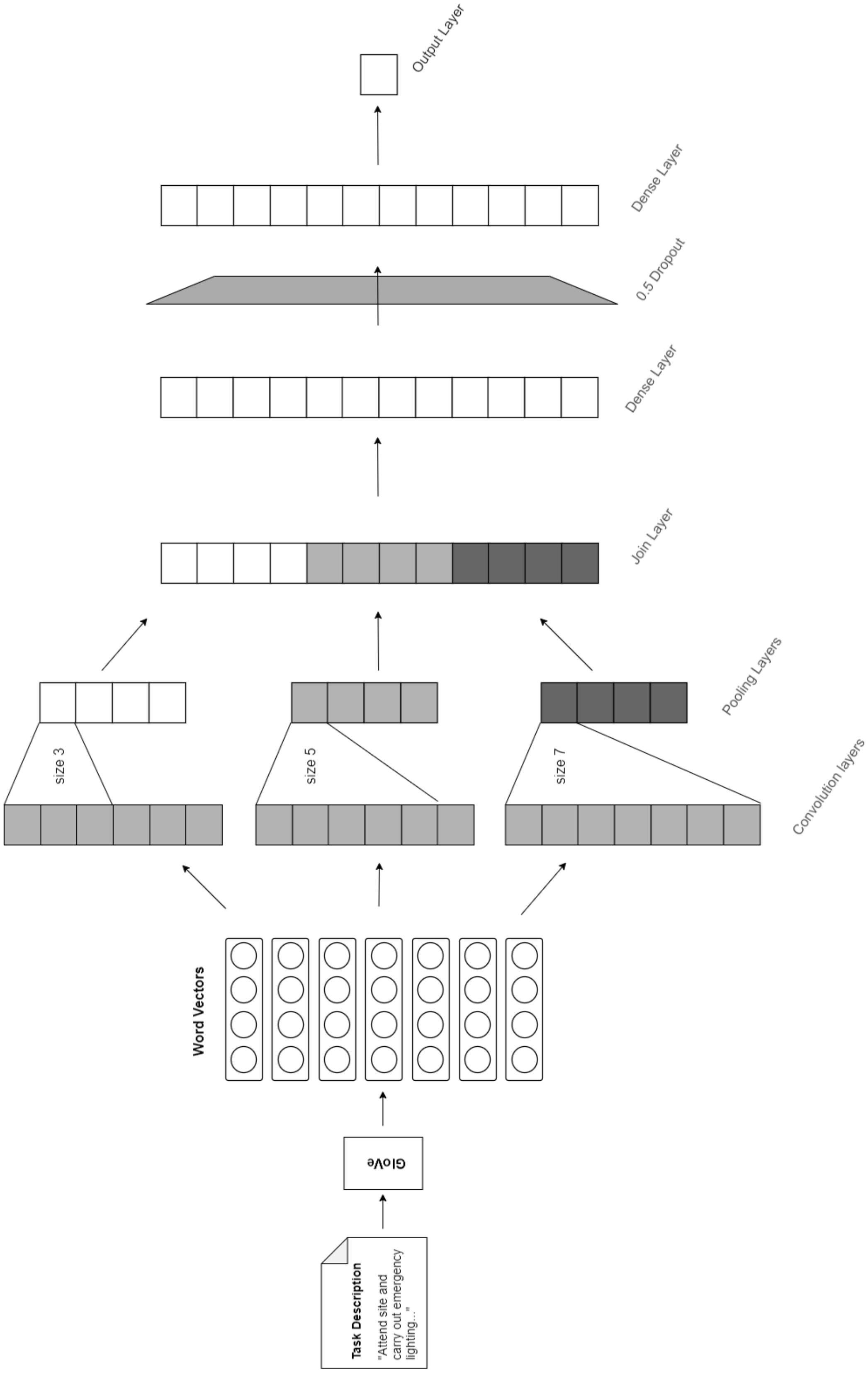


Fig. 4.7 Multi-Headed CNN GloVe model diagram

4.3.11 ELMO

ELMO is a model that uses deep contextualized word representation to represent words. Elmo models both syntax and semantic word value, and how these it can vary across context, it was first introduced by Peters *et al.* (2018). In our experiment, we used an ELMO embedding layer with a dimension of 1024. ELMO obtained an accuracy of 72.0% and an F1 score of 71.9%, one of the lowest performances on this study. This was somewhat surprising, but it is possible that the dataset of approximately 20000 samples was just not big enough to fully take advantage of ELMO or merely a poor fit to the current problem and dataset.

Table 4.12 ELMO results

Model	Accuracy	Precision	Recall	F1
ELMO	72.0%	72.4%	72.0%	71.9%

4.3.12 BERT

BERT has been one of the top models of choice for NLP tasks in recent years since it was introduced in Devlin *et al.* (2019). BERT claims state of the art performance in various publicly available datasets at the time of the writing. This can be seen on the state of the art tracker created by the website Paperswithcode (2020). In our experiment BERT outperformed the models presented so far with an accuracy of 74.6% and an F1 score of 74.6%.

Table 4.13 BERT results

Model	Accuracy	Precision	Recall	F1
BERT	74.6%	74.9%	74.6%	74.6%

4.3.13 XLNet

XLNet was introduced in Yang *et al.* (2019). XLNet is another great model based on the transformer architecture that showed strong performance on this experiment with an accuracy of 74.6% and an F1 score of 74.6%.

Table 4.14 XLNet results

Model	Accuracy	Precision	Recall	F1
XLNet	74.6%	74.7%	74.6%	74.6%

4.3.14 ROBERTA

The ROBERTA model is a variation on the BERT model introduced in Liu *et al.* (2019). The name stands for *Robustly Optimized BERT Pretraining Approach* and as the name implies it proposes some modification to the original BERT model to improve performance. In our experiment ROBERTA performed slightly worse than the BERT model with an accuracy of 74.2% and an F1 score of 74.2%.

Table 4.15 ROBERTA results

Model	Accuracy	Precision	Recall	F1
ROBERTA	74.2%	74.4%	74.2%	74.2%

4.3.15 ALBERT

The model ALBERT was introduced in Lan *et al.* (2020) mainly to deal with the high memory requirements and slow training speed of BERT. In the original paper this model was able to outperform BERT while being 1.7 times faster. In our experiment ALBERT performed slightly worse than the BERT model, but trained 1.7 faster than BERT as can be seen in table 4.18.

Table 4.16 ALBERT results

Model	Accuracy	Precision	Recall	F1
ALBERT	74.1%	74.4%	74.1%	74.0%

4.3.16 fastText

The fastText model was created with efficiency in mind by the Facebook AI Research group. For this it employs a combination to methods, a "Bag of Tricks" as the authors call it. The fastText model aims to provide an efficient and impressive performance in the task of task classification that sometimes rival state of the art deep learning classifiers while having faster training times (Joulin *et al.*, 2017).

For our experiment we created two versions of this model. The first version uses unigrams (single words) as inputs. The second version additionally uses bigrams (the combination of two words in a sequence). The *fastText - bigrams* model achieves the best performance of all models tested with an accuracy of 74.9% and an F1 score of 74.7%.

Of note is the slight imbalanced confusion matrix as seen in fig 4.8. This model predicted 12435 tasks as short and 9171 as long. This contrasts with the 50%/50% distribution of the actual classes. It is possible that this merely reflects the characteristics of the dataset.

Table 4.17 fastText results

Model	Accuracy	Precision	Recall	F1
fastText - unigrams	72.6%	72.7%	72.6%	72.6%
fastText - bigrams	74.9%	75.5%	74.9%	74.7%

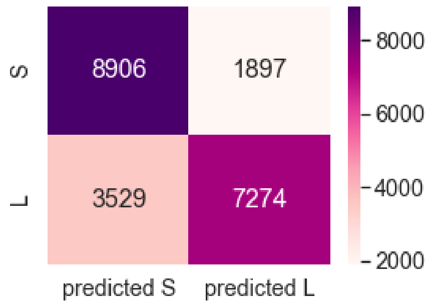


Fig. 4.8 fastText - bigrams - confusion matrix

4.4 Results Compiled

This section compiles the results for the experiment of task duration classification task. We present all the results in the table 4.18 and the graph 4.9 for easy comparison. All the models here presented were tested with the 10-fold cross-validation method. The training time presented correspond to the total time of the 10 training cycles on the same computer hardware.

Table 4.18 Models Results Compiled

Model	Accuracy	Precision	Recall	F1	Training Time (s)
1R	50.0%	25.0%	50.0%	33.3%	0.1s
Random	50.0%	50.0%	50.0%	50.0%	0.1s
NB - BOW	71.8%	71.9%	71.8%	71.8%	20.1s
NB - BOW - P	72.3%	72.3%	72.3%	72.3%	20.3s
NB - TF-IDF	72.9%	73.2%	72.9%	72.9%	56.6s
NB - TF-IDF - P	72.2%	72.6%	72.2%	72.1%	50.4s
KNN	72.1%	72.1%	72.1%	72.1%	5876.8s
XGBoost	73.1%	73.3%	73.1%	73.1%	3256.8s
GB	73.8%	73.9%	73.8%	73.8%	14532.8s
SVM	74.6%	74.7%	74.6%	74.5%	40982.2s
CNN - UWV	73.8%	73.8%	73.8%	73.8%	142.9s
CNN - Word2Vec	73.4%	73.4%	73.4%	73.4%	94.0s
CNN - GloVe	74.0%	74.1%	74.0%	74.0%	106.8s
CNN MH - GloVe	74.1%	74.2%	74.1%	74.0%	242.6s
CNN LSTM - GloVe	73.7%	73.8%	73.7%	73.7%	3055.7s
MLP - GloVe	72.1%	72.1%	72.1%	72.1%	266.5s
LSTM - GloVe	73.9%	74.1%	73.9%	73.9%	8305.3s
ELMO	72.0%	72.4%	72.0%	71.9%	9612.1s
BERT	74.6%	74.9%	74.6%	74.6%	5064.4s
XLNet	74.6%	74.7%	74.6%	74.6%	6177.2s
ROBERTA	74.2%	74.4%	74.2%	74.2%	4996.2s
ALBERT	74.1%	74.4%	74.1%	74.0%	3036.4s
fastText - unigrams	72.6%	72.7%	72.6%	72.6%	351.6s
fastText - bigrams	**74.9%**	**75.5%**	**74.9%**	**74.7%**	1016.7s
Human Performance	74.1%	74.2%	74.2%	74.1%	-

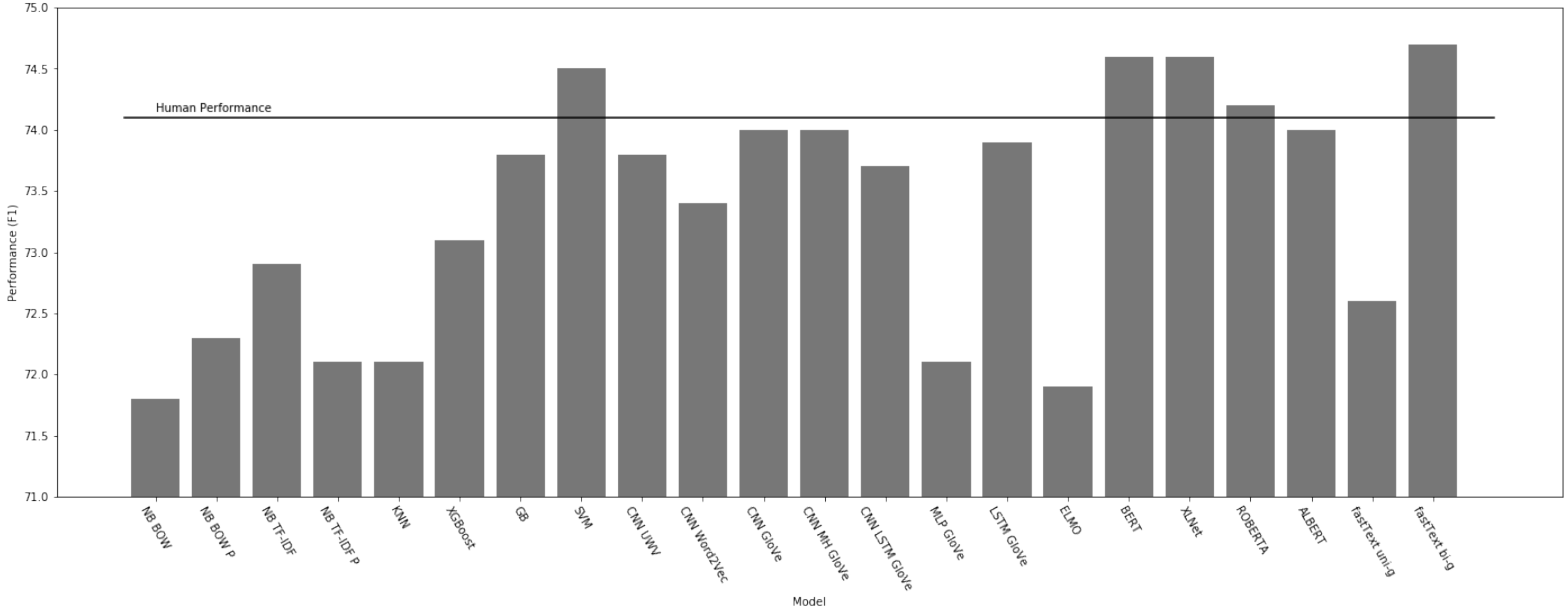


Fig. 4.9 Text Classification Models Performance
The y scale is 71-75% instead of 0-100% for better distinction of the difference between models.
All models had F1 scores in this range.

4.5 Discussion

This experiment tested a large number of models from classical to state of the art. It is not often that the best model is also one of the most efficient, but this was the case in this experiment. The model fastText - bigrams proved not only the most accurate but also one with the lowest training times.

In general the results matched with the progression of models in the field of natural language processing. In the more classical word representation models, TF-IDF outperformed the simpler BOW model when used with Naive Bayes. It was interesting to find that TF-IDF offered better performance without the additional text preparation of removing stop words and stemming.

SVM was one of the surprises outperforming some of the more recent neural network models. Unfortunately, this came at the cost of high training time. SVM scales poorly with high dimensional representations and a large number of samples, as is the case of our dataset. SVM has a computational complexity of $O(n_{features} \times n_{samples}^{2})$ to $O(n_{features} \times n_{samples}^{3})$ as per Pedregosa *et al.* (2011). This causes SVM to scale poorly as the number of samples increase, making it unpractical for problems that have samples in the thousands or millions. While this result was unexpected since most modern approaches in NLP are based in neural networks models, this is not an isolated case. Some research like Van-Tu and Anh-Cuong (2016) were able to achieve a good performance using SVM in an NLP problem. Young *et al.* (2017) also shows a strong performance by SVM when compared with modern neural networks models in a part of speech tagging problem.

Of the models using neural networks and word vectors as representation, the models CNN - GloVe and CNN MH - GloVe proved the most accurate. They also offered an excellent training performance, training an order of magnitude faster than the best model found. It is interesting that the pre-trained GloVe model outperformed the pre-trained Word2Vec. The Word2Vec was trained on a larger corpus (100 billion words) based on Google News articles while the GloVe model trained on a smaller corpus (6 billion words) of Wikipedia articles and Gigaword. This may be due to the GloVe model being superior or due to the training corpus of Wikipedia articles being able to provide more relevant language information.

The models that were based on the transformer architecture *(BERT, XLNet, ROBERTA, ALBERT)* all achieved strong accuracy values, with BERT and XLNet obtaining similar performance. This seems to suggest that they are superior to the previous dominant approach of using word vectors and neural networks for text classification.

The best model fastText - bigrams model was able to outperform our human performance baseline, which was an impressive feat.

These results promise a great future for the field of natural language processing. The amount of data in text form is vast and growing. Processing such large amounts of data require an automated way. It is encouraging to see that NLP models are now able to achieve or even surpass human-level performance in certain NLP tasks. Such was the case in this study and in other recent works, like in the work of Lan *et al.* (2020) on the Stanford SQuAD2.0 dataset.

Chapter 5

Conclusion

This research investigated two main research questions:
- *How can historical data be used to forecast the workload for the next day?*
- *How can a task text description be used to identify short tasks and long tasks?*
Based on our results we were able to answer both questions with positive results.

By testing naive, classical forecasting models and modern developments in the area of forecasting we were able to create models capable of closely capturing the behaviour of the workload time series in our dataset. We trained and tested machine learning models and deep learning models with a dataset created from previous time steps and added additional information about date and weather. By utilizing the historical data and date information it was possible to greatly improve on our baselines and provide an accurate forecast. The best model found was the multivariable Random Forest with date features, obtaining an RMSE error of 15.215. To contextualize this value, the median day on the dataset has a workload of 114.65 hours and the day of the highest workload has 206.08 hours. This means that for a typical day this model would be able to, on average, predict the workload value with approximately 13% error. And for high workload days with an error of less than 10%.

On the text description task, we were able to obtain great results for the dataset tested that surpassed the performance of human experts. We used machine learning and deep learning models to predict tasks as short or long in duration.

We explored different forms of text representations and different classifiers, from classic machine learning to state of the art approaches. Given the results presented in the literature, we expected the fastText model to be one of the top performers. But to outperform more recent and complex models such as BERT and XLNet was an interesting result. BERT, XLNet and other transformer based architectures have been dominating the state of the art in NLP tasks. They were still among the top performers, closely behind our best model. This shows how strong the transformer architecture is for NLP tasks.

These results reinforce the idea of how important it is to test more than a few models. No single model is going to provide the optimal solution for every problem, as argued by Wolpert and Macready (1997).

One of the contributions of this study is the performance comparison of classical time series forecasting models with modern machine learning and deep learning methods. We showed that on the dataset studied, machine learning models outperformed the classical models that are still widely popular in the field (Hyndman and Athanasopoulos, 2018).

This study also showed that using deep learning text classification methods can be used to estimate a task duration in the maintenance industry, in our case, outperforming human performance at the task. It also contributes a performance review of the state of the art text classification models on a real-world industry dataset.

As a final contribution, we show with our results the remarkable flexibility of neural network models. Neural networks achieved great results in both the forecasting and text classification experiments. These are two very different tasks and neural network models showed great results. These are important contributions that have both industry and academic importance.

For the industry, the company that provided the data for this study can in the future integrate these models into their workflow. The text classification model created can be used to automatically classify incoming task requests, facilitating the work of the human operator. The forecasting model can provide a forecast of the next day workload on a daily basis, as well as the confidence interval for better resource allocation.

As for the academic importance, our results showed that machine learning models provided improved results over classical time series models in the dataset tested. Similar time series problems may benefit from similar models.

The results of the task classification problem showed that there is a natural ambiguity and stochasticity in trying to predict the duration of a task from natural language, at least in this dataset. The process may not be completely deterministic as is often the case when studying human activity. But state of the art models were able to achieve results that outperformed our human baseline performance.

The results presented are limited to the dataset tested, based on one company's data. They may not necessarily apply to different datasets. Future work could expand the experiments with different datasets.

In future work for this project, we are planning to integrate the models created into scheduling software used in the industry. How to best integrate, distribute and keep the models up to date. After integration, additional work can be made by measuring model performance in the live application and how real-world performance compares to the results obtained in this study.

While this study concerns itself with the workload in the maintenance industry, any scheduling software has the potential to benefit from the integration of similar models to the ones presented in this study. The models employed are easily adapted to other industries since there is no handcrafted knowledge specific to the maintenance industry. The models merely need to be retrained in datasets specific to the application. This study trained and tested an extensive range of models. It serves as a great source to select models likely to succeed in future applications similar in nature.

Our results suggest that models that try to imbue human knowledge or make assumptions on the data can be outperformed by models that are more generic in nature. Given enough data and computing power, a general model can find their own rules and methods to reason about the data. Richard Sutton, researcher at DeepMind and one of the fathers of Reinforcement Learning, argues so in *The Bitter lesson* (Sutton, 2019). Sutton makes the argument that general-purpose methods that can leverage computation power and have the ability to scale are ultimately the most effective. Trying to leverage human knowledge of a problem will be limited by human understanding. It was so in chess and more recently in Go (Silver *et al.*, 2016).

In today's world, more data is generated every day than all the data generated in one year 15 years ago (Guo *et al.*, 2014). The need for automatic processing of data is growing, and with it, more developments in research.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. (2015) TensorFlow: Large-scale machine learning on heterogeneous systems software available from tensorflow.org

Albertetti, F. and Ghorbel, H. (2020) Workload prediction of business processes – an approach based on process mining and recurrent neural networks

Aldor-Noiman, S., Feigin, P.D. and Mandelbaum, A. (2009) Workload forecasting for a call center: Methodology and a case study *The Annals of Applied Statistics* **3**(4), p. 1403–1447

Altman, N. (1992) An introduction to kernel and nearest neighbor nonparametric regression

Amari, S.I. (1993) Backpropagation and stochastic gradient descent method *Neurocomputing* **5**(4), pp. 185 – 196

Bayes, T.P.R. (1763) An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, communicated by mr. price, in a letter to john canton, ma. and f.r.s. (53)

Biebricher, P., Fuhr, N., Lustig, G., Schwantner, M. and Knorz, G. (1988) The automatic indexing system air/phys - from research to applications in: *Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* SIGIR '88 p. 333–342 Association for Computing Machinery, New York, NY, USA

Boser, B.E., Guyon, I.M. and Vapnik, V.N. (1992) A training algorithm for optimal margin classifiers in: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* COLT '92 p. 144–152 Association for Computing Machinery, New York, NY, USA

Box, G.E.P. and Jenkins, G. (1970) *Time Series Analysis, Forecasting and Control* Holden-Day, Inc., USA

Breiman, L. (2001) Random forests *Mach. Learn.* **45**(1), p. 5–32

Brownlee, J. (2017) *Deep Learning for Natural Language Processing* Machine Learning Mastery 1st ed.

Brownlee, J. (2018a) *Deep Learning for Time Series Forecasting* Machine Learning Mastery 1st ed.

Brownlee, J. (2018b) *Introduction to time series forecasting with python* Machine Learning Mastery 1st ed.

Chen, T. and Guestrin, C. (2016) XGBoost: A scalable tree boosting system in: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* KDD '16 pp. 785–794 ACM, New York, NY, USA

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H. and Bengio, Y. (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation *CoRR* **abs/1406.1078**

Chollet, F. (2017) *Deep Learning with Python* Manning Publications Co., USA 1st ed.

Chollet, F. *et al.* (2015) Keras https://keras.io

Crawford, S.L., Fung, R.M., Appelbaum, L.A. and Tong, R.M. (1991) Classification trees for information retrieval in: *ML* pp. 245–249

Dalal, M. and Zaveri, M. (2011) Automatic text classification: A technical review *International Journal of Computer Applications* **28**

De Gooijer, J.G. and Hyndman, R. (2006) 25 years of time series forecasting *International Journal of Forecasting* **22**, pp. 443–473

Devlin, J., Chang, M.W., Lee, K. and Toutanova, K. (2019) Bert: Pre-training of deep bidirectional transformers for language understanding *ArXiv* **abs/1810.04805**

Drucker, H., C, C., Kaufman, L., Smola, A. and Vapnik, V. (2003) Support vector regression machines *Advances in Neural Information Processing Systems* **9**

Feldman, R. and Sanger, J. (2006) *Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data* Cambridge University Press, USA

Fliess, M., Join, C., Bekcheva, M., Moradi, A. and Mounier, H. (2019) Easily implementable time series forecasting techniques for resource provisioning in cloud computing

Friedman, J.H. (2001) Greedy function approximation: A gradient boosting machine. *Ann. Statist.* **29**(5), pp. 1189–1232

Fu, R., Zhang, Z. and Li, L. (2016) Using lstm and gru neural network methods for traffic flow prediction in: *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)* pp. 324–328

Fukushima, K. (1980) Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position *Biological cybernetics* **36**(4), p. 193—202

Gamboa, J.C.B. (2017) Deep learning for time-series analysis

Gergonne, J. (1974) The application of the method of least squares to the interpolation of sequences *Historia Mathematica* **1**(4), pp. 439 – 447

González-Díaz, R., Gutiérrez-Naranjo, M.A. and Paluzo-Hidalgo, E. (2019) Two-hidden-layer feedforward neural networks are universal approximators: A constructive approach *CoRR* **abs/1907.11457**

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep Learning* MIT Press http://www.deeplearningbook.org

Gulli, A. and Pal, S. (2017) *Deep Learning with Keras* Packt Publishing

Guo, H., Wang, L., Chen, F. and Liang, D. (2014) Scientific big data and digital earth *Chinese Science Bulletin (Chinese Version)* **59**, p. 1047

Hecht-Nielsen (1989) Theory of the backpropagation neural network in: *International 1989 Joint Conference on Neural Networks* pp. 593–605 vol.1

Heckerman, D., Horvitz, E., Sahami, M. and Dumais, S. (1998) A bayesian approach to filtering junk e-mail in: *AAAI Workshop on Learning for Text Categorization*

Hochreiter, S. and Schmidhuber, J. (1997) Long short-term memory *Neural Computation* **9**(8), pp. 1735–1780

Hyndman, R. and Athanasopoulos, G. (2018) *Forecasting: Principles and Practice* OTexts, Australia 2nd ed.

Indurkhya, N. and Damerau, F.J. (2010) *Handbook of Natural Language Processing* Chapman amp; Hall/CRC 2nd ed.

Ingersoll, G.S., Morton, T.S. and Farris, A.L. (2013) *Taming Text: How to Find, Organize, and Manipulate It* Manning Publications Co., USA

Institute, P.M. (2013) *A Guide to the Project Management Body of Knowledge* Project Management Institute

Joulin, A., Grave, E., Bojanowski, P. and Mikolov, T. (2017) Bag of tricks for efficient text classification in: *EACL*

Karevan, Z. and Suykens, J.A.K. (2018) Spatio-temporal stacked lstm for temperature prediction in weather forecasting

Karner, G. (2010) Resource estimation for objectory projects

King, W.I. (1912) *The Elements of Statistical Method* The Macmillan Company

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., Brown, D., Id, L. and Barnes (2019) Text classification algorithms: A survey *Information (Switzerland)* **10**

Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012) Imagenet classification with deep convolutional neural networks in: *Advances in neural information processing systems* pp. 1097–1105

Laberge, g., Shirzad, S., Diehl, P., Kaiser, H., Prudhomme, S. and Lemoine, A.S. (2019) Scheduling optimization of parallel linear algebra algorithms using supervised learning *2019 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)*

Lai, G., Chang, W.C., Yang, Y. and Liu, H. (2017) Modeling long- and short-term temporal patterns with deep neural networks

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. and Soricut, R. (2020) Albert: A lite bert for self-supervised learning of language representations *ArXiv* **abs/1909.11942**

Lewis, D., Info, C., Studies, L. and Ringuette, M. (1996) A comparison of two learning algorithms for text categorization *Third Annual Symposium on Document Analysis and Information Retrieval*

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V. (2019) Roberta: A robustly optimized bert pretraining approach *ArXiv* **abs/1907.11692**

Manayaga, M.C.E. and Ceballos, R.F. (2019) Forecasting the remittances of the overseas filipino workers in the philippines *arXiv: Applications*

Manning, C.D., Raghavan, P. and Schütze, H. (2008) *Introduction to Information Retrieval* Cambridge University Press, USA

Mehtab, S. and Sen, J. (2020) A time series analysis-based stock price prediction using machine learning and deep learning models

Mei, J., He, D., Harley, R., Habetler, T. and Qu, G. (2014) A random forest method for real-time price forecasting in new york electricity market vol. 2014 pp. 1–5

Mikolov, T., Chen, K., Corrado, G.S. and Dean, J. (2013a) Efficient estimation of word representations in vector space *CoRR* **abs/1301.3781**

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J. (2013b) Distributed representations of words and phrases and their compositionality

Mikolov, T., Yih, W.t. and Zweig, G. (2013c) Linguistic regularities in continuous space word representations in: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* pp. 746–751 Association for Computational Linguistics, Atlanta, Georgia

Olya, M.H., Zhu, D. and Yang, K. (2018) Multi-task prediction of patient workload

Paperswithcode (2020) https://paperswithcode.com/

Pascanu, R., Mikolov, T. and Bengio, Y. (2013) On the difficulty of training recurrent neural networks in: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28* ICML'13 p. III–1310–III–1318 JMLR.org

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011) Scikit-learn: Machine learning in Python *Journal of Machine Learning Research* **12**, pp. 2825–2830

Pennington, J., Socher, R. and Manning, C. (2014) Glove: Global vectors for word representation vol. 14 pp. 1532–1543

Perone, G. (2020) Arima forecasting of covid-19 incidence in italy, russia, and the usa

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L. (2018) Deep contextualized word representations in: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* pp. 2227–2237 Association for Computational Linguistics, New Orleans, Louisiana

Reilly, J. (2019) *Applied Statistics* Statistical Solutions

Ruder, S. (2020) nlpprogress.com https://nlpprogress.com/

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1988) *Learning Representations by Back-Propagating Errors* p. 696–699 MIT Press, Cambridge, MA, USA

Sainath, T.N., Vinyals, O., Senior, A. and Sak, H. (2015) Convolutional, long short-term memory, fully connected deep neural networks in: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* pp. 4580–4584

Samuels, A. and Mcgonical, J. (2020) Sentiment analysis on customer responses *ArXiv* **abs/2007.02237**

Sansom, D., Downs, T. and Saha, T. (2003) Evaluation of support vector machine based forecasting tool in electricity price forecasting for australian national electricity market participants *Journal of electrical and electronics engineering, Australia* **22**

Schank, R.C. and Tesler, L. (1969) A conceptual dependency parser for natural language in: *Proceedings of the 1969 Conference on Computational Linguistics* COLING ’69 p. 1–3 Association for Computational Linguistics, USA

Shi, X., Chen, Z., Wang, H., Yeung, D.Y., kin Wong, W. and chun Woo, W. (2015) Convolutional lstm network: A machine learning approach for precipitation nowcasting

Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T. and Hassabis, D. (2016) Mastering the game of go with deep neural networks and tree search *Nature* **529**, pp. 484–489

Sokolova, M. and Lapalme, G. (2009) A systematic analysis of performance measures for classification tasks *Information Processing Management* **45**, pp. 427–437

Sutton, R. (2019) The bitter lesson http://incompleteideas.net/IncIdeas/BitterLesson.html

Taylor, S.J. and Letham, B. (2017) Forecasting at scale *PeerJ PrePrints* **5**, p. e3190

Thongtan, T. and Phienthrakul, T. (2019) Sentiment classification using document embeddings trained with cosine similarity in: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop* pp. 407–414 Association for Computational Linguistics, Florence, Italy

Turing, A.M. (1950) I.— Computing Machinery and Intelligence *Mind* **LIX**(236), pp. 433–460

Van-Tu, N. and Anh-Cuong, L. (2016) Improving question classification by feature extraction and selection *Indian Journal of Science and Technology* **9**

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u. and Polosukhin, I. (2017) Attention is all you need in: *Advances in Neural Information Processing Systems 30*, (Eds.) I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett pp. 5998–6008 Curran Associates, Inc.

Winters, P.R. (1960) Forecasting sales by exponentially weighted moving averages

Wolpert, D.H. and Macready, W.G. (1997) No free lunch theorems for optimization *Trans. Evol. Comp* **1**(1), p. 67–82

Woods, W.A. (1970) Transition network grammars for natural language analysis *Commun. ACM* **13**(10), p. 591–606

Yang, P., Sun, X., Li, W., Ma, S., Wu, W. and Wang, H. (2018) Sgm: Sequence generation model for multi-label classification

Yang, Z., Dai, Z., Yang, Y., Carbonell, J.G., Salakhutdinov, R. and Le, Q.V. (2019) Xlnet: Generalized autoregressive pretraining for language understanding in: *NeurIPS*

Young, T., Hazarika, D., Poria, S. and Cambria, E. (2017) Recent trends in deep learning based natural language processing

Appendix A

Tools

This appendix lists the software and hardware used to create the development environments used in this research.

Software

Microsoft SQL Server - https://www.microsoft.com/en-us/sql-server/
JupyterLab - https://jupyter.org/
python - https://www.python.org/
scikit-learn - https://scikit-learn.org/
pandas - https://pandas.pydata.org/
matplotlib - https://matplotlib.org/
Tensorflow - https://www.tensorflow.org/
PyTorch - https://pytorch.org/
Keras - https://keras.io/
xgboost - https://xgboost.ai/

Hardware

During the research process the AWS and google cloud platforms were used to train and test machine learning models.
AWS SageMaker - https://aws.amazon.com/sagemaker/
Google Colab - https://colab.research.google.com/

In addition to these cloud platforms, a computer with the following specifications was used to train models.
12 core CPU
2560 CUDA cores Nvidia GPU
32 Gb Ram memory

Appendix B

Data Set

The datasets created and used for this research contain private company data and are not publicly available. For access to the full dataset please submit a request to https://civic.ie. In this section we present sample data from the datasets.

B.1 Daily Workload

date	worload
2018-02-01	118.74924215207653
2018-02-02	119.18421552048274
2018-02-03	14.522512580901639
2018-02-04	0.0
2018-02-05	106.11532810138435
2018-02-06	94.40786584408926
2018-02-07	125.53950734895263
2018-02-08	83.65785326318763
2018-02-09	95.58298461700365
2018-02-10	9.106944444444444
2018-02-11	0.0
...	...

B.2 Date features

date	dayofweek	quarter	month	year	dayofyear	day	week	holiday
2018-02-01	3	1	2	0	32	1	5	0
2018-02-02	4	1	2	0	33	2	5	0
2018-02-03	5	1	2	0	34	3	5	0
2018-02-04	6	1	2	0	35	4	5	0
2018-02-05	0	1	2	0	36	5	6	0
2018-02-06	1	1	2	0	37	6	6	0
2018-02-07	2	1	2	0	38	7	6	0
2018-02-08	3	1	2	0	39	8	6	0
2018-02-09	4	1	2	0	40	9	6	0
2018-02-10	5	1	2	0	41	10	6	0
2018-02-11	6	1	2	0	42	11	6	0
...	...	...	...	...	...	...	...	...

B.3 Weather features

date	rain	maxtp	mintp	gmin	soil	wdsp	hm	hg	sun
2018-02-01	0.0	6.2	1.6	0.6	3.200	15.0	23	32	5.8
2018-02-02	1.8	6.9	1.0	-0.7	3.075	10.3	17	23	5.9
2018-02-03	0.3	6.2	1.8	0.7	3.725	13.0	20	30	0.8
2018-02-04	2.0	7.0	-1.2	-0.8	3.300	6.2	13	17	2.0
2018-02-05	0.9	5.7	-2.3	-5.5	2.700	4.5	10	14	2.4
2018-02-06	0.8	3.5	-1.6	-5.7	2.325	11.3	21	34	7.6
2018-02-07	0.3	5.9	-2.9	-5.5	1.425	10.0	18	25	0.6
2018-02-08	1.3	8.3	3.2	-0.5	4.425	10.7	25	36	0.1
2018-02-09	0.3	5.5	-0.3	-1.4	3.125	14.2	26	34	7.5
2018-02-10	4.9	10.8	3.1	-0.6	5.100	18.2	34	47	2.6
2018-02-11	0.0	3.9	-0.6	-0.4	3.525	18.1	26	35	6.3
2018-02-12	0.0	6.0	-0.9	-2.1	2.225	14.9	22	33	6.6
...	...	...	...	...	...	...	...	...	...

B.4 Task Description

Task description	Class
-PMR Attend site - fire exit door at the front of the building needs to be painted cream. the paint has peeled off the door. there is also some scuff marks around the sales floor that need touch ups - black and red paints	1
Attend site and carry out 8hr air conditioning PPM.	1
Attend site: Please attend to review the lighting to the sign to the enterence - potential driver missing. Lighting is to main ivy sign. 0522/755	0
Attend site and carry out emergency lighting PPM	1
Attend site : door handle broken - sales floor to stock room.	0
To investigate and repair leaks found during drain jetting works. Must up-date soon as from store	0
Attend site: Store has called & their shutter was not operating.	0
Attend site and carry out fire alarm ppm.	1
AC Fan Motor Repairs. Following a recent visit our engineer reported that the bearings in the AC fan were worn down causing it to trip the overload. We propose to attend site and strip the unit down. Remove the defective fan. Take the fan to a specialist contractor to be assessed and see if the fan can be repaired. Reinstall the fan once repaired. Clean down works area and remove all debris from site	1
Attend site and carry out monthly 4hr PPM	1
Attend site : HAVING THE SAME PROBLEM AGAIN & THE DOUBLE SOCKET AT THE MICROWAVE STATION & 1 OF THE SOCKETS ISNT WORKING & THIS IS LEAVING US WITH JUSY 1 MICROWAVE IN USE	
Following you recent request please see below costs to attend and carry out the following: Move six floor boxes and run in 4 cat 6 circuits from 3rd floor to 2nd floor Assemble new furniture as per plan on 2nd floor Mirror tv from 1st floor to 2nd floor Fit silent clock in meeting room Clean down works area and remove all debris from site	1
Lights keep flashing after flood last week getting worse and aircon not working properly	0
Attend site and carry out Radars installation and other related works.	0
...	...

Appendix C

Publications

Silva, P. and F. Tellez, J. Cardiff (2015) An Univariable Approach for Forecasting Workload in the Maintenance Industry. *Computación y Sistemas.* **24 No 2** 2007-9737 https://www.cys.cic.ipn.mx/ojs/index.php/CyS/article/download/3399/2851