

**Honours Project Final Report**

**2019-2020**

**Submitted for the Degree of: BSc Computing**

**Stream Data: An Investigation into the Application of Machine Learning for Real-Time Rail Service Predictions**

**Name: Andrew Gaw**

**Programme: Computing**

**Matriculation Number: S1626042**

**Project Supervisor: Huaglory Tianfield**

**Second Marker: Yan Zhang**

**Word Count: 10961**(***excluding contents pages, figures, tables, references and Appendices***)

**“Except where explicitly stated, all work in this report, including the appendices, is my own original work and has not been submitted elsewhere in fulfilment of the requirement of this or any other award”**

****

**Signed by Student: Date: 23/04/20**

Abstract

The vast amount of data being created on a day to day basis along with the difficulty in being able to process and react to the data while it is still in motion has created a massive rise in popularity and focus in the area of machine learning, as many business sectors including the travel sector turn to these algorithms as a solution to processing and reacting to key information from rapidly generated data.

This project was undertaken using a develop and test approach with the aim of determining whether a machine learning algorithm could quickly and accurately make updated arrival time predictions for UK rail services.

Results from this report suggest that while the algorithm was unable to provide high levels of accuracy at real-time speeds for the specific data feed used, using it on alternative execution system with a larger selection of data could yet provide insight into how well suited the algorithm was in this research area.

# Acknowledgements

I would like to take this opportunity to thank my supervisor, Huaglory for all his help and support throughout this project’s development.

I would also like to thank my Family for all the support and encouragement they have provided throughout my honour’s year.

Contents

[Acknowledgements 3](#_Toc38548186)

[1. Introduction 6](#_Toc38548187)

[1.1 Project Background 6](#_Toc38548188)

[1.2 Project Overview 9](#_Toc38548189)

[1.2.1 Project Outline 9](#_Toc38548190)

[1.2.2 Project Aims & Objectives 10](#_Toc38548191)

[2. Literature Review and Technology Assessment 12](#_Toc38548192)

[2.1 Investigation into Machine learning and its usage 12](#_Toc38548193)

[2.1.1 Types of Machine Learning and their uses 12](#_Toc38548194)

[2.1.2 Applications of machine learning in the travel sector 13](#_Toc38548195)

[2.2 Examination of Determining an Algorithm’s suitability 14](#_Toc38548196)

[2.2.1 Determining an Algorithm’s suitability 14](#_Toc38548197)

[2.2.2 Analysis of machine learning for time predictions 14](#_Toc38548198)

[2.3 Investigation into Stream Processing 15](#_Toc38548199)

[2.3.1 Stream Processing and Processing Platforms 15](#_Toc38548200)

[3. Execution 17](#_Toc38548201)

[3.1 Research Method 17](#_Toc38548202)

[3.2 Project Lifecycle 17](#_Toc38548203)

[3.3 Implementation 18](#_Toc38548204)

[3.3.1 Choosing the Programming Language and Development Environment 18](#_Toc38548205)

[3.3.2 Choosing and connecting to the chosen Data stream 18](#_Toc38548206)

[3.3.3 Implementing the Stream Storage and Processing Platform 20](#_Toc38548207)

[3.3.4 Setting up Apache Kafka 20](#_Toc38548208)

[3.3.5 Setting up Apache Spark 22](#_Toc38548209)

[3.3.6 Choosing the Machine Learning Algorithm 23](#_Toc38548210)

[3.3.7 Tuning the Machine Learning Algorithm 24](#_Toc38548211)

[4.0 Testing & Evaluation 25](#_Toc38548212)

[4.1 Evaluation Methods 25](#_Toc38548213)

[4.2 How Tests Runs were Documented 25](#_Toc38548214)

[4.3 Conducting the First Test Run 26](#_Toc38548215)

[4.4 Conducting the Second Test 28](#_Toc38548216)

[4.5 Conducting the Final Test 30](#_Toc38548217)

[4.6 Limitations in The Application 31](#_Toc38548218)

[4.6.1 Hardware 31](#_Toc38548219)

[4.6.2 Relationships between data variables 31](#_Toc38548220)

[4.7 Ethical and Professional Issues 32](#_Toc38548221)

[4.7.1 Legal 32](#_Toc38548222)

[4.7.2 Privacy 32](#_Toc38548223)

[5.0 Discussion & Conclusion 33](#_Toc38548224)

[5.1 Final Discussion of Results 33](#_Toc38548225)

[5.2 Project Limitations and Further Work 33](#_Toc38548226)

[5.3 Conclusion 34](#_Toc38548227)

[6.0 References 35](#_Toc38548228)

[7. Figures 41](#_Toc38548229)

[8.0 Appendix 42](#_Toc38548230)

[8.1 Analysis of Requirements 42](#_Toc38548231)

[8.1.1 Functional Requirements 42](#_Toc38548232)

[8.1.2 Non-Functional Requirements 42](#_Toc38548233)

[8.2 Design 42](#_Toc38548234)

[8.2.1 Application Architecture 43](#_Toc38548235)

[8.3 Testing 44](#_Toc38548236)

[8.3.1 Test Run 1 Results 44](#_Toc38548237)

[8.3.2 Test Run 2 Results 47](#_Toc38548238)

[8.3.3 Test Run 3 Results 52](#_Toc38548239)

# 1. Introduction

This section provides the reader with a description of what machine learning is and a brief history on how the algorithms have been developed over time while also showing some present-day applications of machine learning in different computing sectors. It also highlights to the reader the current rise in the amount of data being produced by everyday devices and tells them how machine learning can be used to process these larger amounts of data. Furthermore, it will provide the reader with a brief overview of the project’s objectives to be completed from the use of a literature review and the practical objectives to be completed throughout the implementation of the application.

## 1.1 Project Background

The interest in artificial intelligence has increased exponentially over the last few years due to the growing popularity of “Internet of Things” devices and subsequently the rising volume of data which these devices have been generating. The International Data Corporation forecasts that the worldwide spending on the Internet of Things is set to increase in 2019 by 15.4% compared to 2018 (1. IDC. 2019), while the total number of devices is set to exceed 50 billion by 2022 (2. Juniper Research. 2018). As previously mentioned, the rise in popularity of these devices has seen a massive increase in the amount of data being formed, with 90% of the data in the world being generated between 2016 and 2018 (3. Marr, B. 2018). Cisco highlights the Internet of Things driving this increase and estimates that the total amount of data generated (and not necessarily stored) by any device will reach 847 ZettaBytes (ZB) per year by 2021, compared to 218ZB per year in 2016 (4. Cisco. 2018). Although many companies have the means to store these large streams of data being generated, the critical challenge is using the data while it is still in motion – and extracting valuable information from it (5. Sharma, S. K. Wang, X. 2017). Many companies have turned to machine learning algorithms to process the large volumes of stream data.

machine learning, at its core, is an application of artificial intelligence which gives systems the ability to dynamically learn and improve from experience without being explicitly programmed (6. Expert System. 2019). Although the foundations for machine learning have been around since the 1800s from mathematical equations, the first reference to machine learning was made in 1950 when Alan Turing published Computing Machinery and Intelligence, in which he asked: “Can machines think?” (7. BBC. n.d), The paper gave an initial attempt to describe how artificial intelligence could be developed and posed the famous Turing Test, a spin on the “Imitation Game”, where 3 terminals are used, two operated by humans and one operated automatically by the computer itself. One person acts as the questioner and asks various questions to the other human and computer participant, the other human and computer make responses to the questioner and they are then left to decide which response they think was the person and which was the computer (8. Rouse, M. 2019). Following this, in 1952 Arthur Samuel created a program which helped an IBM computer to play a game of checkers, performing better the more games it played. Samuel then coined the term “Machine Learning” in 1959 (9. Javatpoint. n.d).



Figure 1. Arthur Samuels playing Checkers with an IBM 701 Computer, 1956.

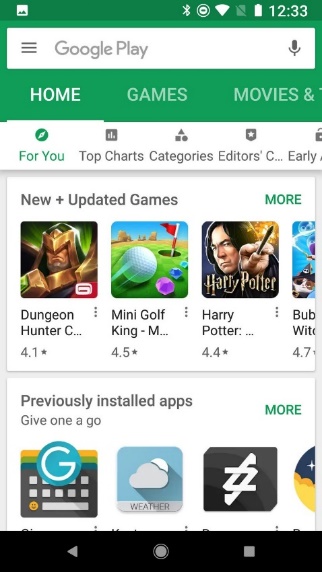
Machine learning, like other forms of artificial intelligence has surged in popularity over the last few years. The IDC predicted that the worldwide spending on Cognitive and Artificial intelligence will reach $77.6 Billion in 2022 with a compound annual growth rate of 37.3% (10. IDC. 2018). While Mckinsey Global Institute have said that artificial intelligence and machine learning have the potential to create an additional $1.4-2.6 Trillion in value for the sector of Marketing and Sales, also potentially creating an additional $1.2-2 Trillion in Supply Chain Management and Manufacturing (11. McKinsey Global Institute. 2018). Machine learning has beneficial uses in various business sectors, one of the most widely known uses of machine learning is recommender systems, Google for example use machine learning in their recommender systems for both YouTube and their Google Play store services to recommend videos which the user might want to watch after their current video has ended and to recommend apps which Google thinks the user might enjoy based on the other apps they install (12. Google. n.d).

Figure 2 Google Play’s “For you” section

Another area which makes use of recommender systems in their websites and applications is the travel sector. Many travel companies have implemented the previously mentioned recommender systems into their websites to highlight offers and holiday packages which might be of interest to the user based on previous searches or bookings made (13. Bulanov, A. n.d). Machine learning has also been used in the travel sector through the implementation of “chatbots”, which can provide customers with automated support 24/7 and enhance the user’s travel experience via reminders and recommendations for things to do and places to visit (14. Chawla, S. 2019).

While the travel sector has majorly benefitted from machine learning implementation to improve the overall user experience, the sector still struggles with day to day operations and communication with travellers. From 2016 to 2017, the BBC highlighted that 11.7% of the trains running in the UK were late, with 16.3% of those trains being later than 3 minutes (15. BBC. 2017). In 2019, it was reported that a third of British rail services failed to reach stations on time (16. The Guardian. 2019). The Office of Rail and Road (ORR) found in their first quarterly review for 2019 to 2020 that there were 24.9 complaints made per 100,000 journeys for franchised operators, and although this is a decrease from the first quarter of 2018 to 2019, they found that punctuality and reliability of rail services remains the largest category of passenger complaints this year (17. ORR. 2019). In their most recent Quarterly update (2019-20 Q3) the ORR found that 59.2% of all UK services arrived at stations on time or within one minute of the original arrival time. 73.9% of the services arrived within 3 minutes of the original arrival time, and 97.8% arrived within 15 minutes of the time (18.ORR,2020). These results were a drop from the previous quarter, which saw 65.1% of all passenger services arriving on time or within a minute of the original arrival time, 84.3% arriving within 3 minutes of the original arrival time, and 98.5% arriving within 15 minutes of the original time (19. ORR, 2020).

A screenshot of a cell phone

Description automatically generated

Figure 3. Punctuality of Recorded Station Stop.

As with Internet of Things Devices, one of the large challenges faced in the travel and specifically rail sector is the amount of data being generated and processed. In the UK alone there are 2566 train stations (20. Department of Transport, 2019) and on average around 20,000 daily passenger services (21. Department of Transport, 2018). As highlighted earlier, many companies and sectors have begun implementing machine learning to process the streams of data they generate. There are many ways in which implementing machine learning would improve stream data processing, for example machine learning algorithms are able to identify specific trends and patterns in data which are often not visible or obvious to the human eye (22. DataFlair, 2019). Machine learning can also improve stream data processing as the algorithms remove the need for human input. Machine learning algorithms can make decisions by themselves and subsequently improve their future decision-making by analysing both the errors and successes of previous decisions (23. Agrawal, A, n.d). Another way in which machine learning algorithms can improve stream data processing, slightly highlighted in the last point, is that algorithms continuously improve over time, as they can learn and improve from previous decisions made, helping to increase their efficiency and accuracy over time. Although there are many benefits of implementing machine learning algorithms to improve the processing of stream data, machine learning also has some pitfalls. One of the main disadvantages of utilising machine learning is the selection of an algorithm to implement. This is still a manual job, and there is no “One size fits All model”, finding an algorithm which offers the highest result accuracy for a specific dataset can often be both resource and time consuming (24. TechVidVan, 2020). Another disadvantage of utilising machine learning is the ongoing argument that it fuels the increase of “Dataism”, a term coined by Yuval Noah Harari which refers to the current stage civilisation is going through where often people trust the results of algorithms and related data over our own human judgement (25.Stewart, M, 2019).

## 1.2 Project Overview

This section aims to highlight the outline of the project and ultimately what the project’s research aims to answer at the end. It will also identify the objectives which will be completed through the undertaking of a literature review & technology assessment as well as presenting the primary research objectives for the project.

### 1.2.1 Project Outline

As a result of the increasing amounts of data being generated In the world and the growing challenges of quickly and efficiently analysing and reacting to the results of these streams of data, alongside the continuous criticism rail services receive due to their overall reliability and punctuality when providing updates to service running times, and the increasing focus on machine learning’s application to provide a solution to processing large volumes of stream data. These factors give clear justification for the creation of an application which will process large volumes of rail service update data and quickly predict new arrival times based on the updated information. Thus, the Research Question for this project is:

**Can an application implementing a Machine learning algorithm be applied to railway stream data to provide accurate arrival time predictions?**

### 1.2.2 Project Aims & Objectives

The aim of this project is to develop an application which makes use of a machine learning algorithm to analyse railway service update stream data and provide accurate predictions for what time each service while arrive at its corresponding train station. To find an answer to the research question some objectives have been defined which are paramount in the successful execution of the project. The objectives of this project can be split into 2 categories: Objectives which will be found through the completion of the Literature Review and Technology Assessment, and Practical Objectives which will be completed throughout the primary research phase.

**Objectives which will be found through the completion of the Literature Review and Technology Assessment are:**

* **Investigation into machine learning and its usage**

This will involve research into the different types of machine learning algorithms available and a review of their suitability for the specified project. Will also include an investigation into other applications of machine learning in the travel sector.

* **Examination into determining an algorithm’s suitability**

Analysis of factors which should be taken into consideration when choosing a machine learning algorithm, helping to assist in the choice of algorithm which will be utilised for the project. Will also look more closely at other work making use of machine learning for time predictions to analyse algorithms used and their suitability for this prediction type.

* **Investigation into Stream Processing and Processing Platforms**

Research into the concept of stream processing with a comparison between traditional methods and stream processing methods. Will also investigate different stream processing platforms available and evaluate their suitability for the project.

**Practical Objectives which will be completed through the primary research phase are:**

* **Choose and connect to the chosen data feed**

An appropriate data feed will be chosen and the initial connection to the data feed will be established. Once connected the data will be analysed to identify the types of data provided and its structure. Datatypes and overall structure will be formatted to prepare it for the machine learning algorithm.

* **Implement the stream processing platform**

Using knowledge gained from the Literature Review and Technology Assessment to choose a suitable stream processing platform for the application. Once a processing platform is chosen, the storage system for the data will also be chosen and connected to at this stage.

* **Implement the machine learning algorithm and perform initial testing**

Once the stream processing platform has been implemented, insight gained from the completion of the Literature Review and Technology Assessment will allow for the implementation of an appropriate machine learning algorithm which is suitable for the data feed chosen. Upon successful implementation of the algorithm, the data will be split into training and testing sets. An initial test will be performed to analyse the accuracy of the predictions made by the algorithm on the datasets.

* **Tune the machine learning algorithm and perform final testing**

Evaluation of the initial test results will then allow for the algorithm to be fine-tuned until the highest accuracy is achieved. After the algorithm is tuned, more in depth testing will take place, allowing the algorithm to run for a set period at three different time slots throughout the day. Results achieved throughout these time slots will be documented.

* **Evaluate the final test results and find the conclusion**

Completion of the final testing will provide in depth documentation for the prediction accuracy of the machine learning algorithm during each time slot. A conclusion will then be drawn from these results to indicate the suitability of the algorithm for this specific dataset and to determine the success of the overall application of machine learning in this project area.

# 2. Literature Review and Technology Assessment

The Literature Review and Technology Assessment are a crucial element of the project. This stage allows for the forming of the knowledge required to successfully undertake and complete the main project objectives. Previous research will be studied to help build an overall understanding of the project area, while analysis of different technologies in the area will provide the knowledge required for the author to decide how the project will be implemented and what technologies it will utilise.

The Literature Review and Technology Assessment will investigate and complete the following research objectives defined in the project outline stage:

* Investigation into Machine learning and its usage
* Examination into determining and algorithm’s suitability
* Investigation into Stream processing and processing platforms

## 2.1 Investigation into Machine learning and its usage

### 2.1.1 Types of Machine Learning and their uses

Machine learning is described as a field of computer science that studies algorithms and techniques for automating solutions to complex problems that are hard to program using conventional programming methods (26. Rebala, Gopinath. et al. 2019). Two of the most widely utilised machine learning types are Supervised and Unsupervised learning, however Semi-Supervised learning and Reinforcement learning are other available methods (27. SAS. n.d). In Supervised learning, training data is labelled and consists of an input vector X and an output vector Y, provided by a human or computer ‘supervisor’, whereas Unsupervised learning doesn’t utilise a supervisor and the data is unlabelled (28. Bashier, E. Khan, M. Mohammed, M. 2016). If a system uses Supervised learning, it is possible to split the problem area based on the nature of the data, If the output value for the system is categorical, it is a classification problem. If the output is a continuous real value in a certain range, then it is a regression problem (29. Ciaburro, G. 2017). Supervised learning utilises both classification and regression techniques to create predictive models. A labelled dataset of input and output variables is used to initially train the supervised learning model. The trained model is then used to generate predictions for the test dataset or when new data is received (30. Shoba, G. Rangaswamy, S. 2018). Supervised learning methods include linear and logistic regression, decision trees and support neural networks. For Unsupervised learning, one of the main techniques is Clustering, where unlabelled data is grouped together by putting the features of the data into a metric called a similarity measure (31. Google Developers. n.d). For example, if the dataset is films, similarities could be found from the film’s Director, Actors or Age Rating. Unsupervised machine learning algorithms include K-means clustering, K-Nearest-Neighbour (KNN) and principal component analysis. Semi-Supervised learning is a branch of machine learning which tries to solve problems using both labelled and unlabelled data with an approach that employs concepts from both clustering and classification methods (32. Bonaccorso, G. 2018). Whereas Reinforcement Learning is a technique which aims to develop algorithms that can learn and adapt to changes in the environment. The environment will initially send a state to the agent (Learning Algorithm), once the agent sees the state, it then sends a response action back to the environment, the next time the environment sends the state over, it will also send a “reward” response, allowing the agent to update its knowledge based on the reward (33. Huang, K-H. 2018).

### 2.1.2 Applications of machine learning in the travel sector

As mentioned in the project background section, many companies have implemented machine learning algorithms to solve a wide variety of problems in different business areas including the travel sector. There are various applications of machine learning specifically in the travel sector, one application is the previously mentioned recommender systems, which are often used in travel specifically by holiday booking companies who are able to utilise these systems to recommend different locations, hotels, and travel packages based on the user’s previous page visits, time spent on each page and previous bookings made. For example, if a user tends to book weekend trips to Spain every August, offering them a two week holiday in March will most likely not be of interest to them, but by offering them a trip more specific to their regular bookings, there is a much higher chance of sparking interest. Displaying user-tailored recommendations at the right moment can be financially beneficial for businesses by helping to generate more sales on their pages, while also having the benefit of improving user engagement by providing personalised experiences, and improving overall customer loyalty by making the users feel like the company really understands what they are looking for (34. Altexsoft, 2018). Although recommender systems are one of the most popular machine learning implementations in the travel sector, algorithms can also be effectively utilised in other prediction and classification problems. Qiang Ye et al investigated how machine learning algorithms could be implemented to perform sentiment classification of online travel reviews. They used three supervised machine learning algorithms: Naïve Bayes, Support Vector Machine (SVM) and the character-based N-Gram for sentiment classification of travel blog reviews based on seven popular destinations across Europe and the United States. Their results showed that SVM and the character-based N-Gram models performed better than the Naïve Bayes model. However, all three models could achieve more than 80% Classification correctly. They also noted that when using the models on smaller datasets, the difference in performance was extremely significant (35. Ye, Q. Zhang, Z. Law, R. 2008). Looking at predictions problems, Nikolas Julio et al conducted a comparison of three supervised machine learning algorithms: Artificial Neural Networks, Support Vector Machine (SVM) and Bayes Networks, for the prediction of bus travel speeds using traffic shockwaves. Two types of data were used in this investigation, historical data and real-time data which had a ten-minute delay period. The result found that, although different datasets and use cases could provide alternative conclusions, Artificial Neural Networks performed the best on the dataset used, obtaining an improvement of up to 23% in root mean square error values (RMSE) which gave an indication of how far from the real values the predicted speeds were for each model, while being an improvement of 3.3% over the second best model (SVM) used (36. Julio, N. et al. 2016).

## 2.2 Examination of Determining an Algorithm’s suitability

### 2.2.1 Determining an Algorithm’s suitability

Although there are some machine learning algorithms which can be implemented to tackle a wide range of problems, most algorithms are tailored to perform better on specific problem areas and scenarios, and as such, provide better results in these areas than others. To ensure the most suitable algorithm is chosen for the specific project area, it is first important to look at the different factors which can play a part in choosing an algorithm's suitability. An investigation by (37. Cui, L et al. 2018) into the application of machine Learning for the areas of “Traffic Profiling, Device Identification, Security, Edge Computing and Software-Defined Networking”, analysed Supervised and Unsupervised Learning approaches for each area, concluding that different machine learning techniques were suitable for each area depending on the type of data being generated and the device which generated the data. Another investigation into the use of machine learning for data generated from “Smart Cities”, found that there are 3 major factors to determine which machine learning algorithm to use. The first factor was the characteristics of the smart device and what kind of data it was generating. The second factor was looking at the specific features and characteristics of the generated data, and the final factor was considering the taxonomy of each algorithm and determining what the algorithm was looking to accomplish with the data, such as finding patterns, predicting values, predicting categories or feature extraction (38. Mahdavinejad, M et al. 2018). Joseph Siryani et al (39. Siryani, J et al. 2017) implemented and compared the use of 4 different machine learning techniques (Bayesian Network, Naïve Bayes, Decision Trees and Random Forest) to produce a data-driven Data Support System (DSS) which created a prediction about whether or not a technician should be sent to the location of a customer’s smart meter, or if the problem could be solved remotely, concluding that although all 4 algorithms were suitable for this implementation, some of the algorithms had higher accuracy success ratings than the others. Random Forest was the most suitable method for analysing the smart meter’s data, providing the highest accuracy of 96.69% and the lowest error rate percentage, closely followed by Naïve Bayes which had an accuracy of 96.57%. Bayesian Network had a much lower accuracy rating of 54.92%, highlighting that the other two methods were much more suitable for this type of system and dataset.

### 2.2.2 Analysis of machine learning for time predictions

To further improve the likelihood of a suitable algorithm being chosen for the problem area and dataset, it is important to analyse other work in the field of time prediction to help identify any algorithms which have been previously utilised that provided inadequate results and which algorithms performed better for the purpose of this prediction type. An investigation was conducted by (40. Vanajakshi, L. Rilett, L. R. 2007) into the usage of machine learning for short term travel time predictions. The investigation made use of both historic and real time traffic data obtained from a traffic management centre in Texas. They utilised a Support Vector Machine for this scenario and used it to compare results of those obtained from an Artificial Neural Network, which they noted is a popular algorithm for time predictions and has been thoroughly researched by others. The results of the investigation found that SVM performed just as well as the Artificial Neural Network in most scenarios, however noted that the SVM provided better performances on smaller datasets and larger data variations, whereas the Artificial Neural Network performed better on larger datasets with very little data variation. Tingting Yin et al similarly made use of both SVM and Artificial Neural Network algorithms for the purpose of predicting bus arrival times at stops with multi-routes, using three main input data variables: the arrival time of the preceding bus on the target route, other routes passing the same stop and the travelling speed of the bus. Results found that both algorithms provided a high accuracy when predicting arrival times. However, the Artificial Neural Network edged out the SVM when looking at the overall comparison of each model. Although the Artificial Neural Network performed better overall, the mean absolute percentage errors (MAPE) of predictions were less than 10% for both models in most cases (41. Yin, T. Zhong, G, Zhang, J. Shanglu, H. Ran, B. 2016).

## 2.3 Investigation into Stream Processing

### 2.3.1 Stream Processing and Processing Platforms

The difficulty of being able to analyse data in real-time to provide a quick response, highlighted in the project background section, is one of the main reasons for the increasing focus on stream processing. Stream processing is a technology which lets users query continuous data streams and detect conditions quickly within a small time period of receiving the data (42. Perera, S. 2018). Stream processing architectures have many benefits over the more traditional data processing architectures, traditional approaches are often slow and complex between the ingestion of data and the analysis of the data thereafter. Traditional architectures can also be very one-dimensional, as they often utilise one singular back-end database which all applications need to query to access the data (43. Tzoumas, K. Friedman, E. 2016). There are many different platforms available for stream data processing, and they often provide support for machine learning implementation. One processing platform with this support is Apache Spark, which provides an extension to its base platform specifically for stream processing called Spark Streaming. Spark Streaming provides a fault tolerant and scalable option for processing live data streams, while being able to ingest data from multiple sources including Amazon S3 Buckets, Apache Kafka Topics and basic file formats such as XML and JSON before processing the data and outputting it to storage systems including databases, dashboards or a Hadoop Distributed File System (HDFS) (44. Apache, n.d).

A picture containing drawing

Description automatically generated

Figure 4. Basic Spark Streaming Architecture

Alongside the ability to store the processed data, Spark Streaming can also be used in conjunction with one of the two available Spark Machine Learning libraries, MLib and SparkML, to perform machine learning on the processed data streams. Of the two libraries, MLib is the more mature option and provides a higher-level API for machine Learning and statistical analysis than the core Spark API (45. Guller, M. 2016). The MLib library also contains a large selection of machine learning algorithms providing both supervised and unsupervised algorithms, including the Classification, Regression and Clustering methods mentioned previously (46. Apache. n.d). Another platform tailored to stream processing which offers support for machine learning algorithms is Apache Storm, a hugely popular platform which has been utilised by companies including the likes of Yahoo and Twitter. Although Storm doesn’t provide machine learning libraries like Spark, there are many packages available for Storm including Trident-ML, which offers support for Linear Regression, Classification and Clustering as standard (47. Nalya, A. Jain, A. 2014). Spark and Storm both have similar advantages when looking at things like fault tolerance, debugging and error handling. However, when looking at overall support for each platform and their ease of implementation, Storm is a much more complex platform to make use of and has very limited resources. Whereas Spark has a lot more resources at its disposal (48. DataFlair. 2018), including the Machine Learning Libraries mentioned earlier, which would be very beneficial for the execution of the project. Alex Lui gives an example of how machine learning alongside the processing platform Apache Spark can be utilised for the purposes of fraud detection. This example focuses on the implementation of Random Forest and Decision Trees but also highlights how other methods, including Linear regression could be suitable depending on the type of data being analysed (49. Lui, A. 2016).

# 3. Execution

The execution section is also a hugely important part of the project as it defines the main problem area which the project aims to investigate. It also gathers the knowledge and understanding gained from the completion of the Literature Review and Technology Assessment to allow for clear justification of the approach taken in the attempt to find an answer to the research question.

The section will also document the technologies used and the implementation methods of the Primary Objectives which were defined in the Project Background section:

* choose and connect to the chosen Data Feed
* Implement the stream processing platform
* Implement the machine learning algorithm and perform initial testing
* Tune the machine learning algorithm and perform final testing

## 3.1 Research Method

The aim of this project was to investigate and document whether a machine learning algorithm could be used to process stream data and accurately predict the arrival time of trains at their respective stations using real-time update data. To provide a complete evaluation of the application, the primary research method used was a Develop and Test approach. For the successful completion of the machine learning prediction application, the main stages undertaken were: Analysis of Requirements, Design, Implementation, Testing and Evaluation. The Requirements and Design of the application have been documented in the Appendix and as previously stated, Testing and Evaluation will be discussed in Section 4.

## 3.2 Project Lifecycle

The Lifecycle chosen for this project was an Iterative approach, as it allowed for the creation of a basic prototype early in the development process. The initial prototype helped to provide a foundation for the project and also allowed for the chosen machine learning algorithm’s suitability when predicting travel times to be tested early, meaning if the accuracy provided by the algorithm was unsatisfactory it could be changed promptly before the final testing was conducted and documented. Creating iterations of the project also allowed for more thorough tuning of the machine learning algorithm as smaller initial tests were ran at the end of each iteration to analyse how accurate the algorithm was with each change to the parameters tuned. This again allowed for higher accuracies to be achieved before the final testing was conducted, further improving the likelihood of the application successfully answering the research question.

## 3.3 Implementation

This section will present the decisions made by the author when implementing all aspects of the project and will aim to justify why these decisions were made based on previous knowledge gained from the literature review and technology assessment. It will also show the implementation and completion of the primary objectives documented in the Project Background Section.

### 3.3.1 Choosing the Programming Language and Development Environment

After researching the streaming platforms and machine learning libraries available and investigating the support these packages provided for different programming languages, it was decided that a Python Application would be created using Anaconda and Spyder (50. Anaconda, 2020). The decision to use Spyder and Python to implement the application was due to a number of factors, the ease of implementation which python provides, alongside its support for all stream processing platforms analysed in the Literature Review and Technology Assessment, and the author’s previous experience of working with both Spyder and Python meant that the initial prototype for the project could be created rapidly using this programming language and environment. To improve the readability of the code it was decided that separation of the connection and the storage of the data stream from the main machine learning application would be advantageous, as such two separate files were used. This also made debugging and error handling of the data ingestion easier as it could be established quickly if the problem lay in the data storage file or the file reading from the storage.

### 3.3.2 Choosing and connecting to the chosen Data stream

The first part of the Implementation stage was to choose a suitable datastream for the problem area and establish the connection to the stream so data being requested could be simply displayed in the command terminal. There were two main sources of rail data available, the Network Rail Data feeds and the National Rail Darwin Data feeds. Upon further investigation into both sources, it was decided that the Nation Rail Darwin Feeds (51. National Rail, n.d) would be utilised. This decision was made due to the amount of data available through these feeds as national rail provides a wider range of data than the network rail feeds, including feeds for both historical and real-time data. Along with the types of feeds available, the Darwin Push-Port feed included access to real-time update data, including the necessary prediction variables for arrival times for specific trains at their corresponding stations which could be used as the target variable when implementing the machine learning algorithm. Another advantage when utilising the Darwin Push-Port feed was the amount of support provided when accessing the feeds. The National Rail Feeds have a large Github community where sample code for accessing each type of data feed is provided in numerous coding languages including Python. Once an account was registered and the desired data feed was subscribed to, the credentials required to access the relevant data were created. Making use of the sample Github code was advantageous at this stage as it meant the connection didn’t need to be established from scratch and the time saved here provided more leniency in the later implementation stages (51. Hicks, P et al. 2019). Once the sample code was set up in Spyder and the credentials had been entered, a simple line of code to view the data in the command terminal was used and when ran, the output was a continuous stream with the data structure documented in figure 5:

A picture containing photo, table, black, white

Description automatically generated

Figure 5. Initial Structure of the data.

As shown in the figure above, the initial data was in the format of XML and the <Pport> tag Included a custom schema, which is individual links used when accessing each response type in the feed. For the purpose of this project only Schema five (PushPort/Forecasts/v3) was used. When looking more closely at the data received, it was noted that all values were sent as type String. There were five main variables which were pulled from the Forecasts : The Station Name, Public Times, Which are the specific times displayed publicly on the respective station’s arrival/departure boards, Working times which are the updated times followed by workers and associates of the railway (these times are not displayed to the public), Forecast Times, which are the initial prediction times for the arrival/departure, and the working prediction times, which are the most recent predictions, the working prediction time was the variable used as the target output for the machine learning algorithm later on . Due to the project’s aim being to predict time variables, the decision was made to reformat the data so time values would be represented as numbers with decimal points. For example, if the time was “12:30”, the data would be reformatted to appear as “12.30”, some of the variables included second values which could only be 30, so to reformat these 5 was added on to the end of the decimal variable (as 30 seconds is 0.5 minutes), so if the previously mentioned time was “12:30:30”, this was reformatted to be 12.305. To improve the performance of the connection application and to avoid needing to remove the custom schema later, it was also beneficial at this stage to utilise the XML Element Tree API (Version 20.5) to parse the XML, remove the provided custom schema and then create a new schema to specifically pull the data required from the Forecasts. Once this data was extracted, the values were put into a python dictionary with more meaningful variable names to improve the understandability of the data being used later, a second python dictionary was then defined and used to convert the data format of the necessary original data from String to Float Type. A simple If statement was also used here to ensure that every variable placed in the python dictionaries had a value, removing the need to drop empty values when preparing the data to be used in the machine learning algorithm later. Figure 6 documents how the reformatted data in the python dictionary looked when displayed in the command terminal:

A screen shot of a computer

Description automatically generated

Figure 6. Reformatted DataStream

### 3.3.3 Implementing the Stream Storage and Processing Platform

Before implementing a storage method for the data stream, it was important to first decide which stream processing platform would be used, as each platform offers different storage options. The completion of the Literature Review and Technology Assessment gave insight into two of the most popular stream processing platforms, Apache Spark and Apache Storm. The decision was made to use Apache Spark (52. Apache, 2020) due to the support Spark provides for both stream data and machine learning through its additional packages and also the performance benefits which Spark provides through its ability to store data in RAM while its processing rather than needing to store the data in main system memory. Once the processing platform was chosen, the next decision was to choose a storage system for the data. Spark applications can read data from various sources, including S3 Buckets, Apache Kafka and the general file types mentioned in the Literature Review. Upon further investigation into the advantages and disadvantages that each of the storage options had over the others, it was decided that Apache Kafka (53. Apache, 2020) would be used as an initial pipeline for the stream data. The decision to use Kafka was made due to its low latency, being able to ingest and store streams with a maximum of 10 millisecond latency and also Kafka’s ability to process larger volumes of data as a result of its low latency, which are both hugely advantageous when making use of streaming data.

### 3.3.4 Setting up Apache Kafka

Apache Kafka works by using Kafka Producers which send records of data to a Kafka Node, also called a Broker. Records are stored in Kafka Topics, and as Topics increase in size, they split in partitions to provide the performance benefits mentioned earlier. Kafka Consumers can then be used to subscribe to the Kafka Topic and read the data stored in this topic (54. Kozlovski, S. 2017). To be able to run Apache Kafka, Apache Zookeeper is also required as it is used by Kafka to track and configure brokers, topics and partitions (55. Apache, n.d). A basic tutorial was followed to aid with the initial setup and configuration of Apache Zookeeper (56. Aslam, S. 2017). Once the Setup was complete, a command terminal was used to test that Zookeeper was working properly, it was run on the default port 2181:

A picture containing sitting, table, display, black

Description automatically generated

Figure 7. Starting Apache Zookeeper using “zkserver”.

Once Apache Zookeeper was correctly configured, the next stage was to set up Apache Kafka and create the Kafka topic which would be used to store the data. Another tutorial was followed here to help initially configure Apache Kafka (57. Aslam, S. 2017). To help organisation, Kafka and Zookeeper were both installed to individual subfolders inside a Project folder. After configuring Apache Kafka, an administrator command terminal was opened in the windows subfolder of the main Kafka download folder. Figures 8 and 9 show the command used to launch Kafka and the response in the command terminal respectively:



Figure 8. Command to run Apache Kafka.

A picture containing bottle, photo, black, sitting

Description automatically generated

Figure 9. Running Apache Kafka.

After Apache Kafka was set up and an initial connection could be established with Zookeeper, the next stage was to create a Kafka topic and update the connection python file to include the required code to send the stream data to the Kafka topic, ensuring the Data was being sent accordingly. To create the initial topic, a new command prompt was opened in the bin\windows directory of the main Kafka folder, Figure 10 shows the command entered in the command window to create a new Kafka topic and the response when the command was run:

A close up of a screen

Description automatically generated

Figure 10. Creating the Kafka Topic “Main\_Test”.

After creating the topic, the connection python file was then updated to send the contents of the datastream to the Kafka topic. Necessary dependencies were imported for Kafka Producers, Consumers and JSON. A Kafka producer was instantiated which would convert the previously created python dictionary into JSON and then push the JSON to the created Kafka Topic. While a Kafka consumer was unnecessary for the main project application, it was beneficial to instantiate a Kafka Consumer here to connect to the Kafka topic and ensure the data was being received from the producer. Figure 11 shows the output of the Kafka Consumer in the command terminal:

A white sign with black text

Description automatically generated

Figure 11. The Contents of the Kafka Topic.

### 3.3.5 Setting up Apache Spark

A picture containing sitting, table

Description automatically generatedThe figure above shows that the datastream was successfully being sent to the Kafka topic. After this was confirmed, the next stage was to set up Apache Spark and read the data from the topic. As with the Zookeeper and Kafka initial installations and configurations, a tutorial was followed to help ensure Spark was correctly set up on the execution system (58. Maramreddy, R. R. 2019). After Spark was downloaded and configured, it was then important to ensure it was running properly in the command prompt. Figure 12 shows the command used and the response when attempting to run Spark in an Anaconda Prompt:

Figure 12. Running Apache Spark.

To be able to connect to the Kafka topic, an initial streaming context had to be created which is used to tell spark how often it will query the data source for new data, due to the nature of this project, the context time was initially set to one second, meaning data would be pulled from the Kafka topic every second. Once this was decided, the stream was created by passing the streaming context, port number and the name and partition of the Kafka topic to be read from into the KafkUtils.createStream function. When a stream is started, it is represented as a Dstream object which is a series of Resilient Distributed Datasets (RDD). RDDs are a collection of objects or elements. Once the stream had been set up, a map() function was required to create a new Dstream object which took the original Dstream object and read the stream as JSON. Once the second stream was created, a test was run to ensure the data was being read from the Kafka topic correctly into the Dstream object. The first test run revealed an issue between the topic and processing platform, as the command terminal was outputting empty Dstreams every second. Upon further investigation and debugging, it was found that the Processor of the Execution system was being placed under intense load and creating a bottleneck when running both the connection file and the main streaming application. To overcome this bottleneck, it was decided here to attempt to increase the streaming context time until stability was created and the Dstream could be processed and displayed correctly. The lowest stable Streaming Context time for the application on the execution system used was 20 seconds, creating a 20 second delay between receiving the data and processing it. Figure 13 shows the output of the stream creation once the streaming context had been altered:

A picture containing black, white

Description automatically generated

Figure 13. Output of Datastream.

### 3.3.6 Choosing the Machine Learning Algorithm

Once spark was set up, the next stage was to choose a machine learning algorithm and test its suitability for the chosen datastream. The Literature Review and Technology Assessment completed previously highlighted the main types of machine learning algorithms available. Due to the data in the stream being labelled and the aim of the application being to predict numeric values, it was decided that a Supervised learning approach and a Regression algorithm would be utilised for this project. The Literature Review and Technology Assessment also provided insight into other research undertaken to predict travel times using machine learning algorithms, highlighting that Support Vector Machines (SVM) and Artificial Neural Networks (ANN) were both popular and suitable choices for these types of predictions. It was decided here that it would be advantageous to choose a different algorithm for this project to provide alternative research in the problem area. As such, it was initially decided a Linear Regression Algorithm would be used due to its support in Spark Mlib and the lack of other research using this algorithm for time predictions. To prepare the Data for the Algorithm, the foreachRDD function was used to prepare the data and run the algorithm on each RDD received in the Datastream. Inside the function, the data inside the RDD was converted to the necessary data frame which Spark uses for machine learning, it was also decided here to drop the station name row, as this variable would not be necessary for time predictions.

A screenshot of a computer

Description automatically generated

Figure 14. Spark Data frame displayed in the command terminal.

Once the data was inside the data frame, a vector assembler then had to be used to prepare the two columns, input and output, required by the machine learning algorithm. All columns except Predicted Arrival Time were added to the input column Features. A Train test split of 70/30 was then used to separate each data frame and the initial Algorithm was then trained on the training set for 3 minutes. The results from the training set during this 3-minute period were very inconsistent and highlighted that Linear Regression was not suitable for the data used. it was decided at this point to change algorithms and make use of a Gradient-Boosted Decision Tree model, this decision was made due to the support provided by spark mlib and due to the Gradient-Boosted algorithms advantage of higher prediction accuracies over using a regular decision tree, as Gradient Boosted combines multiple trees rather than using a singular tree.

### 3.3.7 Tuning the Machine Learning Algorithm

Although Gradient Boosted Trees can provide higher accuracies over individual trees, they are more prone to overfitting and can be more difficult to tune. As such, it was deemed advantageous at this stage to perform cross-validation of the gradient boosted model on separate files containing some previously stored data. Cross-validation would allow for the model to run various combinations of parameters before picking whichever parameter combination performed the best on the testing set. This was performed on three different JSON files, each containing 25 rows of national rail data stored prior. The main parameters tuned in the cross-validator were maxDepth, which defines the maximum depth of each individual tree, and maxIter, which defines the maximum number of iterations for boosting. All three runs of the cross-validator found that a maxDepth of 5 and maxIter of 10 provided the most accurate results.

A picture containing sitting, black, table

Description automatically generated

Figure 15. results of Cross Validation.

# 4.0 Testing & Evaluation

This section will discuss the testing and evaluation approaches which were used in order to answer the research question. To successfully test the algorithm’s accuracy, it was left to run during three time periods when travel update frequency would differ, it was decided to run the algorithm just after midday, just after the UK “rush hour” which is generally thought to be between 4pm and 7pm in the evening, and then later in the evening one hour before most services cease operations, as these 3 time periods have different activity levels and as such, busier times have more running services and likely more updates available. The outcome of each test will be documented, analysed and evaluated at each stage. The findings of the evaluation will then be linked back to the original aims and objectives of the project.

## 4.1 Evaluation Methods

In order to give a detailed analysis and evaluation of the time prediction application, numerous methods were taken into consideration. As the main purpose of this develop and test project was to make arrival time predictions on streams of rail data, along with the consideration that predicted arrival times were already provided in the streams. It was decided that the best evaluation method for this project would be to compare each prediction made to the original time estimates provided. Regression analysis variables such as Root Mean Square Error (RMSE) and R Squared (R^2) were also documented to analyse the “goodness of fit” of the algorithm on the data provided.

## 4.2 How Tests Runs were Documented

Prior to conducting any tests, it was deemed essential to break down the values recorded in the tables of each test run to provide a better understanding of how the data was analysed. The prediction values provided by the algorithm often contained multiple decimal places, where the actual target values would only have two decimal places. As such, it was decided that to give a better comparison of the predictions, rounding the algorithm’s predicted value to two decimal places, meaning the predicted value was the same format as the target value for each test run would be beneficial. The algorithm’s original predicted value, the rounded prediction value, and the observed target value were all provided for each run of tests. The difference column was then calculated by taking the rounded prediction value away from the observed target value. The R-Squared value, which is used to evaluate the spread of data points, was then calculated by squaring the difference values. The Root Mean Square Error (RMSE), which indicates the standard deviation of the prediction errors, was calculated by taking the Square Root of the Sum of all the R-Squared values, divided by the Number of values provided. An example of this equation is given in figure 16.

Figure 16. Equation to Calculate RMSE.

The Root Mean Square Error values, along with the prediction accuracy, which was calculated by counting the number of 0 values in the difference column and then dividing that number by the number of times received, were the main two variables analysed to indicate how well the algorithm fit the data inside the stream and how well it was able to make predictions.

## 4.3 Conducting the First Test Run

The previous section defines how each test table was created and how the values of the columns were calculated in each table, once this was highlighted the testing was then carried out. As mentioned earlier in the report, it was decided that the first test would be conducted just after midday, during a time when the railways would be active at a normal day to day rate of services. The first test run was Initiated at 1:15pm and was initially planned to be run for 140 seconds (2 Minutes 20). During this period, 7 Dstreams were received from the Kafka topic containing 136 time predictions. A summary of the results from each Dstream are given below, with the detailed test results provided in the Appendix.

Summary of the First Test Run Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Interval Number & (Secs) | Number of Times in the Dstream | Number of Correct Predictions | Root mean Square Error | Prediction Accuracy |
| 1 (20) | 20 | 9 | 0.022803509 | 45% |
| 2 (40) | 20 | 13 | 0.036193922 | 65% |
| 3 (60) | 18 | 5 | 0.051908038 | 25% |
| 4 (80) | 18 | 4 | 0.023804761 | 20% |
| 5 (100) | 20 | 17 | 0.003872983 | 85% |
| 6 (120) | 20 | 17 | 0.005477226 | 85% |
| 7 (140) | 20 | 6 | 0.781270894 | 30% |

As shown by the table above, the algorithm had a mixed performance during the first test run. When looking at the higher prediction accuracies, it was noted that most of the time updates here were within the same hour period and some during the same minute period, meaning there was a higher similarity between each item in the stream. The opposite of this was also found when looking at the lowest prediction accuracy, during this interval, there were varied hour periods and it was noted that one of the largest difference values came from an item which was for a service running 9 hours after the testing happened. Although many of the prediction accuracies were low, it was worth noting that some of the RMSE’s of the respective accuracies were also lower, which indicates that the incorrect predictions were still close to the actual prediction values, for example in the very first test interval.

It was also noted early on during this run of tests that the time taken for the algorithm to process and predict the data was longer than the context period of 20 seconds initially set, this could be down to a combination of two influences, the first was the way in which the algorithm makes its predictions. As previously mentioned, Gradient Boosted trees create multiple trees per run rather than a singular tree utilised by regular decision tree models, which means they require more time to make predictions but generally give more accurate results. The second reason, which ties in with the Algorithm, was the execution hardware used for this project. Spark Streaming recommends Octa- Core systems when using real-time data, whereas the execution system used was a Quad-Core, the difference in processing power could limit the speed at which the algorithm is able to create each tree, and as a result extend the delay period between each spark context.

Although the algorithm did provide mixed results during this test period, it was still able to predict more than half of the times received, with an average RMSE of 0.15. While the algorithm performed okay on the first run, it had to be noted again that the time the algorithm took to make predictions was considerably longer than the streaming context time set, and as such had a larger delay period than what would be considered “Real Time”.

Conclusion of Results from First Test Run

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Times Received | Correct Predictions | Average RMSE | Overall Prediction Accuracy |
| 136 | 71 | 0.1321901904285714 | 52.21% |

## 4.4 Conducting the Second Test

After Analysing the Results of the first run, the decision was made to slightly adjust the parameters of the algorithm in the hopes of getting both more accurate and consistent predictions while also shortening the overall time to make the predictions. The maxDepth parameter originally set to 5 was readjusted to 4, meaning each tree created by the algorithm would be shorter and as such wouldn’t take as long to process. Similarly, to the first test the same values were documented and calculated at each interval to then calculate the RMSE and Prediction Accuracy for the interval. To provide more in-depth analysis, the interval number was also increased for this run by one minute, allowing for 3 extra intervals to be calculated. As stated previously, the test was run at the end of the UK rush hour meaning that more services and more updates were anticipated during this time. The test was started at 7:10 pm and a summary of results is provided in the table below. Full documented results are available in the Appendix.

Summary of results from Second Test Run

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Interval Number & (Secs) | Number of Times in the Dstream | Number of Correct Predictions | Root mean Square Error | Prediction Accuracy |
| 1 (20) | 20 | 8 | 0.038209946 | 40% |
| 2 (40) | 20 | 9 | 0.040373258 | 45% |
| 3 (60) | 20 | 10 | 0.089749652 | 50% |
| 4 (80) | 20 | 9 | 0.022248595 | 45% |
| 5 (100) | 20 | 11 | 0.020736441 | 55% |
| 6 (120) | 20 | 10 | 0.131909060 | 50% |
| 7 (140) | 20 | 11 | 0.039874804 | 55% |
| 8 (160) | 20 | 16 | 0.005916080 | 80% |
| 9 (180) | 20 | 8 | 1.152471258 | 40% |
| 10 (200) | 20 | 15 | 0.021330729 | 75% |

As shown by the table above, the results were more consistent during this test run than the first run conducted. When taking a closer look at the results, similarly to the first test it was noted that the highest accuracy interval contained times which were all between the same hour period and some had the same minute period, again highlighting the algorithm seemed to perform better when there were more similarities between the items in the stream. When looking at the lowest percentage accuracy and highest RMSE interval, it was noted that two of the items in the stream were in an hour period different from the rest of the results, and these two predictions were also the furthest from the observed target time, highlighting that while the algorithm was able to pick up on similarities between data items, it struggled to find a relationship between the similar items and the two different items.

When looking at performance, it was noted here that the algorithm did make predictions quicker than the first test run, however there was still a delay period present greater than the 20 second streaming context initially set.

The algorithm was able to provide more consistent prediction accuracies overall compared to the first test run and again was able to correctly predict over half of the times received with an average RMSE of 0.155, however again it was noted here that while the application was providing good predictions, it was again unable to run the algorithm within the streaming context time set, and although the time to process was viewed as less than that of the first tests, it was still a longer delay period than what would be considered a “Real Time” response.

Conclusion of Results from Second Test Run

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Times Received | Correct Predictions | Average RMSE | Overall Prediction Accuracy |
| 200 | 107 | 0.1558430872 | 53.5% |

## 4.5 Conducting the Final Test

As mentioned in a previous section, the aim of the final test run was to see how accurately the algorithm would be able to make predictions on data stream updates which were less frequent and contained less items than those received from the two earlier tests. The results of the earlier tests indicated that the algorithm struggled to make predictions on times which weren’t similar to other times in the stream so it was anticipated that the results of this test would be lower than the other two. However, it was felt that it would be beneficial to analyse the performance of the algorithm during this time period to see if it could make predictions within the streaming context time set. The Test was run at 10:15pm for the same interval length as the second test. A summary of the test run is provided in the table below. The full results of the test are available in the Appendix.

Summary of the Final Test Run Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Interval Number & (Secs) | Number of Times in the Dstream | Number of Correct Predictions | Root mean Square Error | Prediction Accuracy |
| 1 (20) | 2 | 1 | 0.0009 | 50% |
| 2 (40) | 4 | 3 | 0.03 | 75% |
| 3 (60) | 5 | 2 | 0.020493902 | 40% |
| 4 (80) | 3 | 1 | 0.020816660 | 33% |
| 5 (100) | 2 | 2 | 0 | 100% |
| 6 (120) | 4 | 1 | 0.055452683 | 25% |
| 7 (140) | 5 | 3 | 0.018973666 | 60% |
| 8 (160) | 4 | 2 | 0.007071068 | 50% |
| 9 (180) | 5 | 0 | 0.146219014 | 0% |
| 10 (200) | 4 | 3 | 0.145 | 75% |

As shown in the table above, the algorithm was much less consistent during the final testing and often struggled to make predictions on the smaller amounts of data in the streams. The highest accuracy interval once again contained two items in which both the same hour and minute period values were present, showing that the algorithm was able to find similarities between the data inside the stream, However when looking at the lower accuracy values, it was clear that the algorithm was struggling to find connections between the variables in each stream as the times which contained different hour values had the most inaccurate predictions made.

While the algorithm failed to make predictions as consistently and accurately as the previous tests, the prediction times on the smaller streams were calculated much quicker and stayed within the streaming context delay period initially set unlike the previous two tests run.

Conclusion of Results from Final Test Run

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Times Received | Correct Predictions | Average RMSE | Overall Prediction Accuracy |
| 38 | 18 | 0.045 | 47.4% |

## 4.6 Limitations in The Application

This section will evaluate the limitations found during the implementation and testing of the application and evaluate the impacts these had.

### 4.6.1 Hardware

The earliest test run on the prediction application found that the time taken to make predictions was longer than the time set for the streaming context to receive more data after the first prediction was made on the stream. The delay time was put down to a combination of two factors which were also summarised during the first test run. The algorithm chosen to make predictions created multiple trees at each interval rather than single trees which regular decision tree models employ. The extra time needed to for the algorithm to make these trees could be a reason for its prediction delay. The other factor considered which would also impact the speed at which the algorithm could create trees was the hardware of the execution system. As mentioned previously, Spark Streaming recommends Octa Core Processors when making use of real time data, whereas the execution system for this application was a Quad Core system. The lower processing power of the execution system could create a limitation in how quickly the algorithm was able to run, and thus was another potential reason for why the delay period was longer than set.

### 4.6.2 Relationships between data variables

Analysis of the three test runs conducted found that during each run the algorithm was more successful when it was able to find similarities in the data received. When there were less similarities between the data it was noted that worse predictions and higher prediction differences were found, indicating that the algorithm was at times struggling to find relationships between the data variables. For example, if a stream received contained a majority of updates around the time of 7pm but also contained one or two updates for services at 11pm, the algorithm struggled to make accurate predictions for the latter items. This could be a result of some of the items in each datastream being irrelevant when predicting arrival times and could also be a result of a lack of relevant variables in each stream, variables such as the current location of the service being updated or current weather in the area which the service operates could both be variables of relevancy when predicting arrival times which were not available in the feed utilised.

## 4.7 Ethical and Professional Issues

This section will highlight the main ethical, professional, social or legal issues which had to be taken into consideration when creating the prediction application.

### 4.7.1 Legal

As mentioned during the implementation section, code for creating the link to the national rail Darwin Push Port Feed was used from an open source Github repository (Item 57 in References). While the code is open source, it is made clear that any redistribution or modification of the code can only be done under the terms of the General Public License Version 3 or later (62. GNU, 2007). While implementing the base code this license was referred to throughout to ensure the terms were not breached. To adhere to the license for modification, a comment line was added before the import statements to specifically state the date at which the code was first modified, along with a reference to the license followed.

### 4.7.2 Privacy

To access the data available from the Darwin Data Feeds, a developer account had to be created before a unique username and password was generated for the account. This created a privacy issue which had to be taken into consideration during implementation as the credentials used in the connection code to gain access to the rail data were generated for the sole use of the account of the author and thus, could not be left in the final code. To deal with this privacy issue, the username and password were both removed at the end of the testing stage and replaced with a commented link to where new accounts could be created.

# 5.0 Discussion & Conclusion

This section will provide a summary of the overall project including the main outcomes and results. The summary will then allow for a final overall conclusion of the project. Limitations of the project will also be detailed along with any further work which could be investigated in the future.

## 5.1 Final Discussion of Results

The main aim of the project was to identify how well a machine learning algorithm could process and accurately predict real-time arrival updates for UK rail service operations. The use of multiple tests across different activity level periods throughout a service day allowed for thorough investigation into how quickly and accurately the algorithm could predict the updated arrival times. While more than half of the predictions made during the test period were correct and showed that the algorithm could make accurate predictions, the number of inaccurate results was still larger than anticipated and the time between the application receiving new data and making the subsequent predictions on the data was much longer during regular and more intense activity levels than what would be considered reasonable for Real-Time updates. As such, the results showed that the Gradient Boosted Decision Tree algorithm was unable to make accurate, real-time predictions on the stream data used.

## 5.2 Project Limitations and Further Work

Whilst the results of this project were unable to provide the level of accuracy and speed of predictions which would be considered as successful in this research area, there are factors within the project that likely caused limitations on the development of the application and the respective results which were found. The resources provided by one person conducting the development, testing and evaluation stages of the project, along with the time restrictions for the project’s completion both identify this as smaller scale research and as such, limitations which could impact the conclusion are not uncommon. A longer timeline combined with more resources and more powerful execution systems could produce faster, more accurate prediction results and thus provide further insight into how well suited the algorithm was in this research area. The application made use of a singular data feed from the Nation Rail services and while the data provided the necessary information for arrival and departure times, there was little detail of other meaningful variables such as the current weather in the service area, faults on the lines used by the service, and the current location of the operating train.

Therefore, the results can only be presented as conclusive to the specific data feed and variables used and cannot present conclusions for predictions made had other relevant factors been taken into consideration.

## 5.3 Conclusion

The aim of this project was to identify how well a chosen machine learning algorithm could process and predict real time travel updates for rail services. The application created was used to store and process a National Rail data feed before making arrival time predictions based on the other variables given in the feed. While this was a smaller scale project, successful completion of research and development allowed for the implementation of an application which could receive data from a datastream source and make time predictions based on the information received.

Overall, the project allowed for the research question to be thoroughly tested by running the application at 3 different time periods during the day where different activity levels were found for the rail service. The use of multiple intervals also allowed for a more in-depth analysis of how well the algorithm was making predictions during each of the test runs.

# 6.0 References

1. IDC: The premier global market intelligence company. (2019). IDC Forecasts Worldwide Spending on the Internet of Things to Reach $745 Billion in 2019, Led by the Manufacturing, Consumer, Transportation, and Utilities Sectors. [online] Available at: https://www.idc.com/getdoc.jsp?containerId=prUS44596319 [Accessed 18 Oct. 2019].

2. Juniper Research. (2019). IoT Connections to Grow 140% to Hit 50 Billion By 2022. [online] Available at: https://www.juniperresearch.com/press/press-releases/iot-connections-to-grow-140pc-to-50-billion-2022?utm\_source=juniperpr&utm\_campaign=internetofthings18pr1&utm\_medium=email [Accessed 18 Oct. 2019].

3. Marr, B. (2018). How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read. [online] Forbes.com. Available at: https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#4917609360ba [Accessed 18 Oct. 2019].

4. Cisco. (2018). Cisco Global Cloud Index: Forecast and Methodology, 2016–2021 White Paper. [online] Available at: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html#\_Toc503317525 [Accessed 20 Oct. 2019].

5. Sharma, S.K. Wang, X. (2017). Live Data Analytics With Collaborative Edge and Cloud Processing in Wireless IoT Networks, in IEEE Access, vol. 5, pp. 4621-4635. [online] Available at: http://ieeexplore.ieee.org.gcu.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=7882669&isnumber=7859429 [Accessed 20 Oct. 2019].

6. Expert System. (2017). What is Machine Learning? A definition - Expert System. [online] Available at: https://expertsystem.com/machine-learning-definition/ [Accessed 20 Oct. 2019].

7. BBC Academy. (n.d.). The history of machine learning. [online] Available at: https://www.bbc.com/timelines/zypd97h [Accessed 23 Oct. 2019].

8. Rouse, M. (2019). What is Turing Test? A definition from WhatIs.com. [online] SearchEnterpriseAI. Available at: https://searchenterpriseai.techtarget.com/definition/Turing-test [Accessed 23 Oct. 2019].

9. www.javatpoint.com. (n.d.). History of Machine Learning - Javatpoint. [online] Available at: https://www.javatpoint.com/history-of-machine-learning [Accessed 23 Oct. 2019].

10. IDC: The premier global market intelligence company. (2018). Worldwide Spending on Cognitive and Artificial Intelligence Systems Forecast to Reach $77.6 Billion in 2022, According to New IDC Spending Guide. [online] Available at: https://www.idc.com/getdoc.jsp?containerId=prUS44291818 [Accessed 23 Oct. 2019].

11. McKinsey & Company. (2018). Visualizing the uses and potential impact of AI and other analytics. [online] Available at: https://www.mckinsey.com/featured-insights/artificial-intelligence/visualizing-the-uses-and-potential-impact-of-ai-and-other-analytics [Accessed 23 Oct. 2019].

12. Google Developers. (n.d.). Recommendation Systems Overview |  Google Developers. [online] Available at: https://developers.google.com/machine-learning/recommendation/overview/types [Accessed 24 Oct. 2019].

13. Bulanov, A. (n.d.). Machine Learning and Data Analytics in the Travel Industry. [online] Django Stars Blog. Available at: https://djangostars.com/blog/benefits-of-the-use-of-machine-learning-and-ai-in-the-travel-industry/ [Accessed 25 Oct. 2019].

14. Chawla, S. (2019). 7 Successful Applications of AI & Machine Learning in the Travel Industry. [online] Hackernoon.com. Available at: https://hackernoon.com/successful-implications-of-ai-machine-learning-in-travel-industry-3040f3e1d48c [Accessed 25 Oct. 2019].

15. BBC News. (2017). Reality Check: When is a train on time?. [online] Available at: https://www.bbc.co.uk/news/world-asia-42024020 [Accessed 25 Oct. 2019].

16. the Guardian. (2019). One in three British rail services run late under tightened standards. [online] Available at: https://www.theguardian.com/uk-news/2019/sep/19/one-in-three-british-rail-services-run-late-under-tightened-standards [Accessed 25 Oct. 2019].

17. Dataportal.orr.gov.uk. (2019). Passenger rail service complaints | ORR Data Portal. [online] Available at: https://dataportal.orr.gov.uk/statistics/passenger-experience/passenger-rail-service-complaints/ [Accessed 27 Oct. 2019].

18. Dataportal.orr.gov.uk. (2020). Passenger Rail Performance | ORR Data Portal. [online] Available at: <https://dataportal.orr.gov.uk/statistics/performance/passenger-rail-performance> [Accessed 27 February 2020].

19. 2019. Passenger Rail Performance 2019-20 Q2 Statistical Release. [pdf] Office Of Rail And Road. Available at: https://dataportal.orr.gov.uk/media/1629/passenger-performance-2019-20-q2.pdf [Accessed 27 February 2020].

20. 2019. Rail Factsheet. [pdf] Department for Transport, p.1. Available at: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment\_data/file/851082/rail-factsheet-2019.pdf [Accessed 27 February 2020].

21. 2018. Rail Factsheet. [pdf] Department of Transport, p.1. Available at: <https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment\_data/file/761352/rail-factsheet-2018.pdf> [Accessed 27 February 2020].

22. DataFlair. 2019. Advantages And Disadvantages Of Machine Learning Language - Dataflair. [online] Available at: https://data-flair.training/blogs/advantages-and-disadvantages-of-machine-learning/ [Accessed 28 February 2020].

23. Agrawal, A., n.d. Highlights The Advantages And Disadvantages Of Machine Learning. [online] Cisin.com. Available at: <https://www.cisin.com/coffee-break/Enterprise/highlights-the-advantages-and-disadvantages-of-machine-learning.html> [Accessed 28 February 2020].

24. TechVidvan. 2020. Exploring The Advantages And Disadvantages Of Machine Learning - Techvidvan. [online] Available at: <https://techvidvan.com/tutorials/advantages-and-disadvantages-of-machine-learning/> [Accessed 28 February 2020].

25. Stewart, M., 2019. The Limitations Of Machine Learning. [online] Medium. Available at: https://towardsdatascience.com/the-limitations-of-machine-learning-a00e0c3040c6 [Accessed 28 February 2020].

26. Rebala, Gopinath. et al. (2019) An Introduction to Machine Learning. 1st ed. 2019. [Online]. Cham: Springer International Publishing. Available at: https://link-springer-com.gcu.idm.oclc.org/book/10.1007%2F978-3-030-15729-6 [Accessed 27 Oct. 2019].

27. Sas.com. (n.d.). Machine Learning: What it is and why it matters. [online] Available at: https://www.sas.com/en\_gb/insights/analytics/machine-learning.html [Accessed 27 Oct. 2019].

28. Bashier, E., Khan, M. and Mohammed, M. (2016). Machine Learning. [online] O’Reilly | Safari. Available at: https://learning.oreilly.com/library/view/machine-learning/9781315354415/ [Accessed 27 Oct. 2019].

29. Ciaburro, G. (2017). MATLAB for Machine Learning. [online] O’Reilly | Safari. Available at: https://learning.oreilly.com/library/view/matlab-for-machine/9781788398435/ [Accessed 28 Oct. 2019].

30. Shobha, G. and Rangaswamy, S. (2018). Handbook Of Statistics Chapter 8 - Machine Learning. [online] ScienceDirect. Available at: https://www.sciencedirect.com/science/article/pii/S0169716118300191 [Accessed 28 February 2020].

31. Google Developers. (n.d.). What is Clustering?  |  Clustering in Machine Learning. [online] Available at: https://developers.google.com/machine-learning/clustering/overview [Accessed 28 Oct. 2019].

32. Bonaccorso, G. (2018). Mastering Machine Learning Algorithms. [online] O’Reilly | Safari. Available at: https://learning.oreilly.com/library/view/mastering-machine-learning/9781788621113/?ar= [Accessed 28 Oct. 2019].

33. Huang, K. (2018). Introduction to Various Reinforcement Learning Algorithms. Part I (Q-Learning, SARSA, DQN, DDPG). [online] Medium. Available at: https://towardsdatascience.com/introduction-to-various-reinforcement-learning-algorithms-i-q-learning-sarsa-dqn-ddpg-72a5e0cb6287 [Accessed 31 Oct. 2019].

34. AltexSoft. 2018. Data Science And AI In The Travel Industry: 12 Real-Life Use Cases. [online] Available at: https://www.altexsoft.com/blog/datascience/data-science-and-ai-in-the-travel-industry-9-real-life-use-cases/ [Accessed 28 February 2020].

35. ye, Q., Zhang, Z. and Law, R., 2008. Sentiment Classification Of Online Reviews To Travel Destinations By Supervised Machine Learning Approaches. [online] Available at: https://www.sciencedirect.com/science/article/abs/pii/S0957417408005022#! [Accessed 1 March 2020].

36. Julio, N., Giesen, R. and Lizana, P., 2016. Real-Time Prediction Of Bus Travel Speeds Using Traffic Shockwaves And Machine Learning Algorithms. [online] Available at: https://www.sciencedirect.com/science/article/abs/pii/S0739885915300895 [Accessed 1 March 2020].

37. Cui, L., Yang, S., Chen,F. et al. Int. J. Mach. Learn. & Cyber. (2018) 9: 1399.[online] Available at: https://doi-org.gcu.idm.oclc.org/10.1007/s13042-018-0834-5[Accessed 31 Oct. 2019]

38. Mahdavinejad, M., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P. and P.Sheth, A. (2017).Machine learning for internet of things data analysis: a survey. [online] Digital Communications and Networks Volume 4, Issue 3, Pages 161-175. Available at: https://www.sciencedirect.com/science/article/pii/S235286481730247X?via%3Dihub#bib56 [Accessed 31 Oct. 2019]

39. Siryani, J. Tanju, B. Eveleigh, T. J. (2017). A Machine Learning Decision-Support System Improves the Internet of Things’ Smart Meter Operations. [online] IEEE Internet of Things Journal, vol. 4, no. 4, pp. 1056-1066. Available at: http://ieeexplore.ieee.org.gcu.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=7964681&isnumber=8004446 [Accessed 31 Oct. 2019]

40. Vanajakshi, L. and Rilett, L., 2007. Support Vector Machine Technique For The Short Term Prediction Of Travel Time. [online] Available at: https://ieeexplore-ieee-org.gcu.idm.oclc.org/document/4290181/authors#authors [Accessed 31 October 2019].

41. Yin, T., Zhong, G., Zhang, J., He, S. and Ran, B., 2017. A Prediction Model Of Bus Arrival Time At Stops With Multi-Routes. [online] Available at: https://www.sciencedirect.com/science/article/pii/S2352146517306889#! [Accessed 27 February 2020].

42. Perera, S. (2018). What Is Stream Processing? A Gentle Introduction - DZone Big Data. [online] dzone.com. Available at: https://dzone.com/articles/what-is-stream-processing-a-gentle-introduction [Accessed 2 Nov. 2019].

43. Tzoumas, K. and Friedman, E. (2016). Introduction to Apache Flink. [online] O’Reilly | Safari. Available at: https://learning.oreilly.com/library/view/introduction-to-7apache/9781491977132/ [Accessed 5 Nov. 2019].

44. Spark.apache.org. n.d. Spark Streaming - Spark 2.4.5 Documentation. [online] Available at: https://spark.apache.org/docs/latest/streaming-programming-guide.html [Accessed 28 February 2020].

45. Guller, M. (2016). Big Data Analytics with Spark: A Practitioner’s Guide to Using Spark for Large-Scale Data Processing, Machine Learning, and Graph Analytics, and High-Velocity Data Stream Processing. [online] O’Reilly | Safari. Available at: https://learning.oreilly.com/library/view/big-data-analytics/9781484209646/ [Accessed 5 Nov. 2019].

46. Spark.apache.org. (n.d.). MLlib | Apache Spark. [online] Available at: https://spark.apache.org/mllib/ [Accessed 6 Nov. 2019].

47. Nalya, A. and Jain, A. (2014). Learning Storm. [online] O’Reilly | Safari. Available at: https://learning.oreilly.com/library/view/learning-storm/9781783981328/?ar= [Accessed 6 Nov. 2019].

48. DataFlair. 2018. Apache Storm Vs Spark Streaming - Feature Wise Comparison - Dataflair. [online] Available at: https://data-flair.training/blogs/apache-storm-vs-spark-streaming/ [Accessed 28 February 2020].

49. Liu, A. (2016). Apache Spark Machine Learning Blueprints. [online] O’Reilly | Safari. Available at: https://learning.oreilly.com/library/view/apache-spark-machine/9781785880391/?ar= [Accessed 6 Nov. 2019].

50. 2020. Anaconda Python/R Distribution Version 3.7 - Free Download. [online] Available at: https://www.anaconda.com/distribution/#download-section [Accessed 8 February 2020].

51. Nationalrail.co.uk. n.d. National Rail Enquiries - Darwin Data Feeds. [online] Available at: <https://www.nationalrail.co.uk/100296.aspx> [Accessed 10 February 2020].

52.Hicks, P., Dawson, R., Moffatt, R. and Wood, T., 2019. Stomp-Client-Python. Github.[Code for Data feed connection] Available at : [**https://github.com/openraildata/stomp-client-python**](https://github.com/openraildata/stomp-client-python)

53. 2020. Apache Spark. Apache. Version 2.4.5 [Software] Available at: <https://spark.apache.org/> [Accessed 11 February 2020].

54. 2020. Apache Kafka. Apache. Version 2.13 [Software] Available at: https://kafka.apache.org/downloads.html [Accessed 12 February 2020].

55. Kozlovski, S., 2017. Thorough Introduction To Apache Kafka™ | Hacker Noon. [online] Hackernoon.com. Available at: https://hackernoon.com/thorough-introduction-to-apache-kafka-6fbf2989bbc1 [Accessed 12 February 2020].

56. 2020. Apache Zookeeper. Apache. Version 3.5.7 [Software] Available at <https://zookeeper.apache.org/releases.html> [Accessed 12 February 2020].

57. Aslam, S., 2017. Installing Apache Zookeeper On Windows. [online] Medium. Available at: <https://medium.com/@shaaslam/installing-apache-zookeeper-on-windows-45eda303e835> [Accessed 13 February 2020].

58. Aslam, S., 2017. Installing Apache Kafka On Windows. [online] Medium. Available at: https://medium.com/@shaaslam/installing-apache-kafka-on-windows-495f6f2fd3c8 [Accessed 13 February 2020].

59. Maramreddy, R., 2019. Guide To Install Apache Spark On Windows—Spark Setup For Beginners. [online] KnowledgeHut Blog. Available at: <https://www.knowledgehut.com/blog/big-data/how-to-install-apache-spark-on-windows> [Accessed 15 February 2020].

# 7. Figures

Figure 1. IBM (n.d.). *Arthur Samuels Playing checkers on the 701*. [image] Available at: https://www.ibm.com/ibm/history/ibm100/images/icp/A138918I23240Y22/us\_\_en\_us\_\_ibm100\_\_700\_series\_\_checkers\_\_620x350.jpg [Accessed 22 Oct. 2019].

Figure 2. Wagoner, A. (2018). *google Play HomePage with "For You" section*. [image] Available at: https://www.androidcentral.com/sites/androidcentral.com/files/styles/xlarge/public/article\_images/2018/05/google-play-search-2.jpg?itok=twcTnF8T [Accessed 24 Oct. 2019].

Figure 3. ORR. (2018). Punctuality at recorded station stops bar graph. [image] Avaliable at:

<https://dataportal.orr.gov.uk/media/1629/passenger-performance-2019-20-q2.pdf> [Accessed 27 February 2020].

Figure 4. Apache. (2020). Spark Streaming Architecture. [image] Available at:

<https://dataportal.orr.gov.uk/media/1629/passenger-performance-2019-20-q2.pdf> [Accessed 25 October 2019]

Figure 5. Initial Structure of the data.

Figure 6. Reformatted DataStream

Figure 7. Starting Apache Zookeeper using “zkserver”.

Figure 8.  Command to run Apache Kafka.

Figure 9. Running Apache Kafka.

Figure 10. Creating the Kafka Topic “Main\_Test”.

Figure 11. The Contents of the Kafka Topic.

Figure 12. Running Apache Spark.

Figure 13. Output of Datastream.

Figure 14. Spark Data frame displayed in the command terminal.

Figure 15. results of Cross Validation.

Figure 16. Equation to Calculate RMSE.

# 8.0 Appendix

## 8.1 Analysis of Requirements

Before attempting to implement the machine learning application, it was important to analyse the core functionality of the application and evaluate exactly what the application would need to do to successfully answer the research question. The main application requirements can be broken down into two types: Functional & Non-Functional. The literature Review and Technology Assessment highlighted the importance of choosing a machine learning algorithm for specific problem areas and datasets. As such, analysing the requirements of the algorithm and documenting exactly what the algorithm looks to achieve with the data was paramount in the selection process of the implemented algorithm.

### 8.1.1 Functional Requirements

**Process Data**

Before Machine Learning can be used, the application will need to be able to ingest and process the datastream chosen, this requirement will be achieved by making use of a stream processing platform which will be discussed in a later section.

**Predicting Times**

The Main Requirement of this application is to be able to predict the arrival times of trains at their designated stations using the other data provided in each update. As mentioned previously, the algorithm will be initially run to analyse its suitability before being tuned and further in-depth testing being conducted.

### 8.1.2 Non-Functional Requirements

**Can the algorithm make accurate arrival time predictions**

The Main Non-Functional Requirement for this application is whether it can make accurate time predictions and if so, how accurate these predictions are. This requirement will be successfully evaluated in the Testing section where the accuracy results from the three-time intervals will be documented, and their overall accuracies collated.

## 8.2 Design

Upon the completion of the requirements analysis for the core application functionality, the next stage in the process was to decide upon a suitable interface design for the machine learning application. Due to the final application having no real end users, and the application not requiring any external user input or interference after the algorithm starts running, It was decided that displaying the input data and results of the algorithm in a command terminal would be a satisfactory interface and would allow for more time to be spent on the tuning of the machine learning algorithm, which in turn would improve the likelihood of higher prediction accuracies being achieved.

### 8.2.1 Application Architecture

To help visualise the main flow of the application and the technologies required, a basic diagram was constructed.

Processed Data Stream

Data Source

Data Storage

Stream Processing Platform

Command Terminal

Data Stream

Machine Learning Algorithm

Unprocessed Data Stream

Algorithm Results

Figure A. Basic application architecture and workflow.

As shown in the Diagram above, the main workflow for the application is:

* Data is received from Source
* The stream of data is stored in the chosen storage method.
* The Stream Processing Platform then reads the stream from the storage and processes the data.
* Once the Data is processed, the chosen machine learning algorithm is then trained and tested on the datastream to predict the arrival time of each train at the corresponding station.
* After the algorithm has finished its operations on the datastream, the results of the prediction will be sent to and displayed on the command terminal

The datastream will also be printed to the command terminal at other stages throughout implementation in the form of comment lines in the main code to help improve the understanding and following of how the data is formatted and structured at each stage.

## 8.3 Testing

### 8.3.1 Test Run 1 Results

Test Run 1 Interval 1 (20 Times)

Test Run 1 Interval 2 (20 Times)

Test 1 Interval 3 (18 Times)

Test 1 Interval 4 (18 Times)

Test 1 Interval 5 (20 Times)

Test 1 Interval 6 (20 Times)

Test 1 Interval 7 (20 Times)

### 8.3.2 Test Run 2 Results

Test 2 Interval 1 (20 Times)

Test 2 Interval 2 (20 Times)

Test 2 Interval 3 (20 Times)

Test 2 Run 4 (20 Times)

Test 2 Run 5 (20 Times)

Test 2 Run 6 (20 Times)

Test 2 Run 7 (20 Times)

Test 2 Run 8 (20 Times)

Test 2 Run 9 (20 Times)

Test 2 Interval 10 (20 Times)

### 8.3.3 Test Run 3 Results

Test 3 Interval 1

Test 3 Interval 2

Test 3 Interval 3

Test 3 Interval 4

Test 3 Interval 5

Test 3 Interval 6

Test 3 Interval 7

Test 3 Interval 8

Test 3 Interval 9

Test 3 Interval 10