Predicting medium-term train delays using machine learning

Student Name: Dominic White

Supervisor Name: Noura Al-Mouyabed

Submitted as part of the degree of BSc Computer Science to the

Board of Examiners in the Department of Computer Sciences, Durham University

Abstract —

A Context / Background

Trains are a popular form of public transport. The perceived likelihood and severity of train delays important both for customers' satisfaction and to various business stakeholders, and, and so a system predicting train delays in the medium-term would improve both customer satisfaction and operating profits. There are currently no solutions to this problem in the literature, nor in practice, although extensive research has been carried out into real-time delay prediction.

B Aims

The aims of this project are twofold. The primary aim is to identify a suitable binary classification model for predicting medium-term train delays. The secondary, supporting aim is to develop a new train delay dataset to train this model on.

C Method

Using historical schedule data, movement data, location data, and weather data from a variety of sources, a high-quality dataset is constructed for use in machine learning. A variety of classification models are trained and tested, and the best-performing is optimised. The effect of including exogenous weather data is investigated.

D Results

An entirely new high-quality dataset has been constructed, comprising 5.2 million rows and 23 columns. Of the classification models tested, a random forest achieved the greatest F_1-measure of 0.76. Evaluation using repeated stratified $10-$fold cross validation resulted in a ROC_{AUC} of 0.77.

E Conclusions

This project explores a relatively unstudied area in train delay prediction. The dataset developed is novel and high-quality, and may prove of use to other researchers. The developed random forest performed well, and together they form a solid foundation for future work in this area.

Keywords — Machine learning, train delay, delay prediction, extract-transform-load, ETL, classification, regression

I INTRODUCTION

1.76 billion rail journeys were made in the 2018 - 2019 financial year[1]. Of these journeys, approximately 12.2% were delayed by more than 5 minutes. Train delays impose a huge cost on passengers and operators by contributing to the inefficiency of train operations (Van Oort, 2011). In 2006 – 07, for example, delays cost a minimum of £1 billion in terms of time lost to passengers (**?**). Furthermore, under a scheme known as 'Delay Repay', Train Operating Companies (TOCs) are automatically obliged to refund 50% of the cost of passengers' tickets if a train is between 30 – 60 minutes late and 100% if it is more than 60 minutes late. 4.6 million fares were repaid in 2018 - 2019.

So it is no understatement to say that train delay is a serious problem. The issue is compounded for passengers, who tend to perceive punctuality as worse than it is for three reasons: passengers tend to recall delayed trains over punctual trains; more passengers travel on late trains than punctual trains (as passengers accumulate while waiting); and operators avoid early running, so late trains are not balanced out by early trains (**?**, p. 130). Furthermore, passengers expect trains to run on time as TOCs are considered to be in greater control of their environment.

This mismatch in perception places great importance on reducing the impact of delays on passengers. So: what if passengers could know in advance how likely it is that a might train is delayed? The aim of this project is not to answer this questions, but to see whether doing so is possible: can train delays be predicted in advance?

Systems already exist for predicting real-time train delay; the problem is well-studied. The majority are analytical (**?**). Such models are online, i.e. updated as information on train movements becomes periodically available. At the other end of the spectrum is the optimisation problem of train scheduling, which is also very well studied. Between the two extremes lies is a poorly-defined region, the medium-term timeframe, neither online nor offline, lower-bounded by the availability of forecasts of exogenous data and upper-bounded by the times trains depart their origins. This is the timeframe this project focuses on. Knowing that a train will be delayed while on it is of little help to most passengers, but knowing it will be a week in advance is.

This project will explore data-driven models, which have recently been gaining traction in the field. Rete Ferroviaria Italiana (RFI), the Italian railway manager, uses a (real-time) model (**?**); Deutsche Bahn (DB), a German railway operator, recently commissioned a comprehensive study into a (real-time) model suitable for their network (**?**). This work aims to develop both a train delay dataset and a data-driven binary classification model to see whether medium-term train delays can be predicted.

A Background

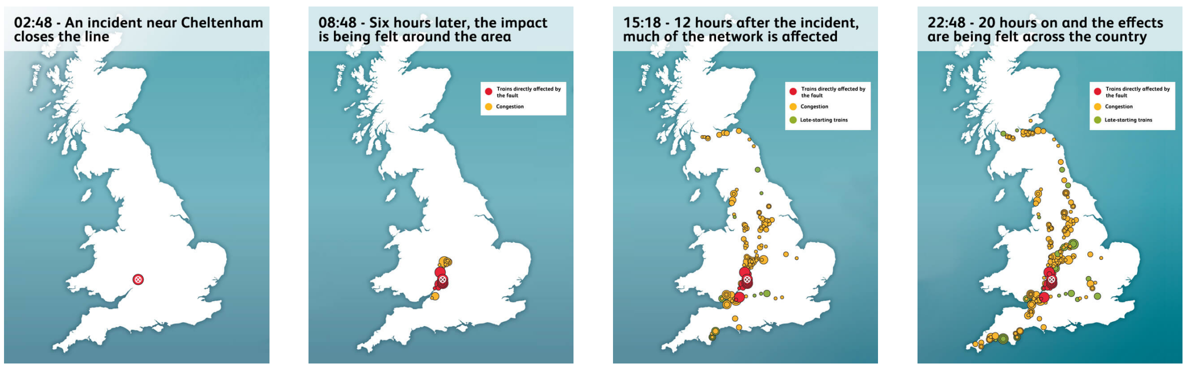
Although precise terminology differs, the literature agrees that there are two principal classes of delay (**?**): primary (exogenous) and secondary (knock-on, consecutive). A primary delay is caused by external stochastic disturbances (**?**), of which there are many potential causes (**?**, **?**). They may be classified into three categories: infrastructure, weather, engineering, and a catch-all (other). This is presented in Table **??**. Of the delays in 2006 - 2007, 42% were caused by infrastructure faults, 38% by TOCs, and the remaining 20% by events such as adverse weather, fatalities and vandalism.

[1]https://dataportal.orr.gov.uk/statistics/usage/passenger-rail-usage/. Data is published quarterly.

Table 1: Causes of primary delays

Class	**Causes of delay**
Infrastructure	Signals and point failures; malfunctioning equipment
Weather	Severe heat; flooding; landslips; leaves; snow and ice
Engineering	Scheduled maintenance; repairs
Other	Prolonged alighting and boarding times; fatalities; accidents

A secondary delay is generated by operational conflict (Cerreto *et al.*, 2016). Primary delays induce a cascade of secondary delays of other trains, which must wait for tracks to be clear, crews to be in the right place, platforms available, and so on. This is visualised in Figure **??**. Predicting secondary delays is very difficult; it is the source of the complexity of real-time models. As Lessan *et al.* (Lessan *et al.*, 2019) note, some secondary delay factors are predictable and controllable: most are neither. This project therefore focuses on predicting primary delays, which depend on exogenous data.


02:48 - An incident near Cheltenham closes the line
08:48 - Six hours later, the impact is being felt around the area
15:18 - 12 hours after the incident, much of the network is affected
22:48 - 20 hours on and the effects are being felt across the country
Trains directly affected by the fault
Congestion
Late-starting trains

(a) 02:48 (b) 08:48 (c) 15:18 (d) 22:48

Figure 1: A visualisation of delay propagation. A single incident at 02:48 causes congestions, delays and cancellations across most of the country for the next 24 hours. Red dots are trains directly affected by the fault; yellow, congestion; and green, late-running trains.

Data is exogenous if it is independent of other input data but the output is dependent upon it. In the context of primary delay prediction, the more of the causes of primary delay that can be incorporated into a model, the better it will perform. Models tend to use a combination of infrastructure (**?**, **?**, **?**), weather (**?**, **?**, **?**, **?**), expert opinions (**?**, **?**) and engineering work (**?**). Information about passenger flows and railway asset conditions is also useful (**?**). Where directly relevant exogenous data is unavailable, it is possible to use proxies (Harris, 1992), such as:

- train length (as a proxy for the number of doors to manage, and passenger demand)
- distance covered (as a proxy for the likelihood of encountering track defects and other technical / operational problems)
- the previous number of stops (as a proxy for cumulative delay resulting from passengers alighting and boarding)
- the age of the motive power unit (as a proxy for reliability)

- track occupation (as a proxy for capacity utilisation of the railway, and thus the likelihood of delays propagating)

B Objectives

The research question proposed is: *can medium-term train delays be predicted*? To address this question, the objectives for this project were divided into three categories: minimum, intermediate, and advanced.

The minimum objective of this project is to construct a high-quality dataset incorporating a variety of exogenous data. The purpose of this objective is twofold. Such a dataset is, first and foremost, a prerequisite for machine learning, and there are none publicly available. There are few objective criteria for what makes a good dataset, save that it ought to be clean, logically structured, and complete, with only relevant fields included.

The intermediate objective is to develop a binary classification model to predict whether a train will be delayed. A variety of classification models will be selected, with selection informed by a literature review.

The advanced objective is to build upon the intermediate objective by extensively testing and fine-tuning the best-performing model using cross validation and hyperparameter optimisation.

II RELATED WORK

In the section, we present work related to our research question: *Can medium-term train delays be predicted?*. Although there has been significant academic effort into producing real-time train delay prediction (TDP) systems, research on medium-term TDP is rather sparse. Fortunately, there is considerable overlap between the two timeframes. Solutions to the problem of real-time TDP use a variety of machine learning regression models, with the field tending towards ensembles / hybrids and random forests. The inclusion of exogenous data pertaining to weather and infrastructure is also popular. There are many classifications of TDP models, based on scope, model type, and solution methods (**?**). Generally, approaches may be divided using two axes: whether they are offline or online, and whether they use analytical or data-driven models. As this project is concerned with latter, the former is discussed only briefly. Online models are updated with new data as it becomes available; data-driven models are trained using historical data.

A Analytical approaches

An analytical model is "primarily quantitative or computational in nature and represents the system in terms of a set of mathematical equations that specify parametric relationships and their associated parameter values as a function of time, space, and/or other system parameters" (**?**). Current state-of-the-art train delay prediction systems use analytical models (**?**): "static rules, built by experts on railway infrastructure, and based on classical univariate statistics".

Simplistic early models, such as those developed by (**?**) make overly restrictive assumptions about railway operations, e.g. no overtakes are allowed, departure times are uniformly distributed, and that the speed of each train is unique and constant. Subsequent work in this area has largely relaxed these assumptions by including factors such as overtakes, different speeds, priority systems, and uncertainties associated with train departure time (**?**, **?**). Further models have become increasingly advanced, incorporating stochastic approximation (**?**), and the impacts of dispatching strategies on train delays and passenger waiting time.

Much work has also been done which blurs the line between analyitical and data-driven models (**?**, **?**) by using stochastic models based on Bayesian networks and trained on historical data to predict delays, with considerable success. However, there is a fundamental limit on the complexity of analytical models, and as the field of machine learning has matured, the applicability of data-driven models to TDP has been increasingly well explored.

B Online / dynamic data-driven approaches

Most real-time TDP systems are online. The first comprehensive attempt at developing a TDP system was made by Oneto et al. (2016). Subsequent papers (**?**, **?**) have expanded the proposed models in scope and performance. The authors consider a rail network a graph, where nodes represent checkpoints and edges tracks connecting them. A train follows an itinerary composed of N_c checkpoints characterised by an origin, a terminus, and intermediate locations such as stops and transits. A schedule is modelled as a time series forecast problem, with performance at previous checkpoints used to predict performance at subsequent checkpoints. A model for each train is trained using historical delay data. The authors tested random forests (RF), Extreme Learning Machines (ELM), and Kernel Methods (KM) and found that their RF performed twice as well as current state-of-the-art TDP systems. The inclusion of weather further improved accuracy by approximately 10%. Their model was very computationally expensive, however, requiring approximately 600,000 models to be trained daily across the entire rail network.

The authors then generalise their work to produce a dynamic data-driven TDP system (**?**), with performance tuned using Thresholdout, which reduces overfitting. They compare the performance of two implementations of shallow and deep ELMs. This work laid the groundwork for their most recent paper (**?**), which combines an experience-based model (EBM) and multiple RFs into a hybrid model (HM). The EBM uses operators' knowledge and experience of the network to inform features. As Martin (2016) notes, in real-world train operations, delay prediction relies heavily on the experience and intuition of a local dispatcher, rather than a network-wide computational instrument. The HM is a decision tree where each leaf is a RF. Trains are directed to the appropriate RF by similarity, eliminating the need for a model for every train. A new leaf is added each time a new train is seen that belongs to a previously unexplored branch of the decision tree. The RF regressor in the leaf is trained based on all the past train movements in that leaf. Trains older than 3 months are 'forgotten', to keep the size of the model constrained. Furthermore, this model naturally handles the mercurial nature of train schedules, which are both released periodically and revised constantly. The model requires only 10 days' of data after a new schedule comes into effect to reach optimal accuracy, with excellent performance also noted for outliers. The HM offers the best trade-off between accuracy and computational requirements, with superior results to all current models.

A similarly impressive model was developed by Lessan *et al.* (**?**), working closely with Deutsche Bahn (DB). The authors used 3.25 years' of data from DB and incorporated a wide range of exogenous data, with approximately 350 features for operational (i.e. currently running) trains and 70 for non-operational trains. The authors tested support vector regression (SVR) but settled on RFs. Three models were used for operational trains: an online RF, mesoscopic simulation, and kernel regression. Two were used for non-operational trains: an offline RF and mesoscopic simulation. The authors found an accuracy of over 80% in predictions within a 60-

minute horizon, a considerable improvement in real-time forecasts, but found that beyond that horizon predictions were only marginally better than the schedule.

C Offline / static data-driven approaches

Static models are not updated with data as it comes available. Early attempts used simple neural networks (**?**) and were essentially proofs-of-concept. The use of neural networks was further investigated by Yaghini *et al.* (**?**) using data from Iranian Railways to predict the late arrival of passenger trains. Their model performed better to alternatives such as decision trees and logistic regression. The authors also tested the impact of various encoding schemes.

Subsequent work is largely exploratory, with no clear sense of progression as described in the previous section. Papers investigate a wide variety of models, such as Fuzzy Petri Nets (FPNs) (**?**), support vector regression (SVM) (**?**) logistic regression (**?**) and gradient-boosted random forests (**?**).

Milinkovic *et al.* (**?**) presented the first use of an FPN. The authors explored two separate FPNs. In the first, expert knowledge was used to define fuzzy sets and rules. In the second, an Adaptive Network Fuzzy Inference System (ANFIS) was trained on historical delay data and then replicated in an FPN. Both were then tested with real data from a Belgrade station node. The ANFIS-FPN produced results within 5% of actual delay values for a subset of the data; slightly worse performance was observed for the expert-defined FPN.

Markovic *et al.* (**?**) presented the first use of SVM. They found it outperformed artificial neural networks. Data for the analysis was again collected from Serbian Railways. The paper used the expert opinions of dispatchers to estimate the likelihood of multiple factors along a rail line, such as single-tracking, junctions, or the number of stations, causing a delay, using the Delphi method to obtain a final estimate. A strong correlation was found between expert opinions and train delays. The focus was on developing a functional relationship between delays and infrastructure so the effect of infrastructure improvements on delays can be predicted and valued.

Wang and Work (**?**) used data from Amtrak to develop two vector autoregression models: one offline, the other online. The offline model improved the RMSE of predicted delays by 12%; the online model, by 60%.

Wang and Zhang (**?**) present a relatively simple gradient-boosted regression trees (GBRT) model. The model was trained on a three-month dataset of weather, train delay, and train schedule records. Their model can make predictions up to 10 days' in advance, the limit of the weather forecast system used, but performed poorly, which the authors attribute to the limited size of their dataset. This paper most closely resembles the objectives of this project.

III SOLUTION

In this solution we present our solution to constructing a dataset and identifying a suitable binary classification model. The `Python` programming language was used throughout, with `NumPy` for numerical operations, `Pandas` for data manipulation and analysis, `scikit-learn` for machine learning, and `Matplotlib` for graphing.

A Dataset construction: the Extract-Transform-Load (ETL) pipeline

Central to the importance of machine learning is a high-quality labelled dataset. Unfortunately, there are no publicly available datasets for train delay. Correspondence between authors of previous papers (**?**, **?**) proved unfruitful, so it was necessary to construct a new dataset, which became the minimum objective of this project. Dataset construction is a complex task. It falls under the purview of the Extract-Transform-Load (ETL) pipeline[2]. This pipeline extracts data from sources, transforms them into a usable format, and loads them into storage for future operations. It is a very time-consuming task; "it is not uncommon for a project team to spend as much as 50% to 70% of project effort on ETL functions" (**?**, p. 284). This proved the case with this project, so we dedicate considerable space to explaining each stage in the pipeline, and how it was realised in the context of this dissertation.

First, we identified suitable data sources for the train data itself (schedules, location, and movement) and the three classes of primary delay (weather, infrastructure, and engineering) identified in Section **??** (**??**). The UK rail infrastructure manager Network Rail (NR) opened its feeds[3] to developer usage in 2011; they are presented in Table 1.

Table 2: NR data feeds and their use in this project. I = investigated, U = used, N = not used.

Acronym	**Description**	**Frequency**	**Usage**
BPLAN	Train planning data, including locations and sectional running times.	Twice a year	I
CORPUS	Location reference data.	Monthly	U
MOVEMENT	Train positioning and movement event data.	Real-time	U
RTPPM	Performance of trains against the timetable, measured as the percentage of trains arriving at their destination on-time.	Once a minute	N
SCHEDULE	Daily extracts and updates of train schedules from the Integrated Train Planning System (ITPS).	Overnight each night	U
SMART	Train describer berth offset data used for train reporting.	Monthly	N
TD	Train positioning data at a signalling berth level.	Real-time	N
TSR	Details of temporary reductions in permissible speed across the rail network.	Weekly (on Friday)	I
VSTP	Train schedules created via the VSTP process	Real-time	I

A.1 Schedules: SCHEDULE

Schedules are available as CIFs[4] from the SCHEDULE feed. A full CIF – a database snapshot – is released every Friday morning, with update CIFs released every morning. Each update must be applied to the latest full CIF to maintain a correct database. Each file contains train schedules,

[2]Sometimes a different order is used: Extract-Load-Transform (ELT). The sole difference is where the transformation stage takes place. The distinction is largely academic here, so ETL is used throughout.

[3]https://www.networkrail.co.uk/who-we-are/transparency-and-ethics/transparency/open-data-feeds/

[4]Common Interface File

metadata, and associations. We ideally would have used the VSTP feed as well, which refers to trains scheduled up to 48 hours before they are due to run. However, the feed was not archived in the repository from which we sourced the rest of our data.

A.2 Location: CORPUS (and NaPTAN)

CORPUS is used in conjunction with the National Public Transport Access Node (NaPTAN) database, a nationwide system for uniquely identifying all points of access (nodes) to public transport in the UK. Only rail stations are, naturally, of concern here. A large variety of codes are used to refer to locations in the UK rail network; they are presented in Table 2.

A.3 Movement: TD, TRUST, and Darwin

The TD feed provides information about the position of trains through a network of berths. A berth usually represents a signal. We discarded TD as too low-level for this project. There are two movement systems currently in by NR: Train Running Under System TOPS (TRUST) and Darwin. TRUST is a NR system used for monitoring the progress of trains and tracking delays in the UK. Darwin uses both TRUST and TD for real-time data, and also incorporates Darwin workstations, Customer Information Systems (CIS), and internal messaging systems.

TD feeds into TRUST, and TRUST into DARWIN. Darwin provides more comprehensive information. While the objectives of this project were being defined, it was undecided whether to focus on real-time or medium-term train delays. The real-time timeframe would have necessitated the use of Darwin, whereas medium-term could have used either. To keep options open, Darwin was therefore used. Darwin messages contain a lot of extraneous information, but fundamentally each movement message contains a timestamp and a location, either a STANOX or TIPLOC. Both CORPUS and NaPTAN are therefore necessary to ensure a message can be geolocated.

A.4 Weather: MIDAS, CEDA, and Datapoint

The primary source of weather data is the Met Office Integrated Data Archive System (MIDAS). MIDAS is a database of land and marine surface observations, collected from 1853 to the present day, by the Met Office station network. MIDAS offers several datasets. The most comprehensive is the Hourly Weather Observation Data, which contains meteorological values measured on an hourly time scale. These observations include 104 fields , though many are for quality control, or too specific to necessitate inclusion. Station data is available from the Centre for Environmental Data Analysis (CEDA). Each station is geolocated by latitude and longitude. For this project to be of practical use, trained models must be applicable to unseen data. Unseen train data are simply train schedules; unseen weather data are forecasts. Forecasts are available from Met Office Datapoint, a service allowing access to freely available Met Office data feeds. They may be obtained as 3-hourly site-specific forecasts up to 5 days' in advance. There are 10 forecast available fields.

A.5 Infrastructure: BPLAN and the Train Planning Network Model

The Train Planning Network Model is used by the Integrated Train Planning System (ITPS). We were unable to integrate either of these datasets, as there was too little pubicly available

Table 3: UK rail network location codes.

	CORPUS	NaPTAN	Description
STANOX	Y		Station Number. First two digits are the geographic area. Can refer to non-station locations such as sidings and junctions. Numbers run broadly north-to-south.
UIC	Y		
CRS / NRS / 3ALPHA	Y	Y	Used primarily to identify stations and on seat reservation labels.
TIPLOC	Y	Y	Timing Point Location. Relates to points used in deriving train schedules. A station often has multiple TIPLOCs if it consists of multiple groups of platforms on different lines.
NLC	Y		National Location Code. Identifies locations on the railway. Used for retailing and accounting purposes.
NLCDESC	Y		A description of the NLC.
NLCDESC16	Y		A description of the NLC.
ATCO		Y	
EASTING, NORTHING		Y	Geographic Cartesian coordinates for a point. Uses EPSG:27700[5], the British National Grid / Ordnance Survey system.
NAME		Y	The name of the location.

documentation.

A.6 Engineering: TSR

TSR provides proxy details of engineering works. It was not archived in the repository from which we sourced the rest of our NR data, so we could not make sure of it.

B Extraction

Extraction is the process of collecting data, often from multiple different sources, and moving it to a *staging area*. Some basic validation also takes place at this stage. The extraction process of the each data source is presented in Table 3.

C Transform

Transformation is the process of applying a set of functions to extracted data, to clean, map, validate, and consolidate datasets, with the aim of making the data conform to a uniform schema. There is a lot of overlap between transformation and preprocessing, the stage just prior to training. The difference lies in ease of repetition: transformation would, ideally, be performed once, as operations are typically expensive, operating on vast quantities of unstructured and semi-structured data, where preprocessing, using heavily optimised code in NumPy and Pandas, can

Table 4: Extraction methods for each data source

Data source	**Notes**
Schedule	Schedule data is available from Peter Hicks' website . Both full and update extracts are downloaded as gzipped files.
Movement	Darwin data is also available from Peter Hicks' website . Each day is a bzip2-compressed tar file containing 1440 files, one for each minute in the day. Each file comprises XML messages. Each minute is extracted in memory and written to one CSV for each day / file.
Location	NaPTAN is available as a zipped CSV from the Department for Transport (DfT). Only the railway nodes are extracted and saved. CORPUS is available from NR as a gzipped JSON. Only the TIPLOC information is extracted and saved.
Weather	Available for download as a CSV file, via FTP from CEDA, though headers must be downloaded separately. Station data is also available from CEDA as a KMZ file[6]. The KMZ file is parsed using pykml and station metadata using BeautifulSoup to produce a CSV.

be iteratively developed.

C.1 Transforming schedules

Train scheduling is a very complex problem. As we mentioned, an update CIF is released every morning. It contains deletions, amendments, and new schedules, which must be applied to the currently-held schedule database. After each update is applied, the schedules for that day are written to file. Full schedules are released weekly, so errors cannot propagate for more than 7 days. A CIF contains 9 types of record. We used 5 of these records to build, for each train, a single row containing metadata, such as the TOC responsible for the train, location data, such as the origin, destination, and number of stops, and datetime data, such as the scheduled departure (from the origin) datetime and the scheduled arrival (at the terminus) datetime.

We discarded 2 types of record to reduce complexity: CRs (change en route) and AAs (associations). An association represents some dependency between two trains, such as crew or engines. They have the potential to be a rich source of data - is, for instance, a train with more associations more likely to be delayed? - but including them would necessitate a much more complex model than the simpler tabular format suitable for machine learning. A change en route indicates that some metadata has changed during a train's journey. Some changes are typically small - a change in seating rules, for instance, or whether reservations are required - and including them would again necessitate a more complex model and greater space requirements. As we believed the impact of CRs would be small, we ignored them.

C.2 Transforming weather

As mentioned previously, there is a need to map from the MIDAS format to DataPoint format. There are 10 available DataPoint fields for forecasts, of which only 7 could be meaningfully mapped to equivalent MIDAS fields. For several, this was a simple unit conversion. For visibility and weather type, predefined codes were used to establish a map between the two formats[7].

C.3 Transforming movement

Each day of train movements is a file roughly 1.5GB in size. This file also contains a variety of other messages: forecasts, schedules, station messages, alarms, warnings, and so on. We converted each train movement into a simple record comprising the type, train, location, and time the movement took place. DARWIN does not include the date at which the movement took place, so this was inferred from other metadata. We also dropped duplicate messages.

C.4 Transforming location

This stage was relatively simple. We merged NaPTAN and CORPUS, and calculated the closest geographical weather stations in MIDAS for future use..

D Load

Load is the process of storing data in a format accessible for future usage. It was a more complex in this project, involving the merging of the aforementioned datasets. We joined together schedule and movement data and constructed a target column, `delayed`, using the the difference between the scheduled arrival time and actual arrival time. We discarded erroneous rows and dropped irrelevant columns. We then joined location data and weather data.

E Model selection

When searching for candidate models, we looked mainly at those with generally good performance at classification tasks, rather than those identified in Section II, though there was some overlap (**?**, **?**). This is because, as discussed earlier, the majority of research into TDP focuses on the real-time timeframe, not the medium-term, and so are not applicable. As this process is exploratory, we chose a wide range of models. As ensembles have proven popular, we focused on those. We also selected several simple linear models, anticipating poor performance, for comparison. For ease of testing, we only used models available in `scikit-learn`. 10 models were chosen for the initial round: an AdaBoost classifier, a decision tree classifier, a Gaussian NB classifier, a gradient-boosting classifier, a histogram-gradient-boosting classifier, a linear Support Vector Classifier (SVC), logistic regression, a random forest classifier, a ridge classifier, and a stochastic gradient descent (SGD) classifier.

[7]https://www.metoffice.gov.uk/services/data/datapoint/code-definitions

IV RESULTS

In this section, we present the results of our ETL pipeline, our classification model, and our hyperparameter optimisation. We also discuss the metrics used to analyse results. We trained and tested each of the models identified in Section III (E) on the dataset produced by our ETL pipeline, after some transformations and resampling. We further analysed the best-performing model to identify important features, and subjected it to an hyperparameter optimisation process to improve performance. The experiment environment system specification is outlined in Table 4.

Table 5: Experiment environment system specification

Component	**Details**
Processor	Intel Core i5-6990K 3.6 GHz
GPU	NVIDIA GeForce 980 TI 8GB
RAM	32 GB 2400 MHz DIMM
OS	Windows 10

A ETL

The output of our ETL pipeline is a single CSV, `1.1 GB` in size, containing approximately 5.2 million rows and 47 columns, over a period of one year (`2018-04-01` to `2019-04-031). Approxim`
and one for `cos`, such that each is encoded as a point on unit circle. For compatibility between different models, we one-hot encoded categorical variables and scaled numeric variables to Gaussian. We later relax this requirement. We also resampled the dataset. Approximately 10% of rows are positive (i.e. delayed). This causes poor performance: a model can learn to always predict negative (i.e. not delayed) to trivially achieve approximately 90% accuracy. To remedy this we investigated different ways to balance the dataset, and chose to oversample the minority class (i.e. delayed) using SMOTE (**?**) and randomly undersample the majority class (i.e. not delayed) to achieve a 50:50 ratio. The final form of our dataset is a $(1178694, 283)$ sparse matrix.

B Binary classification

To evaluate our models, we will produce accuracy (Acc), precision (Prec), and recall (Rec) statistics using the number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) classifications of our models. We also use the F_β-measure (Eq. 1). The F_β-measure is the weighted harmonic mean of precision and recall, which captures the natural trade-off between them. A β of 1 indicates that Prec and Rec are weighted equally importantly.

$$F_\beta = (1+\beta)\frac{precision \cdot recall}{\beta^2 \cdot (precision + recall)} \tag{1}$$

We trained each of the models identified in Section III (E) on the same 70% of the dataset. No limits were imposed on memory, CPU usage, or time. Two potential models were discarded for incompatibility with the sparse matrix: GaussianNB and HistGradientBoostingClassifier. We then tested the remaining models on the remaining 30% of the dataset, and computed a number of metrics. The results are displayed in Table 5.

Table 6: A full statistical analysis of each model after training on the test dataset

Model	**Acc**	**Prec**	**Rec**	F_1
AdaBoostClassifier	0.67	0.62	0.67	0.67
DecisionTreeClassifier	0.68	0.62	0.69	0.68
GradientBoostingClassifier	0.68	0.63	0.67	0.68
LinearSVC	0.67	0.61	0.67	0.67
LogisticRegression	0.67	0.61	0.67	0.67
MLPClassifier	**0.73**	0.67	0.73	0.73
RandomForestClassifier	**0.73**	**0.70**	**0.74**	**0.74**
RidgeClassifier	0.67	0.61	0.67	0.67
SGDClassifier	0.66	0.60	0.66	0.66

Table 5 shows that the random forest is the superior model for solving this binary classification problem, with the greatest recall, precision, accuracy and F_1-measure. The MLPClassifier (a multi-layer Perceptron classifier, a type of neural network) also performed well. Interestingly, the simple models achieved comparable performance to the random forest in a fraction of the training time - under 10 seconds, in the case of the LinearSVC. Based on these results, we investigated random forests further.

C Random forests

Random forests (RFs) are ensembles consisting of multiple random decision trees. Each tree is built on a random subset of the original data; at each tree node, a random subset of features are randomly selected to generate the best fit. They do not require feature scaling or categorical feature encoding, so we simplified the pipeline by removing these features. We then identified five hyperparameters likely to impact performance and used randomised search combined with 3-fold cross validation to optimise them; the results are presented in Table 6.

Table 7: Random forest hyperparameter optimisation

Hyperparameter	**Function**	**Default**	**Optimum**
`n_estimators`	Number of trees in the forest	100	200
`max_features`	The number of features to consider when looking for the best split	auto	sqrt
`max_depth`	The maximum depth of the tree	None	10
`min_samples_split`	The number of samples required to split an internal node	2	10
`min_samples_leaf`	The minimum number of samples required to be a leaf node	1	2
`bootstrap`	Whether bootstrap samples are used when building trees	True	False

We achieved a modest improvement in performance through this process, with all metrics showing an improvement of between 2.70% and 8.57% (in the case of precision), as presented in Table 7. Random forests are typically difficult to fine-tune, so this small improvement was expected.

Table 8: A statistical analysis of the tuned random forest

Model	**Acc**	**Prec**	**Rec**	F_1
RandomForestClassifier	0.76	0.76	0.76	0.76

Finally, we evaluated the model using repeated stratified 10-fold cross validation, which is "generally a better scheme, both in terms of bias and variance, when compared to regular cross-validation" (**?**), and selects folds so that each contains roughly the same class distribution in the original dataset (in this case, the re-sampled dataset). The results are plotted in Figure **??** as a ROC_{AUC} curve. ROC curves visualise the trade-off between the true positive rate (TPR) and false positive rate (FPR) for a predictive model using different probability thresholds. A larger area under the curve (AUC) is better. We obtained a mean ROC_{AUC} of 0.77 ± 0.00. This is because our dataset was perfectly balanced, and sufficiently large that there was essentially no difference between the folds.

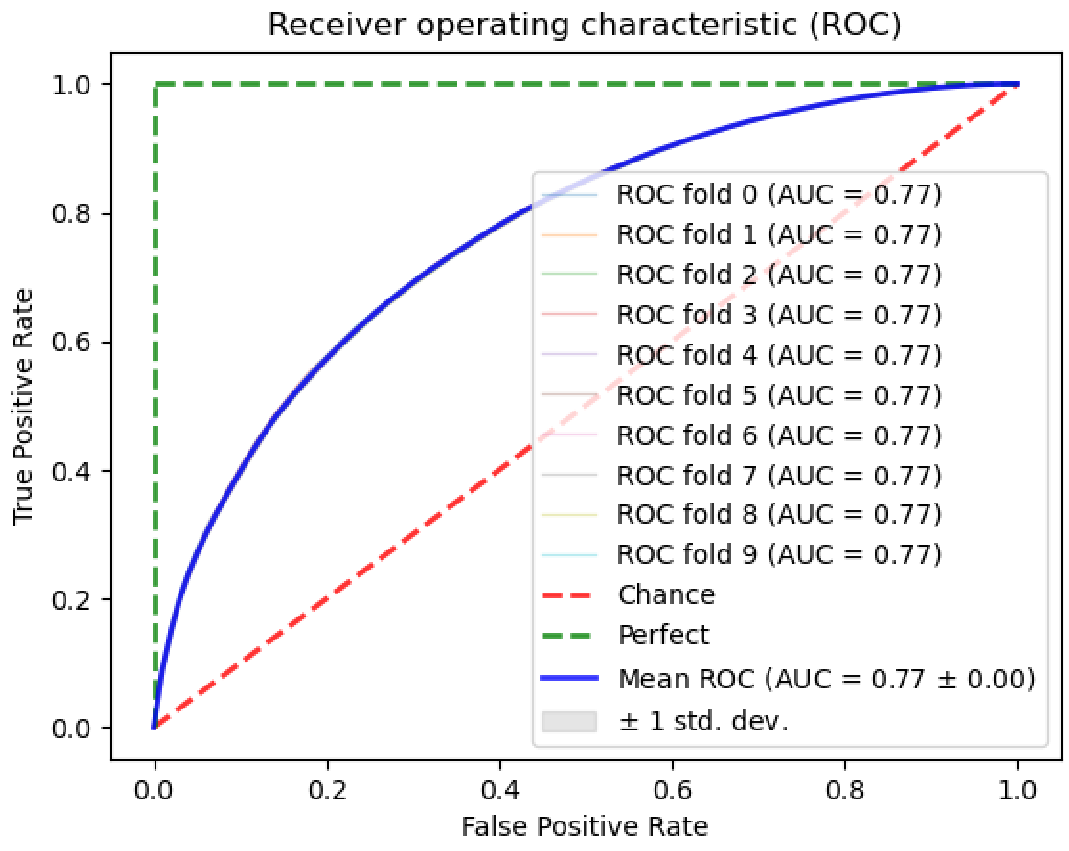


Figure 2: A ROC_{AUC} graph.

D Exogenous data: weather

Finally, we tested the effect of the inclusion of weather data on our model. We joined the MIDAS dataset with our current dataset. This join resulted in an approximate 35% reduction in the number of rows available. We then modified our pipeline to transform the weather features appropriately and re-tested our tuned random forest model. We found a modest improvement in performance; the metrics are presented in Table **??**.

Table 9: A statistical analysis of the tuned random forest with weather data

Model	**Acc**	**Prec**	**Rec**	F_1
RandomForestClassifier	0.76	0.74	0.77	0.77

V EVALUATION

In this section, we evaluate the strengths and weakness of our solutions, reflecting upon the original research question: *Can medium-term train delays be predicted?*.

A ETL

Our approach to constructing a dataset was somewhat haphazard. Railways are a very complex subject matter, and with no expert at hand, this complexity was an intimidating prospect. There was little to be done in the face of obscure file formats and codes save to push on, and comprehension of the data consumed an inordinate amount of time. It is industry knowledge that data preparation accounts for roughly 80% of the work of a data scientist, and this certainly proved the case in this project. Minor changes to any stage necessitated time-consuming re-runs of scripts that processed over to `1 TB` of data, which added to the overhead.

It also took some time to settle on the research question. Was it better to tread the better-worn path of real-time delay prediction, and accept the enormous overhead of modelling a railway network, or to explore medium-term prediction? We chose medium-term prediction at a fairly late stage at the project, and as a result needless effort was expended ensuring the codebase was compatible with both timeframes. This is most obviously manifest in the movement dataset used. TRUST is briefly mentioned as an alternative, simpler, system to DARWIN. However, using it would have precluded the use of forecasts. When investigating real-time delays, we discovered that a key performance indicator of any potential system is how predictions compare to the those made by the current system (in this case, DARWIN). By the time we had chosen to explore medium-term delays, it was too late to revert back to TRUST, and so we incurred an additional overhead there.

As discussed previously, predicting primary delays is dependent on the inclusion of exogenous data. Many of the relevant data feeds were unfortunately unavailable in the archive used and so could not be included. As this dissertation progressed, a new repository, archiving all feeds, was established by OpenTrainTimes[8]. However, switching would have incurred too much work, both from an extraction, but particularly a transformation, perspective, at a late stage of the project. We regrettably could not use several other important data sources. Infrastructure, in particularly, would have likely yielded far better performance; we mentioned earlier that 42% of delays can be attributed to infrastructure faults. Even the inclusion of weather was a challenge. The MIDAS dataset contained 192 columns and was poorly documented.As the project never developed to a working application, it proved unnecessary to convert MIDAS data to Datapoint data.

B Binary classification

Our approach to selecting a binary classification model proved sound. Initial early testing revealed that the class imbalance of our dataset seriously hampered performance, but this issue was neatly resolved by a combination of resampling methods. The simple alternative approach, to balance the classes by discarding the an appropriate proportion of the majority, would have reduced the size of our dataset by 80%, with a corresponding decrease in model performance. Collecting more data might have made this viable; we chose a years' worth both to preserve the

[8]https://networkrail.opendata.opentraintimes.com/

cyclical nature of the dataset and to limit resource consumption. At time of writing, approximately 2 years' worth of data is available online, and the codebase is written such that using this data in future experiments would be easy.

The linear models performed poorly, as expected, with approximately 63% accuracy. We anticipated random forests and gradient-boosted decision trees would perform well, but were surprised by the multi-layer perceptron, which came in second place. This perhaps indicates that investigating alternative neural network architectures would be a profitable route of inquiry.

While `scikit-learn` is designed to be easy to use, we found that we pushed the API to its limits. We spent much time configuring encoders and pipelines for compatibility with our models, but the initial investment was definitely worthwhile. Unfortunately this precluded us from integrating feature importance analysis into our experiments, but we performed small-scale tests with random forests and found that the majority of important features were derived from the origin-departure and destination-arrival datetimes. Several features proved close to irrelevant, and further investigation into this would have been useful, as reducing the number of features improves model training time. It is even possible that the majority of the metadata taken from the SCHEDULE feed could be excluded, and instead only the schedules present in DARWIN used, which would have greatly reduced the complexity of the ETL pipeline.

C Hyperparameter optimisation

Our approach to hyperparameter optimisation proved sound. We did not anticipate great improvement, as random forests are typically resistant to fine-tuning, and were only able to identify 5 parameters to optimise, but achieved a roughly 4% improvement nevertheless. We did not test particularly large values for `n_estimators` - the best optimal value was the maximum, 200 - so further increasing this would likely improve performance further. As the number of estimators increases, however, the training time of the model increases, and as we were running so many cross-validation iterations, a cap was necessary.

D Exogenous data: weather

The inclusion of weather data did not improve performance as much as we anticipated. Oneto *et al.* found a 10% increase in performance when including similiar weather data (**?**). It is likely that the conversion from MIDAS to DataPoint, which proved unnecessary, resulted in errors in the weather type, which we expected to be the most important variable: the majority of values were ”Light Rain”, which, while typical of the UK, are likely incorrect.

VI CONCLUSION

In this project we have demonstrated how a novel, high-quality train delay dataset may be constructed from a variety of complex publicly available sources. To the best of our knowledge, this is the first time these sources have been utilised in this fashion.

Using this dataset, we have produced a binary classification model as a s solution to the problem of medium-term train delay prediction. To achieve this, 8 classifiers were tested and their performance evaluated. The most performant model, a random forest, was subject to further testing in which we systematically modified hyperparameters to improve performance. Our final classification model achieved an accuracy of 76% and an ROC_{AUC} of 0.77.

A suitable direction for future work would be the inclusion of more exogenous data, such as infrastructure, and a more complex model to better capture the secondary delays caused by interactions between trains. A multi-layer perceptron obtained comparable results to our final random forest model, so it is possible that alternative architectures such as Deep Neural Networks (DNN) would have yielded superior results. It would also be easy to extend the work in this project to regression, to predict not only whether a train will be delayed but by how much.

We discussed earlier that the main rationale behind this project is the potential benefits it could yield for passengers. To make this project directly useful for consumers, a system could be developed that subscribes to real-time schedule and weather data to predict live whether a given train will be delayed.