TÉCNICO LISBOA

# Predicting Rail Travel Duration using Machine Learning

**Emma Rose Dennis-Knieriem**

Thesis to obtain the Master of Science Degree in

## Data Science and Engineering

Supervisors: Prof. Mário Jorge Costa Gaspar da Silva
Dr. Bruno Ricardo Leite Ribeiro

### Examination Committee

Chairperson: Prof. Name of the Chairperson
Supervisor: Prof. Mário Jorge Costa Gaspar da Silva
Members of the Committee: Prof. Name of First Committee Member
Dr. Name of Second Committee Member
Eng. Name of Third Committee Member

**June 2024**

This work was created using LaTeX typesetting language
in the Overleaf environment (www.overleaf.com).

# Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

# Acknowledgments

I thank my teachers–Mrs. Bates, Ms. Welchman, PJ, Ms. Agna, Ms. Totty, Janet Sensei, Mary Ellen, Sally, Becky, and my Bubby– whose kindness, patience, and dedication helped shape my curiosity as well as my academic and personal development.

I thank Professor Rose and Professor Willis for their mentorship and for teaching me how to identify systems and patterns in the real world.

I thank Ms. Hale, Ms. Walker, and Professors Baumer, Crouser, and Sandstede, who taught me with mathematical and statistical rigor, and doing so thoughtfully and with humanity.

I thank Bruno and Professor Mário for their guidance and encouragement throughout this dissertation.

I thank my friends for their good humor, thoughtful conversations, and being here for me. I thank Claudia for helping me stay on track, and Diane for bothering me into staying on track.

Most especially, I am grateful to my wonderful moms for their unconditional love and support. Not knowing what I study has never stopped them from being incredibly proud of it. *Something with math, maybe? Why is that "y" wearing a hat?*

# Abstract

This dissertation contributes to the advancement of predictive modelling in the domain of rail transportation by providing a tailored solution for predicting travel durations on regular services. The proposed model offers a practical tool for improving service reliability and enhancing the overall passenger experience in rail travel. The study begins by collecting extensive datasets from a platform managed by the Federal Office of Transport (FOT) in Switzerland, a comprehensive repository of transportation data, encompassing various parameters affecting rail travel duration. Subsequently, regression models including tree-based methods are employed to develop predictive models. The results demonstrate the effectiveness of the proposed machine learning model in accurately predicting travel durations for passenger trains. Moreover, insights gained from the analysis shed light on the critical factors influencing travel duration variability, thereby offering valuable implications for operational planning and resource allocation within rail transportation systems.

# Keywords

Predictive Modeling, Regression Modeling, Rail Travel Prediction, Time Series Forecasting

# Resumo

Esta dissertação contribui para o avanço da modelação preditiva no domínio do transporte ferroviário ao fornecer uma solução personalizada para prever a duração das viagens em serviços regulares. O modelo proposto oferece uma ferramenta prática para melhorar a fiabilidade do serviço e a experiência geral do passageiro nas viagens ferroviárias. O estudo partiu da recolha de extensos conjuntos de dados de um repositório compreensivo de dados das viagens de comboio, do Office féderal des transports (OFT) na Suíça, abrangendo vários parâmetros que afetam a duração das mesmas. Foram a seguir desenvolvidos modelos preditivos de regressão, incluindo métodos baseados em árvores. Os resultados evidenciam a eficácia do modelo de aprendizagem automática proposto na previsão precisa da duração das viagens de comboios de passageiros. A análise esclarece os fatores críticos que influenciam a variabilidade da duração das viagens, oferecendo assim implicações valiosas para o planeamento operacional e a afectação de recursos nos sistemas de transporte ferroviário.

# Palavras Chave

Modelação Preditiva, Regressão, Previsão de Viagens de Comboio, Previsão de Séries Temporais

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **AA** | Actual Arrival |
| **AD** | Actual Departure |
| **ANN** | Artificial Neural Networks |
| **CSV** | Comma Separated Values |
| **dt** | datetime |
| **ETA** | Estimated Time of Arrival |
| **ETD** | Estimated Time of Departure |
| **FOT** | Swiss Federal Office of Transportation |
| **GB** | Gradient Boosting |
| **ID** | Identity |
| **KNN** | K-Nearest Neighbors |
| **LLM** | Large Language Model |
| **LR** | Linear Regressor |
| **NaN** | Not a Number |
| **NaT** | Not a Time |
| **NADIM** | National Data Networking Infrastructure Mobility |
| **NN** | Neural Network |
| **ODMCH** | Open Data Platform Mobility Switzerland |
| **RF** | Random Forest Regression |

**RIF** Rail Infrastructure Fund

**RMSE** Root Mean Squared Error

**RSS** Residual Sum of Squares

**SKI+** System Tasks Customer Information Plus

**SVR** Support Vector Regressor

**XGB** eXtreme Gradient Boosting

# 1

# Introduction

## Contents

Since the advent of the first steam locomotives in the early nineteenth century, railroads have evolved into vital infrastructure, now commonly used for cargo and transportation around the world. Millions of people rely on trains every day for commuting to work and school, so it is imperative that these services run smoothly and precisely. Many people prefer trains to buses or cars for their predictability and avoidance of traffic issues. Some people use public transport for environmental reasons. Compared to travelling by car, CO2 emissions can be reduced by up to 42% by buses and up to 73% by train [1]. When commuting, passengers rely on consistency. Late trains need to arrive within only a few minutes to meet the standards of the public. The times people are willing to wait vary by region depending on a country's infrastructure and standard of living.

## 1.1 Objectives

Accurately predicting a train's arrival time will increase efficiency in rail travel, both for the rail service and for consumer satisfaction. With large datasets, models can be trained to predict train arrival times with high accuracies. One such organization that allows access to its transportation data is the Open Data Platform Mobility Switzerland (ODMCH). Using this publicly available dataset, models were built to more accurately predict arrival times. While this dissertation focuses on rail travel within Switzerland only, the objective is to develop a prediction model that can be used for other countries' rail services, providing that the data collected has the same information used from the Swiss data. With more accurate arrival prediction time, passengers can save time on commuting and do so with more certainty, while rail services can better allocate workers, save on energy costs, and create more efficient schedules, lessening rail congestion and thereby increasing safety.

## 1.2 Contributions

The data were taken from the Swiss Federal Office of Transportation (FOT) and the System Tasks Customer Information Plus (SKI+) [2].

The Swiss FOT publishes transport data daily. While this dissertation focuses only on trains, the data also include information about bus, ship, and other methods of transportation [3]. Importantly, "These evaluations should be interpreted from the perspective that this does not involve any real actual data, but tends to involve the last known forecast data or just simply target data. For instance, at some stations, departure times are calculated only on the basis of the arrival time and stop time" [3].

This dissertation resulted in the prediction of train arrival times using several different machine learning models. This holds value not only for the Swiss transport system under whose data these models were developed, but for other transportation networks that aspire to adopt more accurate predictions,

especially rail systems. These models are flexible in that they can be applied to other data with only basic information: identification of stops, and arrival and departure times.

The Swiss FOT data contains 21 variables, all in Swiss German. For ease of understanding and consistency, they will appear in this dissertation as I translated them into English (see Figure 3.1).

## 1.3 Methodology

To assess the accuracy of the data, margins of 1, 3, and 5 minutes were chosen, reflecting acceptable waiting times for passenger transport in Switzerland, where long waits are uncommon. For each margin, the number of rows that fell within the specified threshold (as well as the category ”More than 5 minutes”) was calculated to evaluate the punctuality of the transport system.

Three models were employed: Linear Regressor, Decision Tree Regressor, and XGB Regressor. For each of these models, the training features were the departure day, hour, and minute, as well as label-encoded previous and current stops. The training target was the delta, defined here as time of travel between stations, calculated in seconds.

## 1.4 Organization of the Document

This dissertation is organized as follows: Chapter 1 serves as an executive report and explains what led me to this dissertation. Chapter 2 provides background on time series analyses and the Swiss train system, introduces related work on train prediction strategies, and gives information about learning models and which models were selected. Chapter 3 introduces the data’s origin and cleaning methods. The architecture of the analysis follows, including evaluation metrics and model configurations. In Chapter 4 are the model performance evaluations and comparisons. Finally, Chapter 5 concludes with a summary of the findings, limitations, and how the work can be extended in the future.

# 2

# Background and Related Work

## Contents

## 2.1 Time Series Forecasting

Time series forecasting is used to make predictions about future events based on historical data. It uses patterns such as seasonality and trends to create more accurate predictions later on. Important applications include weather forecasting [4], predicting the spread of epidemiological outbreaks [5], and economic patterns such as inflation [6].

Time series analysis can also be applied to predicting the arrival and departure times of trains, famous for their timetables, and sometimes infamous for being running behind schedule. Occasional rail services for cargo and passengers alike want to travel as efficiently as possible. Commuters often have limited tolerance to wait for these services [7]. Therefore, it is important that rail services have the most accurate timetable predictions to increase transportation efficiency, consumer satisfaction, and overall rail safety.

One such study based on time series to predict passenger train times is "Online prediction of arrival and departure times in each station for passenger trains using machine learning methods" by Shekoofeh Vafaei and Masoud Yaghini [8]. In this 2024 article, Vafaei and Yaghini cite other papers that have predicted arrivals and departures on train time series data in Table 1 of their paper, included here in Figure 2.1.

Prior papers and the status of current paper among them.

| Author (s) | Prediction techniques | Case study | Real time | Increasing the accuracy | Dependent variable | Different models for stations | Prediction times |
|---|---|---|---|---|---|---|---|
| Li et al. [1] | RF | ✓ | ✓ | × | Arrival and departure delays | × | × |
| Shi et al. [2] | Combining XGBoost and BO | ✓ | × | ✓ | Arrival delays | × | × |
| Yaghini et al. [3] | ANN | ✓ | × | × | Arrival delays | × | × |
| Murali.et al. [14] | Simulation-based technique | ✓ | × | × | Arrival delays | × | × |
| Buijse et al. [19] | Deep learning | ✓ | ✓ | ✓ | Arrival time | × | ✓ |
| Kosolsombat & Limprasert [20] | RF | × | × | ✓ | Arrival time | × | × |
| Chen & Rilett [21] | Multiple linear regression models | ✓ | ✓ | × | Arrival time | × | × |
| This study | ANN, RF, GB, XGB | ✓ | ✓ | ✓ | Arrival and departure time | ✓ | ✓ |

**Figure 2.1:** Table 1 of Vafaei and Yaghini, 2024. This study cites previous studies predicting train arrival and departure times, as well as arrival and departure delays. The prediction techniques used among these studies are ANN, RF, GB, XGB, BO (Bayesian Optimization), simulation, deep learning, and Linear Regression (LR). Six out of the seven other papers referenced were case studies, and three out of the seven increased the accuracy of the dependent variable for that study. These seven papers referenced by Vafaei and Yaghini can be found in this dissertation's bibliography [9] [10] [11] [12] [13] [14] [15].

Vafaei and Yaghini use Artificial Neural Networks (ANN), Random Forest Regression (RF), Gradient Boosting Regression (GBoost), and eXtreme Gradient Boosting Regression (XGB) to predict arrival and departure times at 5 stations along a unidirectional route within the Iranian Railways network.

The four models are employed at each prediction time described by Table 5 of Vafaei and Yaghini, 2024, included here in Figure 2.2.

Prediction times and target variables in each of these times.

| Prediction time | Train Position | Target variables |
|---|---|---|
| $T_1$ | Before leaving Tehran Station | $AD_1$, $AA_2$, $AD_2$, $AA_3$, $AD_3$, $AA_4$, $AD_4$, $AA_5$ |
| $T_2$ | After leaving Tehran station | $AA_2$, $AD_2$, $AA_3$, $AD_3$, $AA_4$, $AD_4$, $AA_5$ |
| $T_3$ | Before leaving Garmsar station | $AD_2$, $AA_3$, $AD_3$, $AA_4$, $AD_4$, $AA_5$ |
| $T_4$ | After leaving Garmsar station | $AA_3$, $AD_3$, $AA_4$, $AD_4$, $AA_5$ |
| $T_5$ | Before leaving Semnan station | $AD_3$, $AA_4$, $AD_4$, $AA_5$ |
| $T_6$ | After leaving Semnan station | $AA_4$, $AD_4$, $AA_5$ |
| $T_7$ | Before leaving Shahrud station | $AD_4$, $AA_5$ |
| $T_8$ | After leaving Shahrud station | $AA_5$ |

**Figure 2.2:** Table 5 of Vafaei and Yaghini, 2024. The 8 prediction times are $T_1$ through $T_8$ have corresponding train positions along the route, where $AA_i$ denotes the actual arrival time from station $i$ and $AD_1$ denotes the actual departure time from station $i$, where $i = 1, 2, ..., 5$ for each of the five stations. In order of route, the five stations are Tehran, Garmsar, Semnan, Shahrud, and Mashhad. There is a predictable cascading: As the train completes station stops along its prescribed route, the models have fewer target variables to predict of stations ahead.

At each of the 8 prediction time, each of the 4 models forecasts the departure time of the current station, and the arrival and departure times of every subsequent station along the route. Vafaei and Yaghini evaluate the models using $R^2$ scores and RMSE, selecting XGB as the best overall model with an averaged $R^2$ score of 0.9195 and an averaged RMSE of about 16 minutes and 51 seconds.

## 2.2 Rail Systems in Switzerland

Among European countries and indeed around the world, the time passengers must wait for trains depends on the standard of living and infrastructure technology.

Switzerland has railway lines 5,200 km long [16]. The average person in Switzerland travels 2,400 km by train annually, a higher rate than any other country. Going through the Swiss Alps, the Gotthard Base Tunnel is the longest railway tunnel at 57 km [17]. The tunnel is used for both freight and passenger transportation, with 325 trains passing through daily at velocities as great as 250 km/h [16]. In 2012, 2015, and 2017, Switzerland ranked highest in the European Railway Performance Index [18]. Swiss passengers are accustomed to waiting only short times for trains. For years 2015–2016, $94.1\%$ of Swiss trains were punctual within 3 minutes, as seen in Figure 2.3 from the European Passengers' Federation

report of train punctuality in Europe [19].

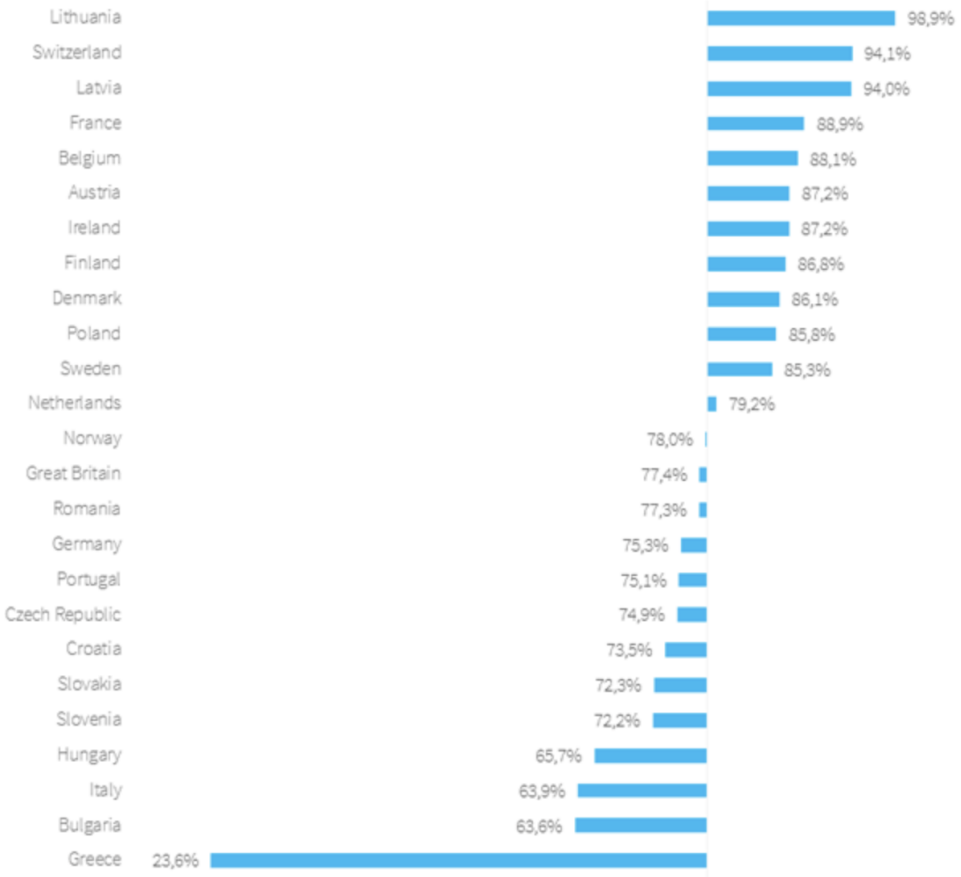



**Figure 2.3:** Image from the European Passengers' Federation report of train punctuality in Europe. The graphic shows information by country of average long-distance train punctuality for years 2015 and 2016. The percentages represent the fraction of trains that were punctual within 5 minutes of expected arrival time. Switzerland is the exception in this graphic, whose percentage is calculated on punctuality within 2 minutes and 59 seconds. It is noteworthy that Switzerland has a more rigorous cutoff, suggesting that the country's rail network is committed to providing, and its regular consumers are accustomed to using, a higher standard of service. Despite the narrower cutoff time, Switzerland ranks second out of all European countries listed, at $94.1\%$ of long, medium, and short distanced trains punctual within 3 minutes.

## 2.3 Learning Models

Regressor models were considered over Neural Networks (NN) and Large Language Models (LLM) because of computational constraints and suitability for well-suited nature for structured tabular data. NNs are black boxes, and model interpretation is important to this problem. NNs are not well-suited to mixed data types, which is what this dataset is comprised of. LLMs are intended for text rather than numbers and are computationally expensive.

Several models were contenders for application in this dissertation: linear, decision tree, Gradient

Boosting (GB), K-Nearest Neighbors (KNN) and Support Vector Regressors (SVR) from scikit-learn [20], and eXtreme Gradient Boosting (XGB) from the XGBoost library [21].

All of these are supervised models, learning from labelled data. These regressors are trained on continuous or categorical input features to predict a continuous target variable.

### 2.3.1 Linear Regressor

This model assumes that the relationship between features and target are linear. If this assumption is incorrect, the model may perform poorly, and another model may be a better fit for the data.

The linear regressor Equation 2.1 is defined below, where $x_1, x_2, ..., x_p$ are features; $y$ is the target; $\beta_0, \beta_1, ..., \beta_p$ are the coefficients, and $\varepsilon$ is the error. The model creates a best fit linear relationship between features and the target using Ordinary Least Squares [22], as seen in Equation 2.2, where the squared distances between predicted and actual data are minimized.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon \tag{2.1}$$

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}\right)^2 \tag{2.2}$$

Mathematically, the interpretation of the intercept is the baseline prediction delay when all feature variables are set to zero (even if it does not make sense with variables in context). Feature coefficients can be used to determine how great or little an impact that feature had on the model. A feature variable with a coefficient of 1 would mean that for every unit increase in that variable, the predicted delay increases by 1 second; and a coefficient of -1 would mean that would mean a 1 second decrease in predicted delay.

### 2.3.2 Decision Tree

The decision tree regressor [23] operates by splitting the feature space. Internal nodes test feature conditions, the outcomes of which become ”branches”, and the predicted values are given at the ”leaves”, or rectangular regions of the feature space, at the ends of these branches.

In Equation 2.3, the prediction $\hat{y}_m$ is the mean of the target values in that region or leaf, $R_m$.

$$\hat{y}_m = \frac{1}{|R_m|} \sum_{x_i \in R_m} y_i \tag{2.3}$$

At each node, the branch that minimizes the Residual Sum of Squares (RSS) in Equation 2.4 is chosen greedily. $M$ represents the number of leaves, and $y_i$ represents an actual target value.

$$\text{RSS} = \sum_{m=1}^{M} \sum_{x_i \in R_m} (y_i - \hat{y}_m)^2 \tag{2.4}$$

For non-linear relationships between features and target, a decision tree may be a good fit. It is also a good model for handling both quantitative and qualitative variables. It is, however, prone to overfitting, and is susceptible to variance even by small changes in data.

### 2.3.3 GB

GB is an ensemble method. Building models of decision trees sequentially, new models try to correct previous models' errors. This process begins with the initial prediction. Typically, the mean of the target variable is used as the initial prediction $F_0(x)$, defined in Equation 2.5, where $y_i$ is an actual value.

$$F_0(x) = \frac{1}{n} \sum_{i=1}^{n} y_i \tag{2.5}$$

During each iteration $m$, the algorithm computes residuals as in Equation 2.6, where $r_{im}$ is the difference between actual value and the model's predictions at iteration $m - 1$.

$$r_{im} = y_i - F_{m-1}(x_i) \tag{2.6}$$

A new decision tree $h_m(x)$ is created to fit these residuals, and the model adds the new tree's prediction scaled by learning rate $\nu$ with Equation 2.7.

$$F_m(x) = F_{m-1}(x) + \nu \cdot h_m(x) \tag{2.7}$$

The stopping criteria for this process can be a fixed number of iterations or when the improvements are negligible.

### 2.3.4 XGB

Similarly to GB, XGB builds an ensemble of decision trees. Unlike GB, however, XGB uses regularization to control model complexity.

Equation 2.8 defines how XGB makes predictions $\hat{y}_i$. Predictions are the sums of the outputs from the $K$ regression trees. Functions $f_k$ belong to $\mathcal{F}$, the space of all possible regression trees in the ensemble.

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F} \tag{2.8}$$

The XGB regressor employs regularization according to its loss function in Equation 2.9. The first summation represents empirical loss between predicted and actual values. The second summation is a regularization penalty, where the large numbers of leaves $T_k$ and large leaf weights $w_{jk}$ are discouraged. This helps prevent trees growing too complex, thus overfitting the data. The optimal leaf weight is given by $w_j^*$ in Equation 2.10, where $g_i$ is the gradient (first derivative) and $h_i$ is the Hessian (second derivative) of the loss with respect to the model's prediction. As in Equation 2.9, $\lambda$ is the regularization term.

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \left( \gamma T_k + \frac{1}{2} \lambda \sum_{j=1}^{T_k} w_{jk}^2 \right) \tag{2.9}$$

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{2.10}$$

### 2.3.5 KNN

KNN identifies a predetermined number of $k$ nearest data points in the training set to create a prediction, usually by averaging the target values of these points. For each new input $x$, the algorithm computes the distance to every training point $x_i$. Equation 2.11 shows one possible distance metric, Euclidean, where $p$ represents the number of features. Other distance metrics such as Manhattan may be used instead.

$$d(x, x_i) = \sqrt{\sum_{j=1}^{p} (x_j - x_{ij})^2} \tag{2.11}$$

The algorithm then selects the $k$ training points with the smallest distances to $x$. The predicted output is given by $\hat{y}(x)$ in Equation 2.12, where $\mathcal{N}_k(x)$ is the indices of $x$'s $k$-nearest neighbors and $y_i$ is the actual value of neighbor $i$.

$$\hat{y}(x) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(x)} y_i \tag{2.12}$$

The KNN model is non-parametric and uses lazy learning, only storing training data and making predictions when needed. The model may also be weighted, giving preference to the closest points among the $k$-nearest neighbors. Before using this model, the data need to be standardized. If features are not on the same scale, KNN will not measure distances accurately.

### 2.3.6 SVR

SVR can be useful for linear and non-linear relationships alike between features and target since it can use kernel function to model complex patterns. It tries to find a function $f(x)$ that predicts target values

within a margin of error $\varepsilon$ from target $y_i$. The linear case of $f(x)$ is given by Equation 2.13, where $\langle w, x \rangle$ represents the dot product between weight and input vectors, and $b$ represents the bias term.

$$f(x) = \langle w, x \rangle + b \tag{2.13}$$

In a nonlinear case, data are mapped to a higher-dimensional kernel function $K$ such as the Gaussian kernel given by Equation 2.14. The nonlinear case of prediction function $f(x)$ is then given by Equation 2.15, where $\alpha_i$ and $\alpha_i^*$ are Lagrange multipliers used to determine the degree of influence each training point $x_i$ has.

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right) \tag{2.14}$$

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) K(x_i, x) + b \tag{2.15}$$

SVR works to solve the optimization problem given by Equation 2.16, where $C$ is the regularization parameter, $\xi_i$ and $\xi_i^*$ are slack variables for points outside the accepted margin, and n is the number of training samples.

$$\begin{aligned} \min_{w,b,\xi_i,\xi_i^*} \quad & \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*) \\ \text{subject to} \quad & y_i - \langle w, x_i \rangle - b \leq \varepsilon + \xi_i \\ & \langle w, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0 \end{aligned} \tag{2.16}$$

Like KNN, SVR requires data standardization so it can calculate distances accurately.
In summary, this section has reviewed several supervised learning models suitable for time series forecasting. The Linear Regressor may serve well as a baseline for other models, but only captures linear relationships. The Decision Tree can model nonlinear patterns, but may be prone to overfitting. GB and XGB build ensembles of trees that may improve accuracy, and XGB adds regularization. KNN uses distance metrics to make predictions in feature space, and SVR uses kernel functions to capture relationships. These models offer a range of trade-offs including accuracy and computational cost.

### 2.3.7 Model Selection

Three of these models were selected for the use of this dissertation: Linear, Decision Tree, and XGB Regressors.

The linear regressor is fast and serves as a helpful baseline for the other models. The decision tree is also relatively fast and can handle more complex relationships in the data than the linear model. XGB is able to capture more complex relationships than the linear regressor, and is less prone to overfitting data than decision trees. XGB also optimizes training better than GB with regularization and parallel computation.

KNN and SVR were not selected due to their high computational and memory costs compared to the models chosen. These two models also require standardization, while the chosen model need no additional data preparation.

These three regressors were selected in combination because they each possess different strengths. The linear regressor is the simplest, and can quickly determine if the relationship between features and target is linear. The decision tree regressor is able to capture nonlinear patterns while still being relatively light computationally. XGB can be thought of as an extension of decision trees, since it is an ensemble method that uses gradient boosting. By running simple models alongside more complex ones, this combination gives a wider scope of options from which to select the best model.

# 3

# Design

## Contents

## 3.1 Background

The data were taken from the Open Data Platform Mobility Switzerland (ODMCH) [2]. The ODMCH website has publicly available CSV files of all transportation that takes place under their jurisdiction, uploaded daily. ODMCH is run by or works with the following organizations: System Tasks Customer Information Plus (SKI+), the Federal Office of Transport (FOT), Rail Infrastructure Fund (RIF), and the National Data Networking Infrastructure Mobility (NADIM) [24]. The way these organizations collect and process the data is illustrated in Figure 3.1, which can be found on the ODMCH's Big Picture site [25], sourced from SKI+'s Escalation process site [26].

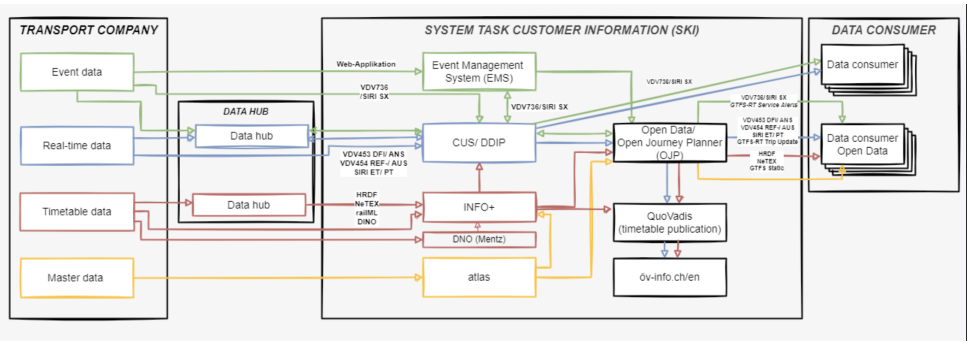



**Figure 3.1:** Image from ODMCH's Big Picture site [25], sourced from SKI+'s Escalation process site [26], detailing how the data used in this dissertation are collated, integrated, and published into publicly available forms.

### 3.1.1 Descriptive statistics

Only the winter months – January, February, and March – of the years 2021, 2022, 2023, and 2024 were used. This subset of months was chosen due to memory and time constraints. Since Switzerland is in a colder climate, trains during winter may be affected by adverse weather conditions or accidents, causing trains to run behind schedule.

The data span dates from 1 January 2021 until 31 March 2024. The 1,854 unique stop names range alphabetically from "Aadorf" to "Zürich Wollishofen" on 22,967 unique lines.

## 3.2 Data Cleaning

The CSV files for each completed day (January through March, 2021 through 2024) were downloaded from the open data source. All modes of transportation except train products were filtered out, and the 21 column names were translated from Swiss-German into English. The data were combined into parquet [29] files. Once read from parquet files into dataframes, the `OPERATION_DATE` column of type

| Original field name (Swiss-German) | Translated field name (English) |
|---|---|
| BETRIEBSTAG | OPERATION DAY |
| FAHRT_BEZEICHNER | TRIP_IDENTIFIER |
| BETREIBER_ID | OPERATOR_ID |
| BETREIBER_ABK | OPERATOR_ABK |
| BETREIBER_NAME | OPERATOR_NAME |
| PRODUKT_ID | PRODUCT_ID |
| LINIEN_ID | LINE_ID |
| LINIEN_TEXT | LINES_TEXT |
| UMLAUF_ID | CIRCULATION_ID |
| VERKEHRSMITTEL_TEXT | TRANSPORTATION_TEXT |
| ZUSATZFAHRT_TF | ADDITIONAL TRIP_TF |
| FAELLT_AUS_TF | FAILS_OFF_TF |
| BPUIC | BPUIC |
| HALTESTELLEN_NAME | STOP_NAME |
| ANKUNFTSZEIT | ARRIVAL TIME |
| AN_PROGNOSE | AN_FORECAST |
| AN_PROGNOSE_STATUS | AN_FORECAST_STATUS |
| ABFAHRTSZEIT | DEPARTURE TIME |
| AB_PROGNOSE_STATUS | AB_FORECAST_STATUS |
| DURCHFAHRT_TF | TRANSIT_TF |

**Table 3.1:** Column name translations from Swiss German to English. For more information about each column, see Figure 3.2.

string was duplicated and the copy was converted to datetime format in order enable grouping and time-based analysis by month and year. The data cleaning steps that followed were removing skipped stops (see Figure 3.5), removing cancelled stops (see Figure 3.6, and removing rows where ETA and/or ETD status was missing (see Figure 3.7. Arrival and departure columns were then duplicated, and the copies were converted from type string to type datetime. Two additional columns were added, indicating the previous stop and its corresponding departure time. This was done by grouping by trip ID and operation date, sorting by arrival time, and shifting the stop name and departure time. After the shift, rows with missing ETA and/or ETD times were removed (see Figure 3.8. A new Delta column of type timedelta was added by subtracting the previous departure time (from the shift function) from the arrival time. A separate column with the Delta values in seconds was also created. Delta values outside the range $120 < \text{Delta} \leq 3600$ were removed (see Table 3.6).

To prepare for modeling, temporal features were extracted from the departure time into month, day of the week, hour, and minute columns. The stop names and previous stop names were transformed from strings into integers using label encoding [30] for better understanding by the models.

| Month | Forecast | Real | Unknown | NaN |
|---|---|---|---|---|
| 2021-01 | 734,358 | 3,489,408 | 510,722 | 0 |
| 2021-02 | 668,870 | 3,192,182 | 474,731 | 0 |
| 2021-03 | 743,876 | 3,498,396 | 553,659 | 0 |
| 2022-01 | 335,594 | 3,631,636 | 449,533 | 413,536 |
| 2022-02 | 304,756 | 3,307,887 | 417,374 | 374,368 |
| 2022-03 | 336,513 | 3,681,231 | 515,588 | 419,662 |
| 2023-01 | 346,591 | 3,756,609 | 378,653 | 423,549 |
| 2023-02 | 294,793 | 3,336,901 | 347,422 | 372,581 |
| 2023-03 | 335,252 | 3,640,461 | 396,638 | 407,185 |
| 2024-01 | 193,731 | 4,049,758 | 273,866 | 431,311 |
| 2024-02 | 177,193 | 3,802,466 | 259,988 | 407,396 |
| 2024-03 | 160,059 | 3,996,171 | 320,323 | 425,087 |

**Table 3.2:** Monthly ETA status counts for data cleaned up to this step. This table contains numerical information that corresponds with the visual information in Figure 3.7(A).

| Month | Forecast | Real | Unknown | NaN |
|---|---|---|---|---|
| 2021-01 | 676,387 | 3,546,750 | 511,351 | 0 |
| 2021-02 | 615,382 | 3,244,831 | 475,570 | 0 |
| 2021-03 | 685,298 | 3,555,834 | 554,799 | 0 |
| 2022-01 | 273,447 | 3,693,337 | 450,664 | 412,851 |
| 2022-02 | 251,926 | 3,360,268 | 418,633 | 373,558 |
| 2022-03 | 275,347 | 3,742,600 | 516,530 | 418,517 |
| 2023-01 | 291,573 | 3,816,554 | 378,353 | 418,922 |
| 2023-02 | 246,432 | 3,388,060 | 346,850 | 370,355 |
| 2023-03 | 280,202 | 3,696,985 | 396,383 | 405,966 |
| 2024-01 | 291,573 | 3,816,554 | 378,353 | 418,922 |
| 2024-02 | 246,432 | 3,388,060 | 346,850 | 370,355 |
| 2024-03 | 280,202 | 3,696,985 | 396,383 | 405,966 |

**Table 3.3:** Monthly ETD status counts for data cleaned up to this step. This table contains numerical information that corresponds with the visual information in Figure 3.7(B).

| STOP_NAME | PREV_STOP | ACTUAL_ARR_dt | ACTUAL_DEP_dt | PREV_DEP_dt |
|---|---|---|---|---|
| Stein-Säckingen | NaN | 2024-01-01 13:41:04 | 2024-01-01 13:41:22 | NaT |
| Mumpf | Stein-Säckingen | 2024-01-01 13:44:21 | 2024-01-01 13:44:39 | 2024-01-01 13:41:22 |
| Möhlin | Mumpf | 2024-01-01 13:48:22 | 2024-01-01 13:49:10 | 2024-01-01 13:44:39 |
| Rheinfelden | Möhlin | 2024-01-01 13:51:30 | 2024-01-01 13:52:11 | 2024-01-01 13:49:10 |
| Rheinfelden Augarten | Rheinfelden | 2024-01-01 13:53:54 | 2024-01-01 13:54:14 | 2024-01-01 13:52:11 |

**Table 3.4:** To validate the shift, multiple random groups of 5 sequential rows were manually checked. Each group passed if the shifted values matched the actual values from the previous row. As expected, the first stop of each trip had `NaN` values in the `previous_stop` column. This table is an example checking if the shift function worked as expected. The data in the table are from a partially cleaned dataframe (up to this stage in the cleaning) based on the trip ID 85857, chosen randomly as all checks were. Here, the head of the data were printed, with both the previous stop and previous departure time (in datetime) being NaN/NaT, as the first stop of the route.

| Field name | type | Description | Example |
|---|---|---|---|
| OPERATION DAY | DD.MM.YYYY | This is the relevant operating day on which the journey took place. | August 20, 2016 |
| TRIP_IDENTIFIER | String | Corresponds to the journey IDLocal transport:[UIC country code]:[GO number]:[Journey reference]Rail transport: [UIC country code]:[GO number]:[VM number]:[Extended reference] | 85:81:20920:001 |
| OPERATOR_ID | String | GO number (Switzerland) or "TU code" (abroad). The country code is placed at the front. | 85:81 |
| OPERATOR_ABK | String | Derived from the OPERATOR_ID information in the master data of the transport company | THURBO |
| OPERATOR_NAME | String | Derived from the OPERATOR_ID information in the master data of the transport company | Aare Seeland mobile (snb) |
| PRODUCT_ID | Enum | Product types are used within the interface to classify quality attributes for journeys. In Swiss public transport, the transport type (e.g., "ship," "bus," "train," etc.) is transmitted as the Product ID. If the Product ID is specified, the respective data-producing transport company must ensure that the transmitted VM type matches the VM types used in the Swiss public transport target timetable collection (INFO+). A list of the VM types supported in INFO+ can be requested from the INFO+ specialist office. | Train |
| LINE_ID | String | The line ID is a purely technical key that is not used for customer display. Local transport: [UIC country code]: [GO number]: [Technical line key] Rail transport: [Journey number] (Journey number may only be obtained from the JOURNEY IDENTIFIER) | 85:65:20920 (NAV) or 20920 (rail traffic) |
| LINES_TEXT | String | The line text is customer-relevant and is displayed on the respective displays if necessary. | S29 |
| CIRCULATION_ID | String | If the transport company transmits circulations, the circulation ID is filled in here. | BUS.391121121 |
| TRANSPORTATION_TEXT | String | Textual description of the means of transport | S |
| ADDITIONAL TRIP_TF | Boolean | true/false (if null=false). Set to true if this is an additional trip. | false |
| FAILS_OFF_TF | Boolean | true/false (if null=false). Set to true if the trip is canceled. | false |
| BPUIC | Numerical | Stop (in BPUIC format). String. E.g. from DiDoK. The ID of the stop. Basic format: UIC country code (2 digits): e.g., 85 UIC stop code (5 digits): e.g., 03000 Stop code (optional): e.g., 02 results in: 850300002 | 8506038 (railway) or 850300002 (NAV) |
| STOP_NAME | String | The textual representation of the stop. It is used as supplied in the original data by the transport company and not derived from BPUIC via BP master data. This means that there may be deviations from the official name in the DiDok list. | Winterthur Wallrüti |
| ARRIVAL TIME | DD.MM.YYYY HH24:MI | The expected arrival time at the stop rounded to minutes. | 12.10.2016 05:40 |
| AN_FORECAST | DD.MM.YYYY HH24:WED:SS | The arrival forecast is without rounding. Note: It's possible that a transport company has already performed the rounding itself, and therefore the seconds are not available. | 12.10.2016 05:41:32 |
| AN_FORECAST_STATUS | Enum | Possible values: UNKNOWN (No forecast or actual times available for this and all previous stops) Empty (= FORECAST) FORECAST (arrival forecast) ESTIMATED (calculated actual arrival time) REAL (effective actual arrival time) | FORECAST |
| DEPARTURE TIME | DD.MM.YYYY HH24:MI | The scheduled departure time at the stop rounded to minutes. | 12.10.2016 05:40 |
| AB_FORECAST | DD.MM.YYYY HH24:WED:SS | The departure forecast without rounding. Note: It's possible that a transport company has already performed the rounding itself, and therefore the seconds are not available. | 12.10.2016 05:41:32 |
| AB_PROGNOSE_STATUS | Enum | Possible values: UNKNOWN (No forecast or actual times available for this and all previous stops) Empty (= FORECAST) FORECAST (departure forecast) ESTIMATED (calculated actual departure time) REAL (effective actual departure time) | UNKNOWN |
| TRANSIT_TF | Boolean | true/if (null=false). During a transit, the vehicle does not stop at this planned stop. | false |

**Figure 3.2:** Image from ODMCH's Actual Data Short Description site in English [27], translated from the Swiss-German site [28]. Each completed day's transportation data are uploaded in a CSV file with 21 fields (in Swiss-German in the CSV files). These data detail all public transportation, including by ship, bus, and train. This dissertation focuses only upon train data. The 21 fields include predicted arrival time, actual arrival time, predicted departure time, and actual departure time. This dissertation aims to better the predictions using machine learning outside the transportation organizations. For fieldname list in original Swiss-German, see Table 3.1.

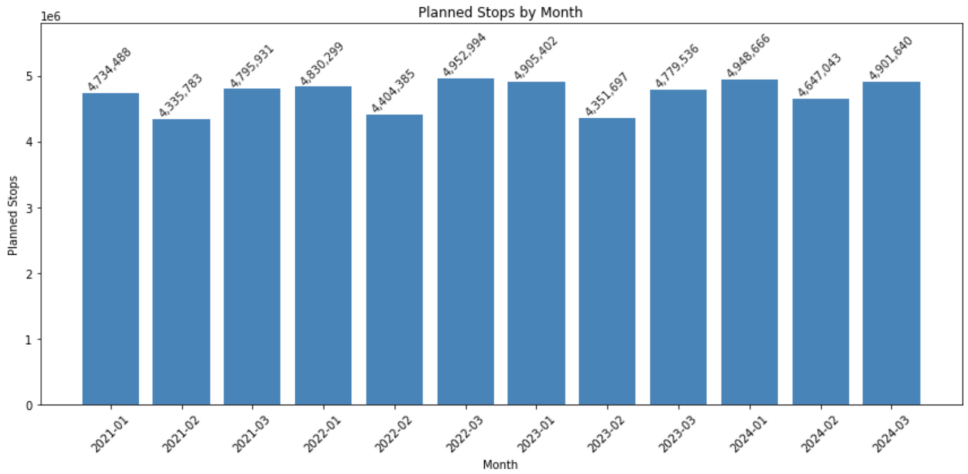



**Figure 3.3:** Monthly intended train stops before data cleaning. The dataset had a length of 56,587,864 rows. The number of unique stops was 1,854, and the number of unique lines was 22,967.

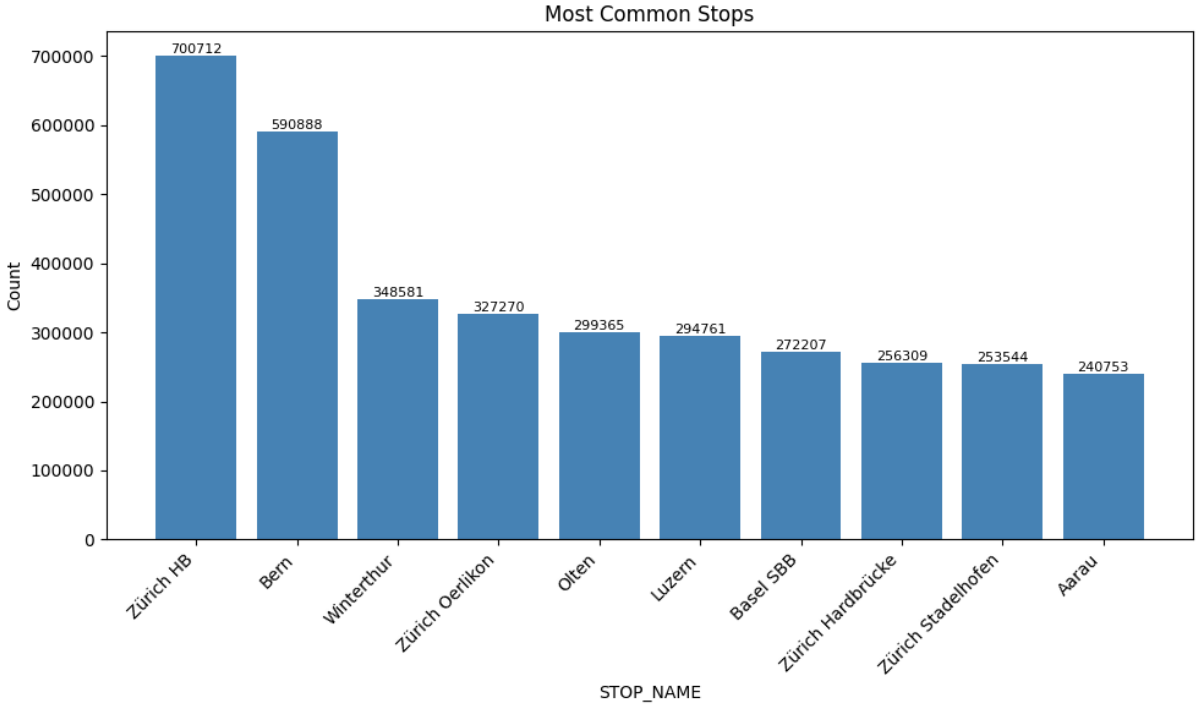
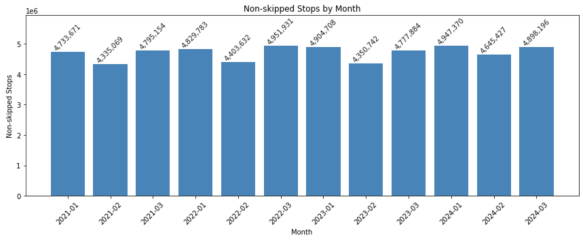
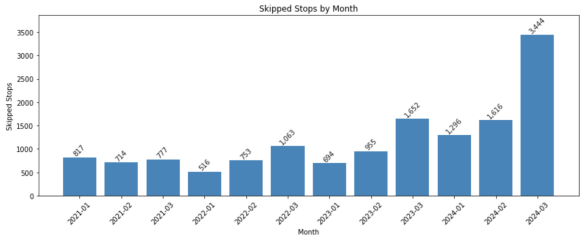



**Figure 3.4:** The 10 most frequented stops on the uncleaned data before data cleaning.

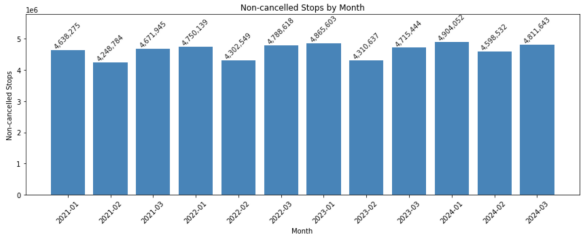



**(A)** Non-skipped stops

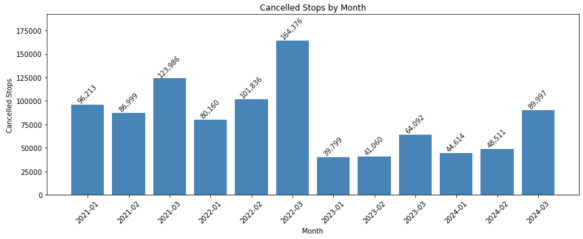



**(B)** Skipped stops

**Figure 3.5:** The Boolean column `NO_STOP` indicates whether the train skipped its planned stop, where `NO_STOP == False` means the train stopped as intended. There were no NaN values in these months of data. Skipped stops were removed from the cleaned data.

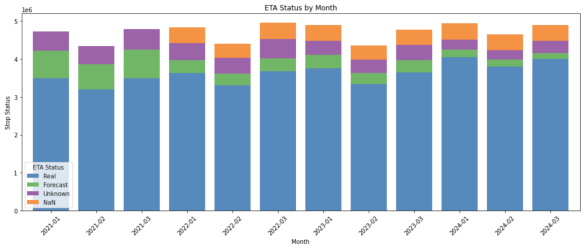



**(A)** Non-cancelled stops

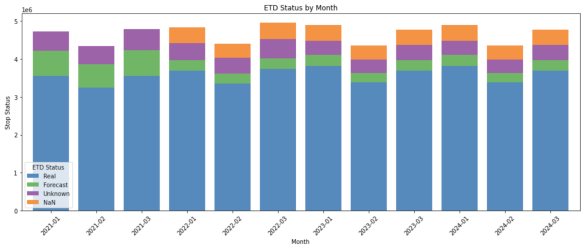



**(B)** Cancelled stops

**Figure 3.6:** The boolean column `CANCELLATION` indicates whether the stop was cancelled (true values indicate cancellations). If false, then the train completed its stop as planned. Only non-cancelled data were kept.




**(A)** ETA Status




**(B)** ETD Status

**Figure 3.7:** The columns `ACTUAL_ARR` and `ACTUAL_DEP` contain arrival and departure times for each stop. The nominal columns `ETA_STATUS` and `ETD_STATUS` indicate whether a given time is Real (also spelled "Real" in Swiss-German), Forecast ("Prognose" in Swiss-German), or Unknown ("Unbekannt" in Swiss-German). Only times marked as `Real` were treated as reliable.

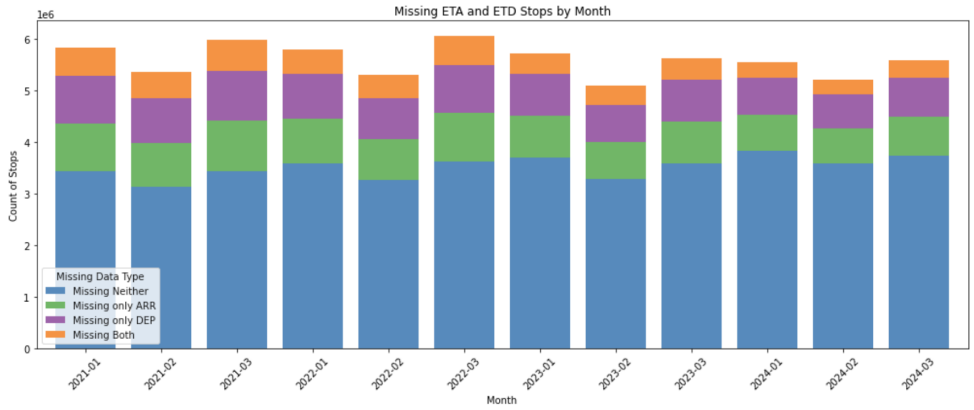



**Figure 3.8:** After shifting, some rows had an intact arrival time, departure time, and stop name, but some rows had one or more of these values missing (such as the first and last stops of each trip). This image depicts missing arrival and/or departures. Only rows with both arrival and departure times were kept.

| Month | Missing only ARR | Missing only DEP | Missing both | Missing neither |
|---|---|---|---|---|
| 2021-01 | 924,016 | 924,740 | 548,223 | 3,433,955 |
| 2021-02 | 854,629 | 855,583 | 509,313 | 3,134,884 |
| 2021-03 | 971,042 | 972,254 | 591,934 | 3,444,569 |
| 2022-01 | 866,547 | 867,852 | 485,443 | 3,581,343 |
| 2022-02 | 796,240 | 797,630 | 447,858 | 3,258,373 |
| 2022-03 | 941,244 | 942,323 | 548,891 | 3,618,318 |
| 2023-01 | 804,816 | 804,778 | 409,068 | 3,704,876 |
| 2023-02 | 722,973 | 722,674 | 374,771 | 3,280,821 |
| 2023-03 | 809,757 | 809,787 | 427,095 | 3,587,087 |
| 2024-01 | 707,457 | 708,004 | 297,584 | 3,830,789 |
| 2024-02 | 670,476 | 672,177 | 282,998 | 3,587,388 |
| 2024-03 | 751,241 | 751,931 | 344,954 | 3,743,422 |

**Table 3.5:** This table contains numerical information of missing arrival and/or departures that corresponds with the visual information in Figure 3.7.

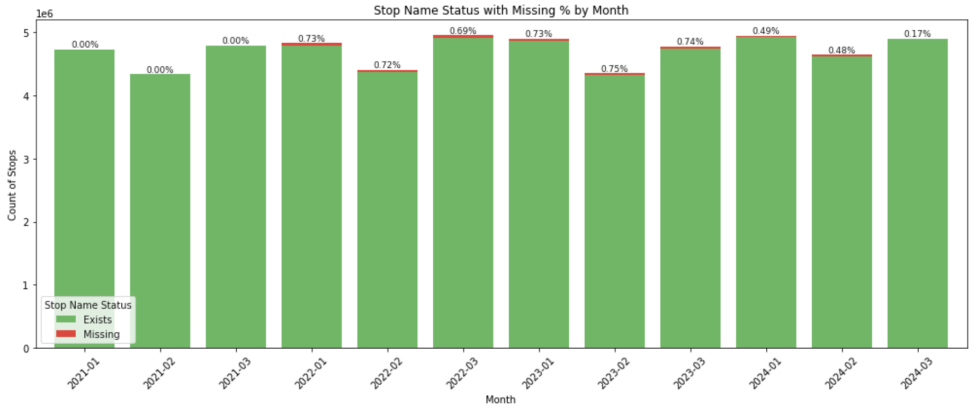



**Figure 3.9:** Existing and missing stop names by month. Rows with missing stop names and/or missing previous stop names were removed.

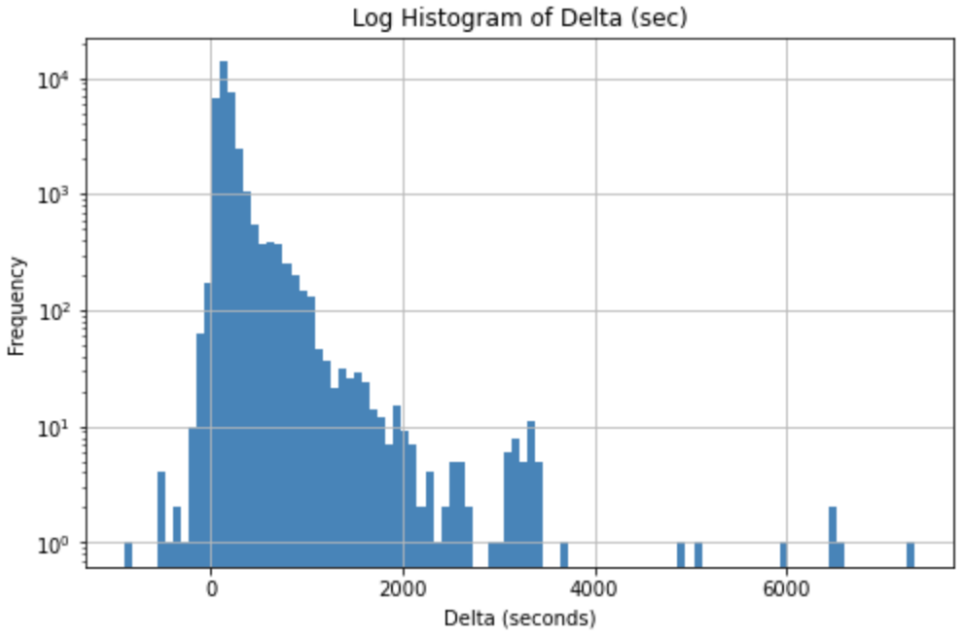



**Figure 3.10:** Histogram of Delta times before Delta cleaning step. The frequency is on a log scale for better understanding of data distribution. For counts by Delta threshold, see Table 3.6.

| Category | Count | Fraction |
|---|---|---|
| Delta $\leq 0$ | 174,251 | 0.005 |
| $0 <$ Delta $\leq 120$ | 11,306,650 | 0.328 |
| $120 <$ Delta $\leq 3600$ | 22,934,245 | 0.663 |
| Delta $> 3600$ | 6,892 | 0.004 |
| Delta is NaT | 0 | 0.000 |

**Table 3.6:** This table gives a breakdown of row count by Delta threshold (in seconds). Since the previous stop's departure time must necessarily come before the current stop's arrival time, Delta values $\leq 0$ do not make sense. Delta values $\leq 120$ seconds do not make sense either, as no stops in this rail system are within 2 minutes of train travel [31]. These errors may be due to faulty sensors or an error with the shift function. Similarly, no consecutive stops in this rail system should take more than 1 hour (3600 seconds). Such instances may be due to weather delays or emergencies on the train track. This small fraction of cases is negligible. Only values where $120 <$ Delta $\leq 3600$ were kept. At over 22 million rows and over 66% of the cleaned data up until this cleaning step, the size is sufficiently large.

### 3.2.1 Descriptive statistics on cleaned data

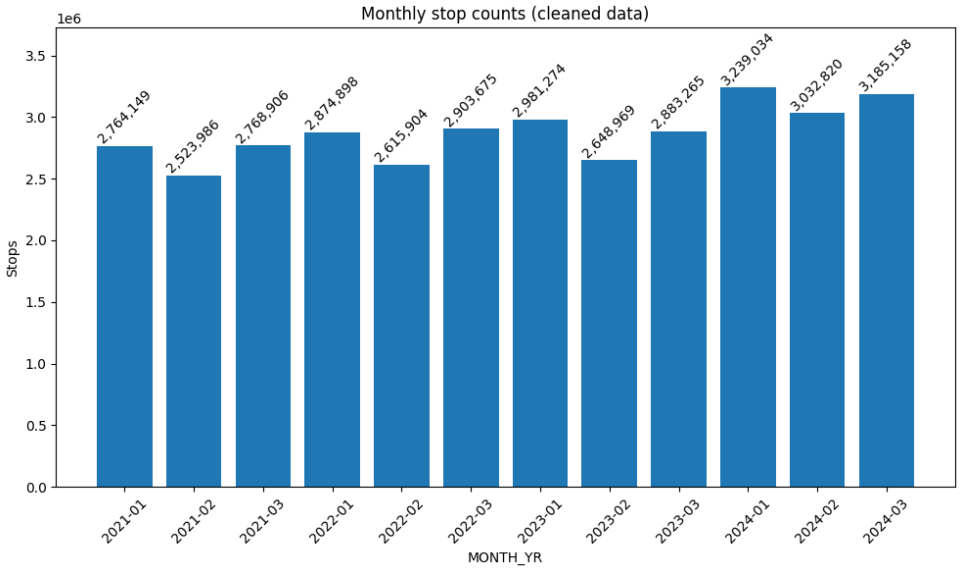



**Figure 3.11:** Monthly train stops from cleaned data. The cleaned data length 34,422,038, which is about 60.83% of the length of the uncleaned data. The number of unique stops in the cleaned dataset was 1,480, and the number of unique lines was 19,488.

## 3.3 Architecture

Once cleaned and prepared, the data were ready for the models. The feature variables were Month, Day of Week, Hour, Minute, Stop Name Encoded, and Previous Stop Name Encoded. The target variable was Delta_seconds. Only January, February, and March were analyzed, so Month could only take on values 1, 2, and 3. Hour and Minute made up the time of departure. Day of Week could take integers 0 through 6, representing Monday through Sunday, respectively [32].

The regression models used were Linear, Decision Tree, and XGB. A train-test split of size 80% training and 20% testing, with the random seed set at 14 for reproducibility.

Cross-validation [33] in the models was also used. Cross-validation is a resampling technique in which data are split into folds or parts. The model trains on all folds but one, which it tests on. The process repeats as many times as there are folds, so every fold is trained on once. The models in this dissertation used 5-fold cross-validation (with seed set at 14 for reproducibility), as it is not too computationally intensive, yet robust enough for reliability.

### 3.3.1 Evaluation Metrics

All models were evaluated using $R^2$ and Root Mean Squared Error (RMSE) metrics, as were percentage breakdowns of predictions within 1, 3, 5, and more than 5 minutes.

The formula for the $R^2$ score is given by equation 3.1, where $SS_{\text{res}}$ is the residual sum of squares defined by Equation 3.2, and $SS_{\text{tot}}$ is the total sum of squares, defined by Equation 3.3.

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \tag{3.1}$$

In Equations 3.2 and 3.3, $y_i$, $\hat{y}_i$, and $\bar{y}$ represent an actual value, a predicted value, and the mean of actual values, respectively.

$$SS_{\text{res}} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{3.2}$$

$$SS_{\text{tot}} = \sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{3.3}$$

An $R^2$ score of 1 is ideal, meaning the model is a perfect fit and explains 100% of the variance in the target variable. An $R^2$ score of 0 explains none of that variance, and means the model performs no better than the mean of the target variable, $\bar{y}$, for every point. Typically, models with $R^2$ scores between 0.7 and 1.0 are considered to have strong correlation, while models with $R^2$ scores between 0.01 and 0.39 are considered to have weak correlation. If $SS_{\text{res}} > SS_{\text{tot}}$, then $R^2 < 0$, meaning the model performs worse than predicting the mean.

RMSE, defined in Equation 3.4, measures the average magnitude of the error between actual values $y_i$ and predicted values $\hat{y}_i$. The number of observations is represented by $n$.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.4}$$

Since errors are squared, larger errors are penalized more than smaller ones. RMSE values that are close to 0 indicate that the model is very good fit, while large RMSE values indicate that the model's predictions are less accurate.

$R^2$ and RMSE metrics are useful metrics for model evaluation, but are not as meaningful to the typical train user as accuracy by minute. For a comprehensible real-world interpretation, each model is also evaluated by percentages of correct predictions within 1, 3, 5, and more than 5 minutes. Ideally, 100% of trains could be accurately predicted within 1 minute of actual travel duration.

The data come with built-in actual arrival, actual departure, target arrival, and target departure times (see Figure 3.2). The proposed model will be measured against averages and medians of the differences

between actual and target arrival times, and actual and target departure times.

| Metric | Value (hh:mm:ss) |
|---|---|
| Arrival (Actual − Target) average | -00:00:47 |
| Arrival (Actual − Target) median | 00:00:39 |
| Departure (Actual − Target) average | 00:10:47 |
| Departure (Actual − Target) median | 00:01:06 |

**Table 3.7:** Summary statistics of Arrival and Departure Timing Differences. On average, trains arrive 47 seconds before the time targeted by the railway, with a median of being 39 seconds late. Trains depart an average of 10 minutes and 47 seconds after the time targeted by the railway, with a median of being 1 minute and 6 seconds late.

### 3.3.2 Hyperparameters

The Decision Tree and XGB models have hyperparameter spaces which can be fine-tuned for optimal model fit (see Tables 3.8 and 3.9).

| Hyperparameter | Values |
|---|---|
| max_depth | 3, 5, 10, 15, None |
| min_samples_split | 2, 5, 10 |
| min_samples_leaf | 1, 2, 5, 10 |
| max_features | None, sqrt, log2 |

**Table 3.8:** The hyperparameter search space chosen for the Decision Tree Regressor. max_depth is the maximum depth of splits the tree can make from the root to a leaf. This range spans from shallow trees which may underfit to deeper, more complex trees which may overfit. min_samples_split is the minimum number of samples required to split an internal node. The values range from 2, allowing maximum growth, to 10, which helps prevent overfitting by requiring more samples before splitting. min_samples_leaf is the minimum number of samples required to be at a leaf node. The values range from 1, allowing for high variance, to 10, allowing for smoother predictions by requiring leaves to represent more data. max_features is the number of features to consider when looking for splits. None means all features are considered, sqrt can help trees with generalization, and log2 introduces randomness which helps reduce overfitting.

### 3.3.3 Randomized Search

The randomized search was chosen because of the immense size of the dataset coupled with limited computing resources. A grid search, on the other hand, exhausts every possible combination within the defined hyperparameter space, but this leads to exponentially increasing runtimes which is computationally expensive. The randomized search instead selects 20 combinations of parameters with which to fit a model. Prioritizing speed and memory, this search approach was optimal.

| Hyperparameter | Values |
|---|---|
| n_estimators | 100, 200, 300 |
| max_depth | 3, 5, 7, 10 |
| learning_rate | 0.01, 0.05, 0.1, 0.2 |
| subsample | 0.6, 0.8, 1.0 |
| colsample_bytree | 0.6, 0.8, 1.0 |

**Table 3.9:** The hyperparameter search space chosen for the XGB Regressor. n_estimators is the number of boosting rounds, meaning the total number of trees added to the model during training. The values range from more modest to more complex ensembles. max_depth is the maximum depth of each tree. The values range from shallower trees which may generalize better, to deeper trees that may capture more feature interactions. The learning_rates range from 0.01, slower and more stable learning, to 0.2, more aggressive learning. subsample is the fraction of training data randomly selected to train each tree in the boosting process. These values range from 0.6 to 1.0, meaning each tree is trained on 60% to 100% of the training data. A lower value introduces randomness and acts as a regularizer, and a higher value uses the full dataset. colsample_bytree is the fraction of features randomly selected and considered for splitting when building each tree. Similarly to the subsample parameter, lower values help reduce overfitting by introducing randomness, and the highest value 1.0 includes all features.

# 4

# Evaluation

## Contents

## 4.1 Measured Outcomes

In this section, models' outcomes are displayed in terms of $R^2$ and RMSE. Feature coefficients are included for the Linear model, while the best performing hyperparameters are discussed for the Decision Tree and XGB models.

For each model, a plot of the actual delay in seconds vs. the predicted delay in seconds is shown, demarcated in color and point shape by thresholds of 1, 3, 5, and more than 5 minutes. Since 5-fold cross-validation was used and only predictions from one fold were stored for later plotting use, the number of points available for plotting was 6,884,408, about one-fifth the size of the cleaned data. Test values under 2 minutes were omitted, as quantities that small do not make logistical sense for train travel between these stops. Furthermore, for ease of viewing and understanding the plots, $1\%$ of the predicted vs. actual delay data are displayed (45,873 coordinate pairs). It can be assumed that the randomly selected $1\%$ is representative of the total.

### 4.1.1 Linear Regressor Model

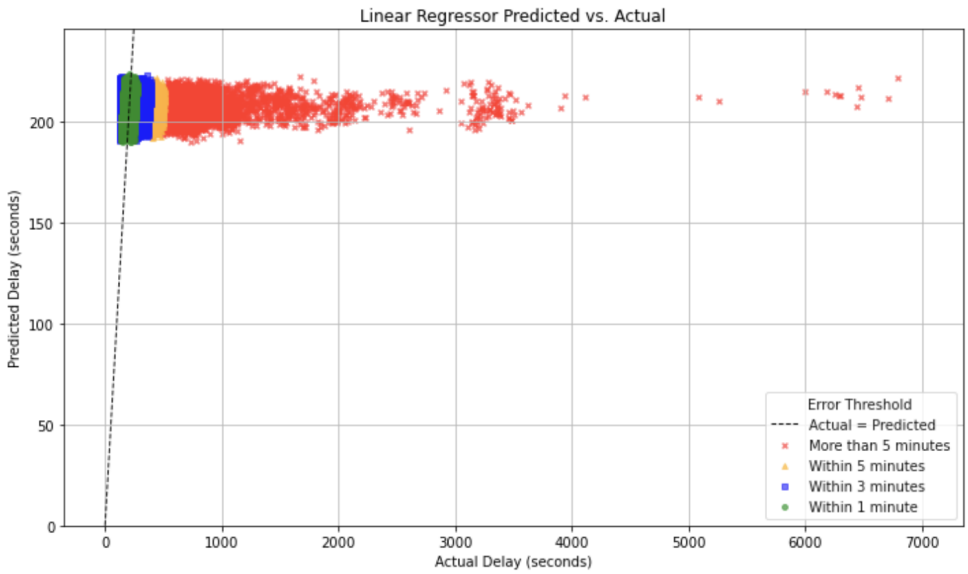



**Figure 4.1:** Linear Regressor Predicted vs. Actual Delay. For a wide domain of actual delays, the model predicts a relatively small range of predicted delays. The identity line (seen as dashed) represents the ideal path for points to fall upon, where the predicted and actual delay times are equal. It appears that much of the data fall around this line, but a significant fraction is also skewed to the right and under the line as well. For actual delays of more than 10 minutes and higher, the model predicts delays of less than 5 minutes. It appears that the model makes more accurate predictions for small values of actual delays, but for larger values it makes predictions with high errors. The errors do not seem to follow the trend of the identity line.

| Metric | Value |
|---|---|
| Cross-validated $R^2$ Scores | [0.00036942, 0.00046652, 0.00048163, 0.00050388, 0.00050007] |
| Average Cross-validated $R^2$ | 0.0005 |
| Test RMSE (seconds) | 245.95 |

**Table 4.1:** Linear Regressor model performance gives a medium-high RMSE of about 245 seconds (4 minutes and 5 seconds). The 5 cross-validated $R^2$ scores appear due to the 5 folds the model uses for training and testing. The averaged $R^2$ value of 0.00005 is very close to 0, indicating the model explains almost none of the variance in the data better than the mean. Based on the $R^2$ and RMSE metrics for evaluation, it appears that the linear model does a poor job of predicting delay times, or that the data are not linear to begin with.

| Feature | Coefficient |
|---|---|
| Intercept | 206.658 |
| Month | -0.325 |
| Day of Week | 0.680 |
| Hour | 0.621 |
| Minute | -0.041 |
| Stop Name Encoded | -0.003 |
| Previous Stop Name Encoded | -0.007 |

**Table 4.2:** Linear Regression Coefficients. The intercept coefficient gave a baseline prediction of a delay of about 207 seconds (3 minutes and 27 seconds). All feature coefficients had absolute values of less than 1, suggesting that none of the features had a strong influence on predicted delay.

### 4.1.2 Decision Tree Model

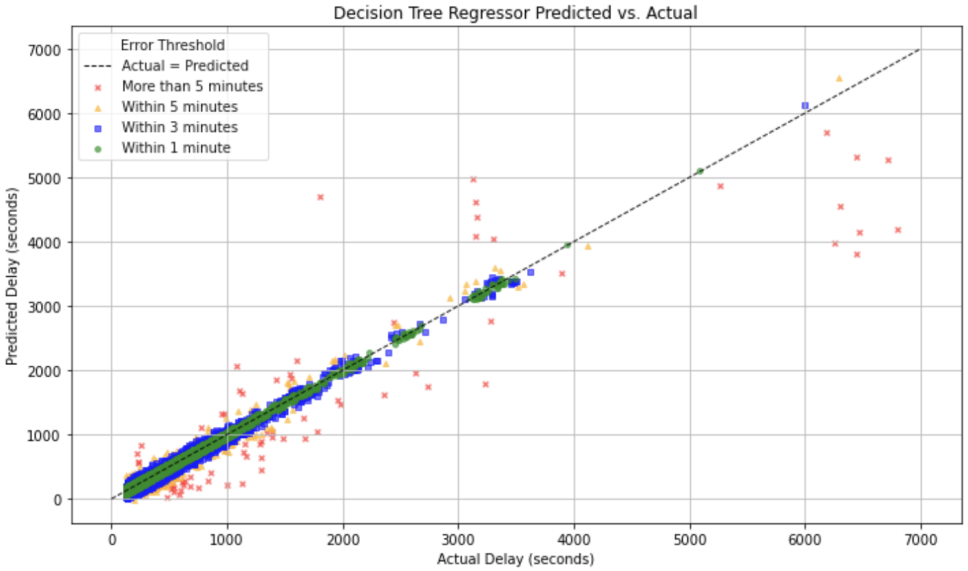



**Figure 4.2:** Decision Tree Predicted vs. Actual Delay. Most of the errors adhere closely to the identity line, indicating the model captures the general trend well. There are also some scattered values of more than 5 minute errors appearing at higher actual delays. This suggests that the model may be overfitting, as it is not predicting the error for long-delay journeys well. For the actual delays with high values, the predicted delays fall below the identity line, meaning that the model predicts these longer journeys will be shorter. It is also possible that the model is predicting accurately for normal case scenarios, and the discrepancies between long actual delays and shorter predicted delays are due to a train malfunction not captured within the scope of the data.

| Metric | Value |
|---|---|
| Test $R^2$ | 0.9599 |
| Test RMSE (seconds) | 49.27 |

**Table 4.3:** Decision Tree model metrics indicate strong model performance. The $R^2$ value means that about $96\%$ of the variance in delay times is explained by the model. The RMSE means that there is an average of about 49 seconds between predicted and actual values. Given that some of the actual values are upwards of 10 minutes, an RMSE of less than 1 minute is a low average error.

| Parameter | Value |
|---|---|
| min_samples_split | 10 |
| min_samples_leaf | 10 |
| max_features | None |
| max_depth | None |

**Table 4.4:** Best Parameters for Decision Tree of those defined in the model's hyperparameter space using a randomized search. The tree requires a minimum of 10 samples to split. A lower minimum than 10 may lead to overfitting, since splitting small subsets might capture noise rather than real trends in the data. The tree also requires at least 10 samples per leaf, limiting unnecessary complexity. With no maximum features, the tree considers all features before splitting. The tree can grow unbounded with no maximum depth. The strong regularization from min_samples_split and min_samples_leaf prevents the tree from growing excessively deep.

### 4.1.3 XGB Model

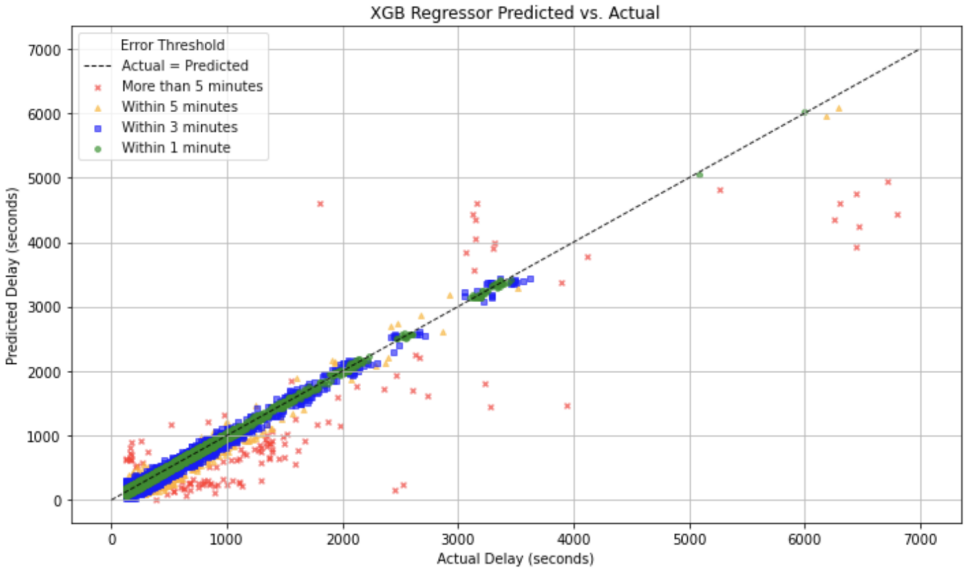



**Figure 4.3:** XGB Predicted vs. Actual Delay. The trend of the errors follows the identity line, meaning that the model is a good fit for the data. For many actual delays, both of large and small values, the model predicts shorter times, as the errors of more than 5 minutes fall below and to the right of the identity line. A possible explanation for more under- than over-predicting could be because the selected learning rate was 0.2. A relatively high learning rate like this can cause the model to make large updates early in the learning process, and biasing toward lower values later in the training process even for longer actual delays.

| Metric | Value |
|---|---|
| Test $R^2$ | 0.9502 |
| Test RMSE (seconds) | 54.90 |

**Table 4.5:** XGB model metrics indicate strong model performance. The $R^2$ value means that about $95\%$ of the variance in delay times is explained by the model. The RMSE means that there is an average of about 55 seconds between predicted and actual values. Given that some of the actual values are upwards of 10 minutes, an RMSE of less than 1 minute is a low average error.

| Parameter | Value |
|---|---|
| subsample | 1.0 |
| n_estimators | 300 |
| max_depth | 10 |
| learning_rate | 0.2 |
| colsample_bytree | 1.0 |

**Table 4.6:** Best Parameters for XGB of those defined in the model's hyperparameter space using a randomized search. The subsample was 1.0, meaning the model used the entire dataset. The number of boosting rounds, n_estimators, was high, indicating that many rounds of new trees trained to correct the errors of existing trees were necessary due to the complexity of the data. The maximum depth was 10, suggesting that deeper trees were necessary as well. The learning rate selected was the highest of those available, meaning the model improved by more quickly adjusting its predictions in each step. Finally, the fraction of features the model uses for each tree (colsample_bytree) was 1.0, so the model benefitted when all available features were used.

## 4.2 Accuracy

| **Category** | **Linear Regression** | | **Decision Tree** | | **XGB** | |
|---|---|---|---|---|---|---|
| | **Count** | **Cumul. %** | **Count** | **Cumul. %** | **Count** | **Cumul. %** |
| Within 1 minute | 2,478,019 | 35.995% | 6,639,462 | 96.442% | 6,575,190 | 95.508% |
| Within 3 minutes | 3,637,094 | 88.826% | 224,129 | 99.698% | 277,704 | 99.542% |
| Within 5 minutes | 322,772 | 93.514% | 13,740 | 99.898% | 18,129 | 99.805% |
| More than 5 minutes | 446,523 | 100.000% | 7,077 | 100.000% | 13,385 | 100.000% |

**Table 4.7:** Summary by time threshold and model for accuracies of predicted delay in seconds compared to actual delay in seconds. Counts and cumulative percentage are included. A visualization of the cumulative percentages compared by model can be found in Figure 4.4 in the subsequent subsection.

## 4.3 Comparisons

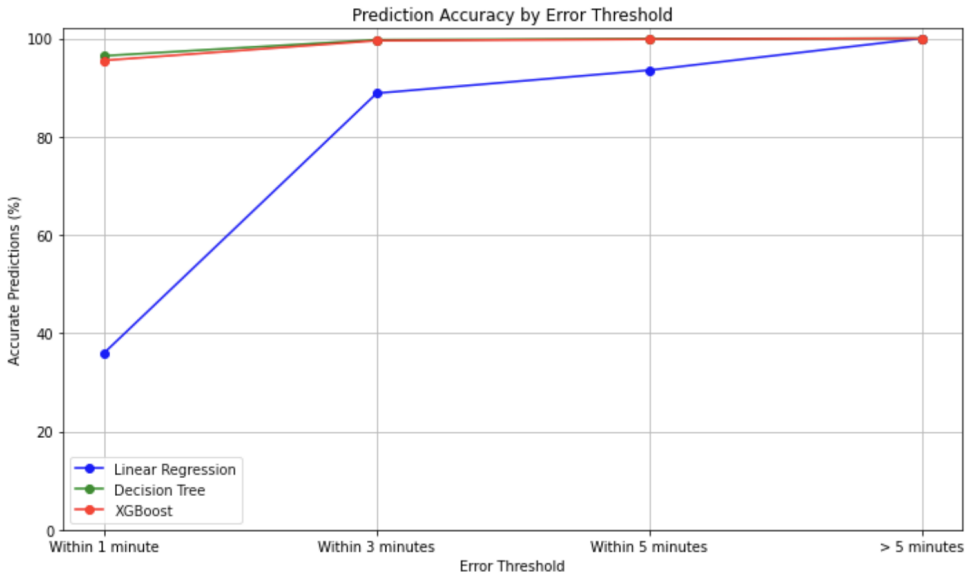



**Figure 4.4:** Model Accuracies by Cumulative Percentage. As seen in Table 4.7, both the Decision Tree and XGB models gave high accuracy. The Decision Tree accurately predicts delays within 1 minute about $96.4\%$ of the time, and within 5 minutes about $99.9\%$ of the time. The XGB model, similarly, accurately predicts delays within 1 minute about $95.5\%$ of the time, and within 5 minutes about $99.8\%$ of the time. The Linear Regressor, on the other hand, performed more poorly than the other two models. The linear model accurately predicts delays within 1 minute only about $36.0\%$ of the time, and within 5 minutes about $93.5\%$ of the time.

Figure 2.3 gives the percentage of trains that arrive within 3 minutes of expected time at $94.1\%$. While based on different years (2015-2016 and 2021-2024), the Decision Tree model puts that percentage at about $99.7\%$.

The best performing model proposed by Vafaei and Yaghini gave an averaged $R^2$ score of 0.9195 and an averaged RMSE score of about 16 minutes and 51 seconds, as stated in Chapter 2. Comparatively, the Decision Tree model here has a higher $R^2$ score by 0.0404. A difference of 0.04 is not substantial, and both $R^2$ scores of above 0.90 indicate strong model performance. Vafaei and Yaghini's train line in Iran and the Swiss train system may not be socially comparable, due to different infrastructure and cultural wait time expectations. It is, however, worth noting that Vafaei and Yaghini's model for a train line with only 5 stops yielded a significantly higher RMSE than the Decision Tree model here, which contains an entire network of train lines.

| Metric | Value (seconds) |
|---|---|
| Arrival (Actual − Target) average | 47 |
| Arrival (Actual − Target) median | 39 |
| Departure (Actual − Target) average | 647 |
| Departure (Actual − Target) median | 66 |
| Linear Regressor RMSE | 245.95 |
| Decision Tree RMSE | 49.27 |
| XGB RMSE | 54.90 |

**Table 4.8:** Comparison of uncleaned arrival and departure averages and medians (see Table 3.7 and model RMSE values. The lowest RMSE value of about 49 seconds, from the Decision Tree, is closest to the arrival average difference of 47 seconds, and is significantly lower than the departure average difference of 647 seconds (over 10 minutes).

## 4.4 Proposed Algorithm

For the reasons below, the Decision Tree Model is the proposed algorithm, should the Swiss rail authority choose to implement arrival time predictions using machine learning, or to incorporate the model into finding the likelihood of unexpected delays between target and actual times.

Of all three models, the Decision Tree ranks highest in percentages accurately predicted for all thresholds measured. It also has the best $R^2$ score and the lowest RMSE, performing within 3 seconds of the average difference between actual and target arrival times, and reduced error by a factor of 13 compared to the average difference between actual and target departure times.

While the differences in model performance between the Decision Tree and XGB regressor are not large, the Decision Tree model is significantly faster to run. The XGB regressor took not only more time, but also more computational power and memory in its execution. For a dataset as large as this, keeping in mind it only included the winter months, not the whole year, the XGB model is costly in terms of time and resources.

It is important to discuss the proposed algorithm's potential drawbacks as well. With the best performing Decision Tree having parameters of no maximum number of features and no maximum depth, the tree can grow very complex and deep. Since it can fit the training data nearly perfectly, it may also inadvertently fit noise. In the case of trains, noise can mean a broken sensor or an emergency causing a train line to stop for the day, giving unreliable data that should not be used for training. Even so, the benefits of accuracy performance in $R^2$ score, RMSE, and error thresholds, as well as relatively low computational cost, outweigh the risks of overfitting.

# 5

# Conclusions

## Contents

Train timetable accuracy is important to passengers for reliability, infrastructure for efficiency, and the rail company for consumer satisfaction. With the modern age of machine learning still in its early stages, applying these bourgeoning technologies to train systems has only recently begun. The objective of this dissertation is to provide one such model to predict travel duration times within the Swiss rail network using their publicly available data to train and test models, and to compare their efficacies.

This dissertation finds the Decision Tree most effective, compared to the Linear and XGB Regressors. Specifically, the Decision Tree with the parameters defined in Table 4.4. The criteria for best model were $R^2$ score, RMSE value, and accurate percentages at 1, 3, and 5 minute thresholds. The proposed tree considers all features and may grow unbounded, yet limits excessive complexity and depth.

These results demonstrate that a tree-based model, relatively simple compared to other possible approaches (see Section 2.3), can still achieve strong predictive performance. This emphasizes the potential of data-driven modelling to improve operational insight and service reliability for modern rail systems.

## 5.1 Review

A major part of this dissertation was preparing the data for machine learning, which required extensive cleaning. With 21 attributes and such a large dataset, every cleaning step had to be carefully considered. Due to limited computing power, files had to be handled creatively. To avoid long runtimes due to file size, larger files were sometimes broken up and recombined after cleaning, with care taken to avoid data leakage or partial duplication. Files were often first sampled to make sure cleaning steps worked as expected. Edge cases were taken into account, such as a first or last station on a line not having a valid arrival or departure time. The shift function to add a previous station needed careful logic, and several examples were checked for proper functionality.

It was interesting to apply theoretical knowledge of machine learning to a dataset by now so familiar. I learned how to pick a combination of models that together would be able to capture and explain different patterns in the data. Metrics for evaluation were selected before running models, so I had to consider what the metrics chosen should focus on.

In researching articles and other sources, I learned more about the field of time series forecasting specifically in transportation: models in the existing literature and different wait time standards in European countries. I also gained experience doing research in general by reading published journal articles.

## 5.2 Limitations

Throughout the dissertation writing process, I encountered several limitations. If not constrained by time and computational resources, I would have liked to delve deeper.

This dissertation makes use of data just from winter months, not the entire year. While I was able to use data from this period over multiple years, working with a whole year's worth of data over multiple years could have provided insight into seasonality trends. While outliers of errors greater than 5 minutes were discussed, a greater focus could be put on them in the future, investigating whether there were any clear patterns that might help mitigate large errors. Future research with more time and memory availability could try using a larger hyperparameter space, a grid search rather than a randomized search, and other models such as KNN and SVR.

There is still much to be discovered within the field of modern machine learning. Future work, supported by greater computing power, may build on this approach to address similar challenges more effectively.

## 5.3 Extrapolations

This dissertation addresses the prediction times only of passenger trains in Switzerland, but the findings have the potential to be effective in other scenarios as well. The Decision Tree Regressor model proposed here, with its best performing parameters would likely perform well in other countries with similar times between stops, especially for short-distance passenger data. It is not yet known how the proposed model would perform for other types of transportation within the same original dataset (such as by ship or bus), or whether the model could be applied to other non-transportation time series data that also have predicted times. These are questions that may be explored in future research.

# Bibliography


[1] Net Zero Nation, “Benefits of public transport,” 2025. [Online]. Available: https://netzeronation.scot/take-action/travel-less-car/benefits-public-transport

[2] Open Data Platform Mobility Switzerland, “Actual data,” https://data.opentransportdata.swiss/en/dataset/istdaten, 2025.

[3] ——, “Cookbook brief description,” https://opentransportdata.swiss/en/cookbook/historic-and-statistics-cookbook/actual-data, 2025.

[4] A. Agrawal, V. Kumar, A. Pandey, and I. Khan, “An application of time series analysis for weather forecasting,” *International Journal of Engineering Research and Applications (IJERA)*, vol. 2, no. 2, pp. 974–980, Mar-Apr 2012. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=4a09fbc9f67613feca328370c95f75622d26f11d

[5] A. Zeroual, F. Harrou, A. Dairi, and Y. Sun, “Deep learning methods for forecasting covid-19 time-series data: A comparative study,” *Chaos, Solitons & Fractals*, vol. 140, p. 110121, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S096007792030518X

[6] H. Iftikhar, F. Khan, P. C. Rodrigues, A. A. Alharbi, and J. Allohibi, “Forecasting of inflation based on univariate and multivariate time series models: An empirical application,” *Mathematics*, vol. 13, no. 7, p. 1121, 2025. [Online]. Available: https://doi.org/10.3390/math13071121

[7] Y. H. Cheng and Y. C. Tsai, “Train delay and perceived-wait time: passengers’ perspective,” *Transport Reviews*, vol. 34, no. 6, pp. 710–729, 2014. [Online]. Available: https://doi.org/10.1080/01441647.2014.975169

[8] S. Vafaei and M. Yaghini, “Online prediction of arrival and departure times in each station for passenger trains using machine learning methods,” *Transportation Engineering*, vol. 16, p. 100250, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666691X24000253

[9] Z. Li, C. Wen, R. Hu, C. Xu, P. Huang, and X. Jiang, “Near-term train delay prediction in the dutch railways network,” *International Journal of Rail Transportation*, vol. 9, no. 6, pp. 520–539, 2021.

[10] R. Shi, X. Xu, J. Li, and Y. Li, “Prediction and analysis of train arrival delay based on xgboost and bayesian optimization,” *Applied Soft Computing*, vol. 109, p. 107538, 2021.

[11] M. Yaghini, M. M. Khoshraftar, and M. Seyedabadi, “Railway passenger train delay prediction via neural network model,” *Journal of Advanced Transportation*, vol. 47, no. 3, pp. 355–368, 2013.

[12] P. Murali, M. Dessouky, F. Ordóñez, and K. Palmer, “A delay estimation technique for single and double-track railroads,” *Transportation Research Part E: Logistics and Transportation Review*, vol. 46, no. 4, pp. 483–495, 2010.

[13] B. J. Buijse, V. Reshadat, and O. W. Enzing, “A deep learning-based approach for train arrival time prediction,” in *Intelligent Data Engineering and Automated Learning–IDEAL 2021: 22nd International Conference, IDEAL 2021*. Manchester, UK: Springer International Publishing, 2021, pp. 213–222, november 25–27, 2021, Proceedings 22.

[14] S. Kosolsombat and W. Limprasert, “Arrival time prediction and train tracking analysis,” in *Trends in Artificial Intelligence: PRICAI 2016 Workshops: PeHealth 2016, I3A 2016, AIED 2016, AI4T 2016, IWEC 2016, and RSAI 2016*. Phuket, Thailand: Springer International Publishing, 2017, pp. 170–177, august 22–23, 2016, Revised Selected Papers 14.

[15] Y. Chen and L. R. Rilett, “Train data collection and arrival time prediction system for highway–rail grade crossings,” *Transportation Research Record*, vol. 2608, no. 1, pp. 36–45, 2017.

[16] Presence Switzerland, “Transport,” https://www.aboutswitzerland.eda.admin.ch/en/transport, 2024.

[17] Geotechnik Schweiz, “Gotthard base tunnel,” https://geotechnikschweiz.ch/?page_id=3972&lang=en, 2025.

[18] S. Duranton, A. Audier, J. Hazan, M. P. Langhorn, and V. Gauche, “The 2017 european railway performance index,” *Boston Consulting Group*, 2017. [Online]. Available: https://www.bcg.com/publications/2017/transportation-travel-tourism-2017-european-railway-performance-index

[19] European Passengers’ Federation, “Punctuality report of european trains,” https://www.epf.eu/wp/10929-2/#:~:text=The%20diagram%20below%20shows%20the%20punctuality%20of%20medium%2D%20and%20short,volumes%20in%20the%20railway%20systems, 2014.

[20] scikit-learn developers, “Machine learning in python,” https://scikit-learn.org/stable/, 2025.

[21] XGBoost Developers, “Xgboost python package documentation,” https://xgboost.readthedocs.io/en/latest/python/python_api.html, 2025.

[22] scikit-learn developers, “Linearregression,” 2025. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[23] ——, “Decisiontreeregressor,” https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegresso 2025.

[24] Open Data Platform Mobility Switzerland, “Cookbook brief description,” https://opentransportdata.swiss/en/cookbook/, 2025.

[25] ——, “Big picture,” https://opentransportdata.swiss/en/cookbook/big-picture/, 2025.

[26] Systemaufgaben Kindeninformation SKI, “Escalation process,” https://www.oev-info.ch/de/datenmanagement/datenqualitaet/eskalationsprozess, 2025.

[27] Open Data Platform Mobility Switzerland, “Actual data short description,” https://opentransportdata.swiss/en/cookbook/historic-and-statistics-cookbook/actual-data/, 2025.

[28] ——, “Ist-daten kurzbeschreibung,” https://opentransportdata.swiss/de/cookbook/historic-and-statistics-cookbook/actual-data/, 2025.

[29] J. Crobak, “parquet-python: A pure-python implementation of the parquet format,” https://pypi.org/project/parquet/, 2020.

[30] scikit-learn developers, “Labelencoder — scikit-learn 1.6.1 documentation,” https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html, 2025.

[31] J. Schubert and ABITZ.COM, “Zugfinder live map europe,” https://www.zugfinder.net/en/livemap-europa, 2024.

[32] NumFOCUS, Inc., “pandas.datetimeindex.dayofweek,” https://pandas.pydata.org/docs/reference/api/pandas.Dateti 2024.

[33] scikit-learn developers, “sklearn.model.selection.cross_validate — scikit-learn 1.6.1 documentation,” https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_validate.html, 2025.

# A

# Code of Project

**Listing A.1:** Setup for models: Reading parquet into dataframe, feature selection, train-test split, cross-validation.

```
df_MLready = pd.read_parquet('df_MLready.parquet')

# Features and target
features = ['Month', 'DayofWeek', 'Hour', 'Minute', 'STOP_NAME_ENC', '
    PREV_STOP_ENC']
X = df_MLready[features]
y = df_MLready['Delta_seconds']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=14)

# Define 5-fold cross-validation
```

```
cv = KFold(n_splits=5, shuffle=True, random_state=14)

# Create df_pred for visualizations
df_pred = pd.DataFrame(index=range(len(y_test)))
df_pred['y_test'] = y_test.reset_index(drop=True)
```

**Listing A.2:** Linear Regressor

```
model = LinearRegression()

# Fit model
model.fit(X_train, y_train)

y_pred_lin = model.predict(X_test)
df_pred['y_pred_lin'] = pd.Series(y_pred_lin, index=df_pred.index)

errors_lin = np.abs(y_test.reset_index(drop=True) - y_pred_lin)
df_pred['errors_lin'] = errors_lin


r2_scores = cross_val_score(model, X_train, y_train, cv=cv, scoring='r2')
print("Cross-validated [U+FFFD]cores:", r2_scores)
print(f"Average Cross-validated [U+FFFD]{r2_scores.mean():.4f}")

rmse = np.sqrt(mean_squared_error(y_test, y_pred_lin))
print(f"Test RMSE (seconds): {rmse:.2f}")

coefficients = pd.Series(model.coef_, index=features)
print("Intercept:", model.intercept_)
print("Coefficients:")
print(coefficients)

thresholds_lin = {
    "Within 1 minute": errors_lin <= 60,
    "Within 3 minutes": (errors_lin > 60) & (errors_lin <= 180),
    "Within 5 minutes": (errors_lin > 180) & (errors_lin <= 300),
    "More than 5 minutes": errors_lin > 300
}
```

```
31
32 summary_lin = {"Category": [], "Count": [], "Percentage": []}
33 total = len(errors_lin)
34
35 for category, condition in thresholds_lin.items():
36     count = condition.sum()
37     percentage = round(100 * count / total, 3)
38     summary_lin["Category"].append(category)
39     summary_lin["Count"].append(count)
40     summary_lin["Percentage"].append(percentage)
41
42 summary_lin_df = pd.DataFrame(summary_lin)
43 print("\nPrediction Accuracy Summary:")
44 print(summary_lin_df)
```

**Listing A.3:** Decision Tree Regressor

```
1 # Hyperparameter space
2 param_distributions = {
3     'max_depth': [3, 5, 10, 15, None],
4     'min_samples_split': [2, 5, 10],
5     'min_samples_leaf': [1, 2, 5, 10],
6     'max_features': [None, 'sqrt', 'log2']
7 }
8
9 # Randomized search
10 grid_search = RandomizedSearchCV(
11     estimator=DecisionTreeRegressor(random_state=14),
12     param_distributions=param_distributions,
13     scoring='r2',
14     cv=5,
15     n_iter=20,
16     random_state=14,
17     n_jobs=-1,
18     verbose=0
19 )
20
21 # Fit model
```

```
grid_search.fit(X_train, y_train)

best_model = grid_search.best_estimator_
y_pred_tree = best_model.predict(X_test)
df_pred['y_pred_tree'] = pd.Series(y_pred_tree, index=df_pred.index)

errors_tree = np.abs(y_test.reset_index(drop=True) - y_pred_tree)
df_pred['errors_tree'] = errors_tree


# Evaluate
r2 = r2_score(y_test, y_pred_tree)
rmse = np.sqrt(mean_squared_error(y_test, y_pred_tree))
print(f"Best Parameters: {grid_search.best_params_}")
print(f"Test R\U+FFFD{r2:.4f}")
print(f"Test RMSE: {rmse:.2f} seconds")


thresholds_tree = {
    "Within 1 minute": errors_tree <= 60,
    "Within 3 minutes": (errors_tree > 60) & (errors_tree <= 180),
    "Within 5 minutes": (errors_tree > 180) & (errors_tree <= 300),
    "More than 5 minutes": errors_tree > 300
}

summary_tree = {"Category": [], "Count": [], "Percentage": []}
total = len(errors_tree)

for category, condition in thresholds_tree.items():
    count = condition.sum()
    percentage = round(100 * count / total, 3)
    summary_tree["Category"].append(category)
    summary_tree["Count"].append(count)
    summary_tree["Percentage"].append(percentage)

summary_tree_df = pd.DataFrame(summary_tree)
print("\nPrediction Accuracy Summary:")
print(summary_tree_df)
```

**Listing A.4:** XGB Regressor

```
# Hyperparameter space
param_distributions = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7, 10],
    'learning_rate': [0.01, 0.05, 0.1, 0.2],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0]
}

# Randomized search
grid_search = RandomizedSearchCV(
    estimator=XGBRegressor(random_state=14, verbosity=0),
    param_distributions=param_distributions,
    scoring='r2',
    cv=cv,
    n_iter=20,
    random_state=14,
    n_jobs=-1,
    verbose=0
)

# Fit model
grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_


y_pred_xgb = best_model.predict(X_test)
#y_test = y_test.reset_index(drop=True)
df_pred['y_pred_xgb'] = pd.Series(y_pred_xgb, index=df_pred.index)

errors_xgb = np.abs(y_test.reset_index(drop=True) - y_pred_xgb)
df_pred['errors_xgb'] = errors_xgb


# Evaluate
r2 = r2_score(y_test, y_pred_xgb)
rmse = np.sqrt(mean_squared_error(y_test, y_pred_xgb))
```

```
print(f"Best Parameters: {grid_search.best_params_}")
print(f"Test R²: {r2:.4f}")
print(f"Test RMSE: {rmse:.2f} seconds")



thresholds_xgb = {
    "Within 1 minute": errors_xgb <= 60,
    "Within 3 minutes": (errors_xgb > 60) & (errors_xgb <= 180),
    "Within 5 minutes": (errors_xgb > 180) & (errors_xgb <= 300),
    "More than 5 minutes": errors_xgb > 300
}

summary_xgb = {"Category": [], "Count": [], "Percentage": []}

total = len(errors_xgb)
for category, condition in thresholds_xgb.items():
    count = condition.sum()
    percentage = round(100 * count / total, 3)
    summary_xgb["Category"].append(category)
    summary_xgb["Count"].append(count)
    summary_xgb["Percentage"].append(percentage)

summary_xgb_df = pd.DataFrame(summary_xgb)
print("\nPrediction Accuracy Summary:")
print(summary_xgb_df)
```