Figure 1: Malmö Central Station underground platforms

Real-time train delay prediction in Skåne

A complete ML project with xgboost model development and pipelines for real-time inference over streamed data.

Examensarbete, 15YHp, Utvecklare inom AI och maskininlärning 400YHp

Handledare: Johannes Öhlin - Tenfifty
Examinator: Kokchun Giang
Klass: AI21

LÉO LASSARADE

Göteborg, Sweden 2023-05

Abstract

The aim of this work is to propose a machine learning model able to output a real time delay prediction on commuter train services; based on current the current state of the network. For data availability reasons, the problem is limited to regional commuter traffic in region Skåne. Acquiring the historical real-time data for the model to be trained involved a consequent data-engineering process. The pipeline is extensively detailed. A Gradient boosted tree model is proposed and evaluated, and an ETL pipeline for doing inference on streaming data is also built and detailed in this paper. A roadmap for testing against inbuilt estimators is also proposed as a part of MLops.

Contents

1 Introduction

During the year 2019, 47 million travels were made on commuter trains in region Skåne (Gomér, 2021). While those volumes have significantly decreased during the pandemic years 2020 and onwards, it was announced that the pre-pandemic number of travels were reached back by july 2022 (Nybogård, 2022). Although most of the commuter trains do arrive on time (Abboud Ado, 2023) in terms the Swedish transport regulator Transportstyrelsen definition of on time — which is within 5 mn after timetable time — disruptions and the resulting delays are the origin of challenging situations for both the operators and the consumers of those millions of travels.

Knowing precisely about delays may help some users to make choices that will bring them to their destination on time, in the context of a dense multimodal offering. Meaning that users who have the possibility of making quick decisions whether to switch to a bus, cycle, another train, or any available mean, could benefit from very precise delay estimations. I found relevance in aiming to better the existing estimates.

Extensive efforts have been made on publishing public transportation related data in Sweden — namely services as TrafikLab, and the Trafikverket API among others — and I considered this vast amount of data produced and published as an opportunity for a machine learning application (ML), that could address the problem of giving accurate real time delay estimations.

The work is based on an assumption about the nature of a railway system: that it is an intertwined network where consequences are chained, and where disruptions on certain end could affect another end. Translated into data requirements, it means the need of a dataset that is a snapshot of the whole state of the network in region Skåne at a given moment. Such data exists in the form of historical real-time data. Rise's project KoDa, as for Kollektivtrafikens datalabb, made such data available, which I fetched. I also fetched historical weather data from SMHI, as another assumption is that weather conditions can have an impact on railway systems. The choice of Skåne for this machine learning application is based on two main characteristics:

- **The density of its public transport system**. Regional commuting with Pågatåg and trans-frontier commuting to Denmark with Öresundståg — traveling through important intersections such as Malmö, Helsingborg, Hässleholm to name a few — makes it an interesting laboratory for aiming to predict delay with a better level of accuracy.
- **Data availability and quality**. I identified region Skåne as a good publisher of data over Trafiklab. It is the one of only two regions that, by the start of this work, had not only made available static and real-time data, but also vehicle positions and occupancy data.

This work's context is the one of a vocational school (Swedish Yrkeshögskola) and its objective is to address most of the aspects of building a concrete AI application; aspects that could be encountered in the context of a practical work assignment in the field of AI. Even though AI has been used in the past for building such services for commercial use, I believe some of the approaches to the problem are original.

This paper begins with a methodology for acquiring the historical data necessary for building the model, another one for transforming the streamed data for adapting it to the model. The conception of the machine learning model is detailed, as well as the results. Finally, some possible future explorations, notably in MLops are presented.

1.1 Purpose and objective

1.1.1 Purpose

The purpose of this work is to build a machine learning model outputting a regression predicted delay value in seconds at a given train station ahead on a train's route, given the current state of the network and the weather conditions. The model should be usable in a production deployment, meaning using live streaming data for doing inference, and serve the delay estimates.

1.1.2 Objective

The objective is to propose a realistic delay prevision model with a minimum absolute mean error. The model should perform well in unusual traffic conditions, and it should be able to make inference from the real-time updates streamed by the regional operators. Furthermore, this work is convceived as a ground for further development, with future objectives such as outperforming the standard predictions included with real-time data, prioritizing in unusual traffic conditions.

2 Method

A premise for developing the desired model is access to historical real-time data as mentioned in the introduction. As we address a supervised ML problem, the first step is acquiring an annotated training dataset.

The training dataset has the form of historical data. There are two kinds of historical data available: GTFS real-time —which provides movement updates and is refreshed a few times during a minute — and GTFS static, updated daily, which provides all the static information such as timetables, routes, stations and such. It is the combination of those two that allows to obtain the target metric: measured delay in seconds. GTFS stands for General Transit Feed Specification, it is a consistent and stable format curated by Google and intended to guide public transport operators in publishing their data for developers to use, benefiting to everyone working with transportation data.

After acquiring a few daily samples from the KoDa API for securing knowledge of data nature and structure, the feasibility of knowing historical delays with absolute certainty could be established. Also, the availability of features on streamed data from the Trafiklab Regional API was verified to ensure that the stream includes — or allows the engineering — of the same features as the historical data the model is trained on.

While fetching a few daily samples was not a consequent work, fetching the whole GTFS data required extensive ETL operations, to address problem of storage and time-management. A pipeline was built, with API calls for downloading raw data at its first step, it outputs csv files for each month of historical data. The data outputted by the pipeline represents roughly 5e-3% of the raw data. This pipeline is detailed in 2.1.1.

A second pipeline was built to transform the current real-time stream into suitable inputs for the model to do inference on. As for the previous pipeline, it is also extensively detailed in 2.1.2.

A xgboost model was trained and parameters evaluated against a validation dataset. The model is tested for inference on streamed data.

2.1 Data engineering

2.1.1 Over historical data

It appeared that an historical of train delays at any given station was not a given dataset to find. KoDa was to my knowledge the only data source in the public domain that I could use for building the delay metric. The main advantage is that the data provided is in a format consistent with Trafiklab's stream data provided by train operators.

Initially a Apache Cassandra database was part of the KoDa project, but as KoDa is a research project, the discontinuation of funding meant termination of public access for this database. Fortunately, two APIs, one for fetching historical static and one for fetching historical real-time are kept up. A main downside is the absence of querying that the database would have allowed, the API's only option being to download all the raw data for one day. Nevertheless, I received support from the team that originally built the project at RISE, notably in confirming gaps in data availability so I would not run the pipeline over some dates in vain.

The available historical data range stretches from February 5 2020, to late 2022. With knowledge of periodic gaps, I ideally expected valid data for 730 days.

The realtime collection of protobuf files for each day weight around 2.5 gb, and require a preparation time up to 15mn on server side. Also, only 3 requests can be processed at the same time without impairing the API server's performance.

A quick calculus of 2.5 gb over 730 days, gives 1.8 terra of protobuff files, which resemble json/python dictionary format, but lighter. Another quick calculus gives a theorical requirement of about 61 hours of downtime-free fetching time. This was a real concern as it would consume a lot of the time allowed for this project before even beginning with working with advanced data exploration and machine learning. The main reason for still choosing to fetch all available data, was to leave open the possibility of integrating seasonality in the modelling. Also, I had confidence in the quality of the data and its usability for the problem.

I built a transformation pipeline with both the purpose of filtering and retaining data related to my problem as well as managing storage space and automate the deletion of unused data. Also, I needed to handle exceptions so that the pipeline would not be affected by non-available data at certain days. Besides, eventhough GTFS format shouldn't be mutable, there is room for interpretation in the fields, so the operator can change the location where a categorical information is stored, in my case where the train operator name I filtered on was located. This is an example of a case that could break the pipeline and stretch on the data acquisition time, as further python development for handling exceptions was needed at times.

The following schema represent the pipeline in its whole, and it will be described thereafter.

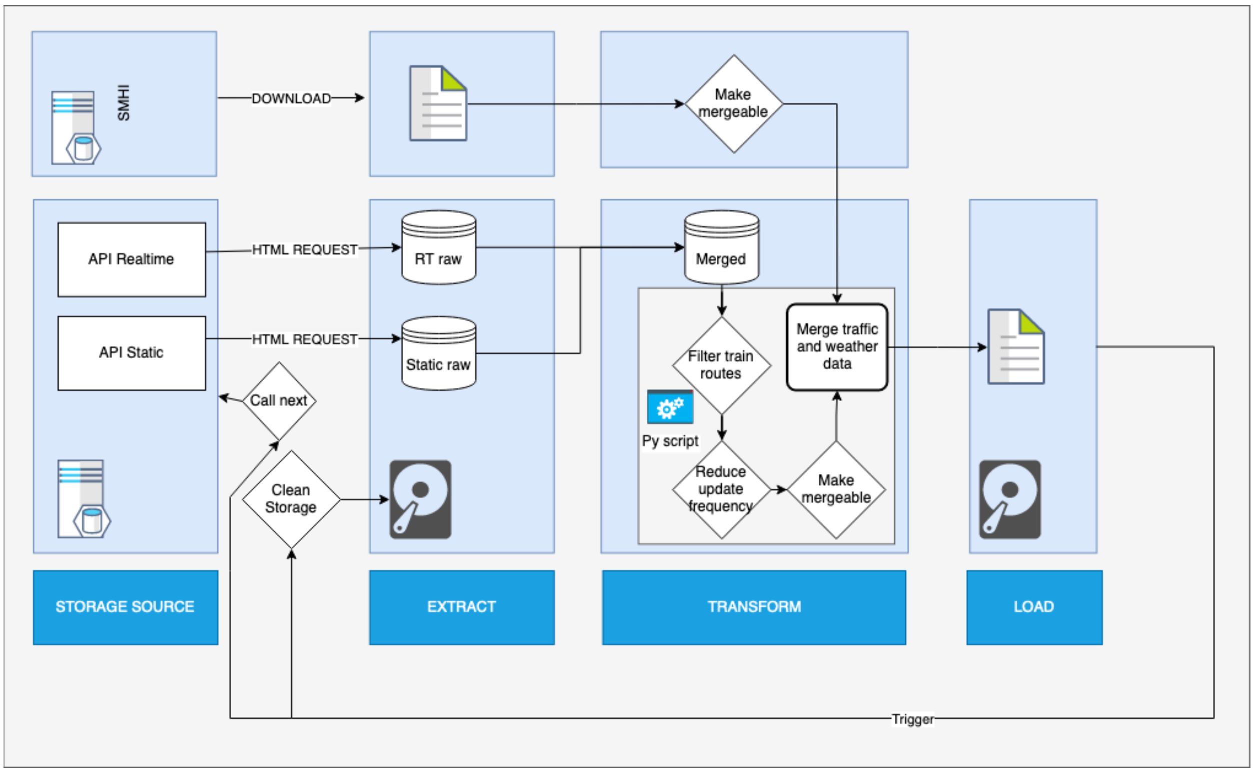


Figure 2: Pipeline for fetching historical data

First, two disctinct api calls are made to KoDa servers. Static and real-time data for a day are obtained, and they have two distinct data structures. They have the advantage to both follow a standard, but they require preliminary transformations for them to be coupled.

A markdown guide for coupling static and real-time GTFS is provided in the annexes. It results in a pandas DataFrame upon which transformations can be applied.

The data for each day then go through transformations. The main one is sizing down realtime-data update frequency from 14 seconds to 5 minutes updates; an update frequency I found satisfactory regarding the problem. This is around 1/21,5 the share of the dataset in its original condition. The other important transformation is to only retain train routes for matching trip id:s in .pb files. As train trips are a minority of all trips at daily data, this implies a great reduction in data volume as well. Finally, for my target variable to be valid, the delay had to be measured with certainty. In the whole data, only some data points corresponding to stopping at a certain station fulfilled this constraint of being ground-truth. Acquiring metrological data was easier, it is just a single csv download with an history for each metric. It's coupling only requiring formatting and merging with the pandas dataframe created earlier. A fully transformed day worth of data has a size between 5 and 10 mb and is a .csv file.

When the pipeline had ran around 700 times and the csvs were concatenated, the whole training dataset consists of a bit over 600000 samples, weighting 109 mb. Namely 0,005% of the original data volume.

2.1.2 Over streamed data

The historical real-time data and static data structure are almost identical to the streamed data, which was helpful while building the python transformation scripts. The APIs for streamed static and real-time are provided by TrafiKlab. SMHI's API is also used for weather data.

In the feature engineering section of this work, I will present aggregates over time, a span of 20 minutes. Using time windows over the snapshot times provided — which is a feature proper to historical real-time data — have implication on the stream data pipeline and requires additional scripting.

As data is streamed, it implies a pre-loading of at least 20 minutes for building and adding the aggregation features reflecting the network status. This is essential to augmenting the data points with those features for inference to be made on. By pre-loading is meant the collection of raw data, filtering of valid samples, then a concatenation of the snapshots, ended with aggregations. The process is schematised in the figure below:

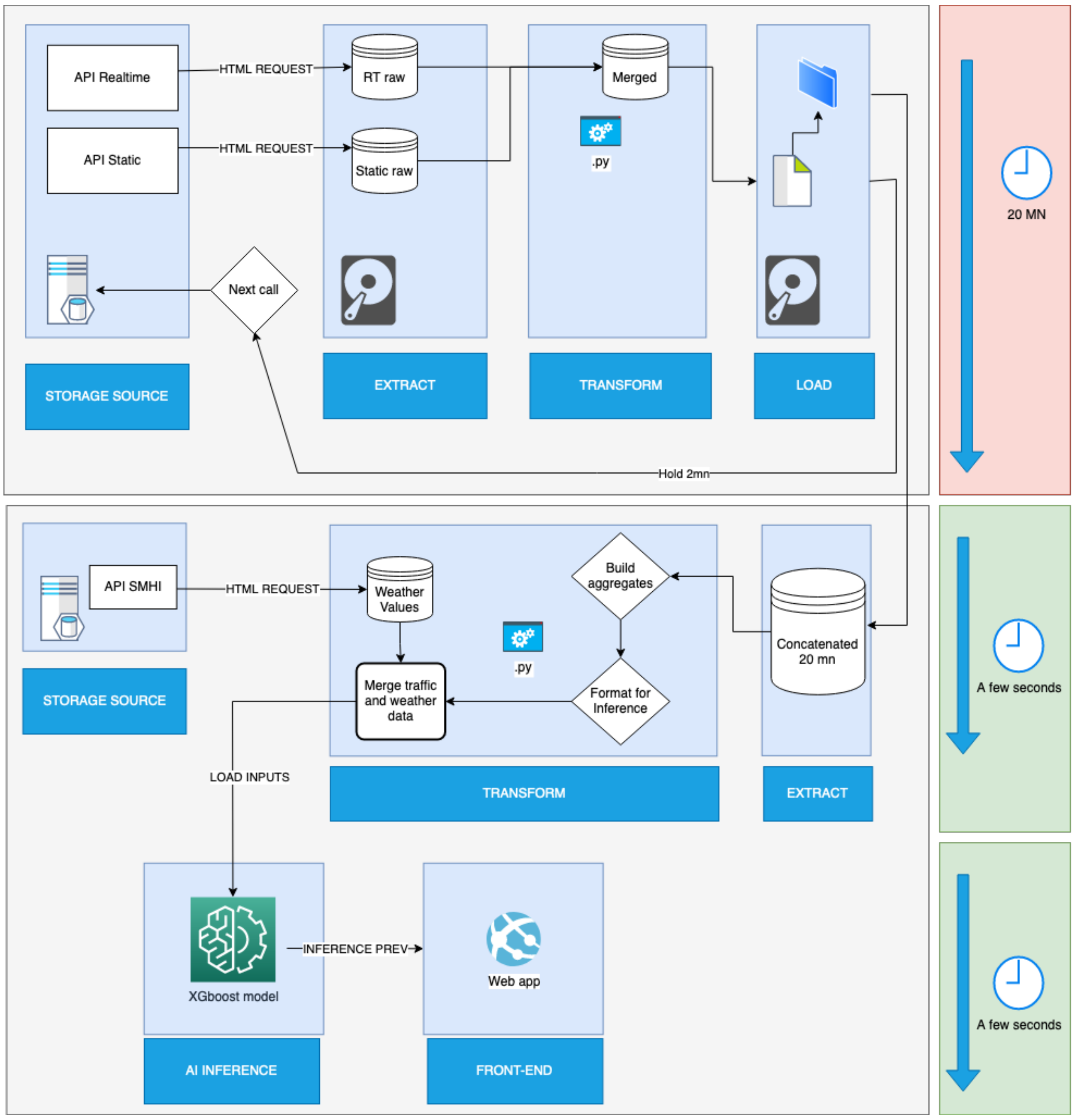


Figure 3: Pipeline for fetching historical data

There are two approaches to running the pipeline:

- If the model is to be used at any time, it would require the pipeline to constantly run as soon as commuter train traffic start each day, so the state of the network is always available to be added to data points fed into the machine learning model. This is a good application for cloud, involving triggers.

- For the research in this project, I need to start the program and fetch data 20mn before I am to make inference on the latest real-time data.

2.2 Feature selection and feature engineering

Coupling a train's live status with static data implies that a datapoint provided by an operator comes along with around 30 features. Given this number of features, and the availability to couple external features such as weather based on datetime, there is an interesting ground for engineering a tabular data-based model.

One advantage of a train network is that it has a very stable static component. Stations don't move around, and new railway is not built at a tremendous pace. This is in fact that which allows building reliable timetables from the beginning.

A consequent part of features is therefore originally static information, giving a sense of position and distance over the network. These features are ordinal categorical and numeric.

- **stop_sequence_x**: The ordered stop number on the route.
- **stop_id**: A unique identifier for a train station.
- **Direction_id**: Gives a sense of direction on the route.
- **Trips_last_known_stop**: If the station is the train's last stop on the route.
- **Route_short_name**: A unique identifier for the route.

Although I assume that those components are subject to few changes over time, there are problems that can upcome that I address in MLops section.

The training data stretches only over two years. Therefore, I chose to limit the time-bounded features to day of the week, hour of the trip and week of the year. I rule out the possibility of picturing a trend with certainty over a longer term.

As weather conditions are often quoted as an influencer of traffic conditions (Stenström, Famurewa, Aditya, & Galar, 2012), I retained two metrics taken from SMHI databases.

- **Lufttemperatur** (Air temperature). Unusually cold or warm impacts the train infrastructure. Warm weather can cause a heat curve on tracks, and extremely cold weather can weaken the infrastructure in different ways.

- **Rådande väder**(Weather observation). categorical, can help making further sense of a simple degrees centigrade measurement.

The feature engineering is about providing a sense of whole-network status. To stick the idea of knowing the current state of the network, I retained three metrics. Mean delay and Standard deviation of delay, both over a 20mn window. Also, a feature is the rate of the on time train over the period, which is a common metric when assessing a transport network performance. The window of 20 mn was selected as it is assumed not too sensitive and still enough to picture a trend, but that is open to discussion with a field expert.

The third is last known delay. This is the last known delay at a station prior to arriving to the current station. This feature imply certainty, and as certainty cannot be established at every station, it implies some missing values for this feature. This was taken into consideration while choosing the ML model.

2.3 Model selection

Although I am open to testing more models in the future for performing regression over my data, I've been confident in the choice of a gradient boosted tree solution as a first option. One motivation is that training xgboost is fast, resource-and time efficient, especially compared to a deep learning model — which they also are competitive against on tabular problems (Grinsztajn, Oyallon, & Varoquaux, 2022) — is fitting well with my time constraint.

Thorough the feature engineering phase, I built knowledge about the nature of my data that are consistent with this choice. Gradient Boosted Trees (GBTs) are a machine learning technique that have proven to be effective on tabular datasets for several reasons. First, It works well with non-linearity. Tabular datasets often have complex non-linear relationships between input features and output variables. GBTs can model these non-linear relationships effectively, using a series of simple tree-based models that are combined to make accurate predictions.

The observation of the correlation matrix over the selected features confirmed a low linear relationship between those features.

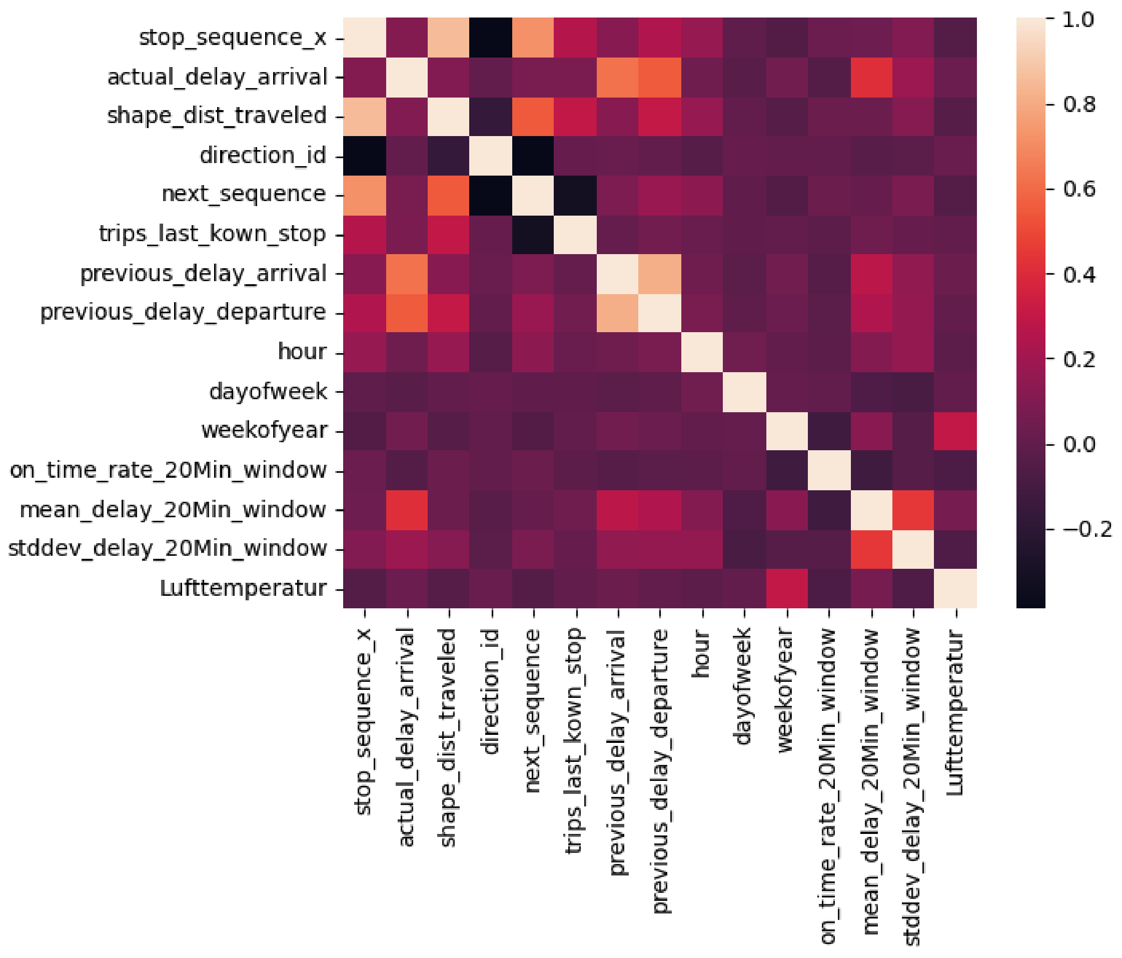


Figure 4: Correlation matrix over the selected features for modelling

A general advantage of a GBT model is it's ability to assess feature importance. GBTs can identify the most important input features in a dataset and assign them higher weights during training. This can help to reduce the impact of noisy or irrelevant features and focus on the most relevant ones for making accurate predictions. Finally, robustness. GBTs are generally robust to outliers and missing data and can handle imbalanced datasets well. This is because they can learn from errors and focus on the areas of the dataset that are most important for making accurate predictions. This is especially helpful regarding my dataset, which present both some missing values and imbalance.

I knew from the field research that most of delay values over the included train routes would be distributed around 0. Commuter trains — although subject to a slight decreasing trend in accuracy performance — were are on time by around 92% in 2022. (Abboud Ado, 2023). The following figure is a comprehensive display of this balance.

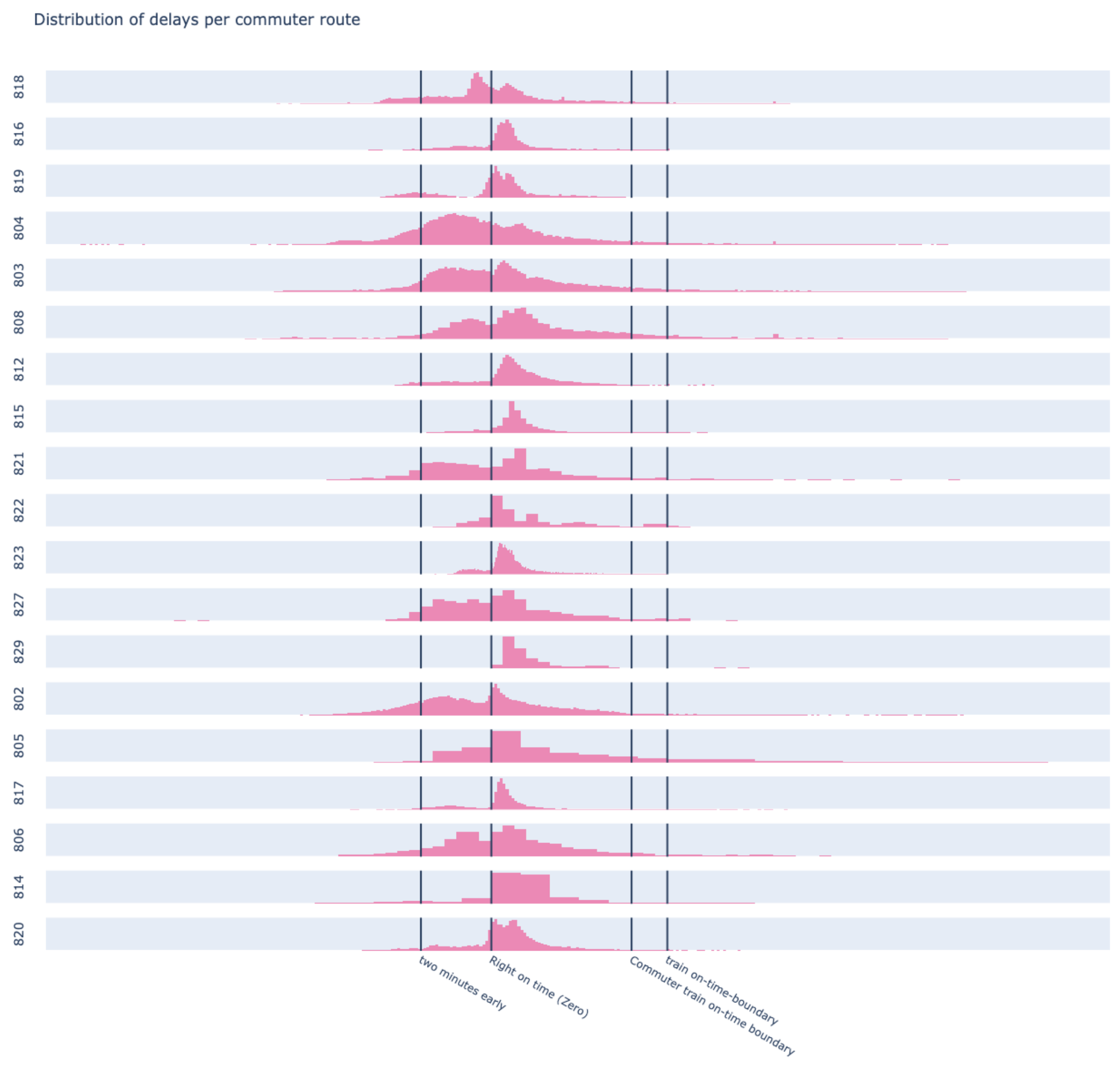


Figure 5: Delay distribution over the train routes included in the training data

Such imbalance of the training data had to be tackled by applying weighting over samples, so that the model would be better at generalizing, and especially over datapoints that represent large delays. Not just be very good at predicting values around 0. Training samples associated to delays over 4 and then 5 minutes were given a larger weight.

A last positive is that XGboost as of version 1.6 supports categorical values. It is a strong case for advocating the model, as the one-hot encoding that would otherwise have been needed to handle the three categorical features of the training dataset would be a performance issue while working with a GBT. (Jarrett, 2023)

2.4 Validation and testing

The parameters for the model were established by grid search cross validation. Dealing with time series data, shuffle of the samples was avoided using the module TimeSeriesSplit from sklearn. 5 splits were established in this variant of K-fold. This also results in creating a welcomed gap between train and test sets.

Best parameters were initially retained after a first round of validation on unweighted samples. They were discarded as weighting came into place and a second CV was performed before proceeding to testing. The best parameters are colsample_bytree: 0.8, 'eta': 0.1, max_depth: 7, n_estimators: 100, subsample: 1.0.

Those parameters are acceptable for addressing concerns of over-fitting. Colsample by tree, which has a default value of 1, is the subsample ratio of columns when constructing each tree. All features aren't included while a tree is constructed. Eta which is the learning rate is also set under the default of 0.3, a lower value can help avoiding over-fitting. A max depth of 7 is retained, just above the default value of 6. A deeper and therefore more complex tree can result in better accuracy, yet it can result in over-fitting as well. A value between 3-10 is to be preferred. n_estimators is kept to default 100. Finally, we have a subsample value of 1, meaning that all training data is kept.

The objective is aimed at reducing the Mean Absolute Error (MAE), are we are interested in an intelligible value that is a delay in second, where the aim is to be as accurate as possible. A model was trained using those parameters, and tests were made on the remaining 20% of the dataset, those being the most recent months.

At this point, it is possible to display the feature importance.

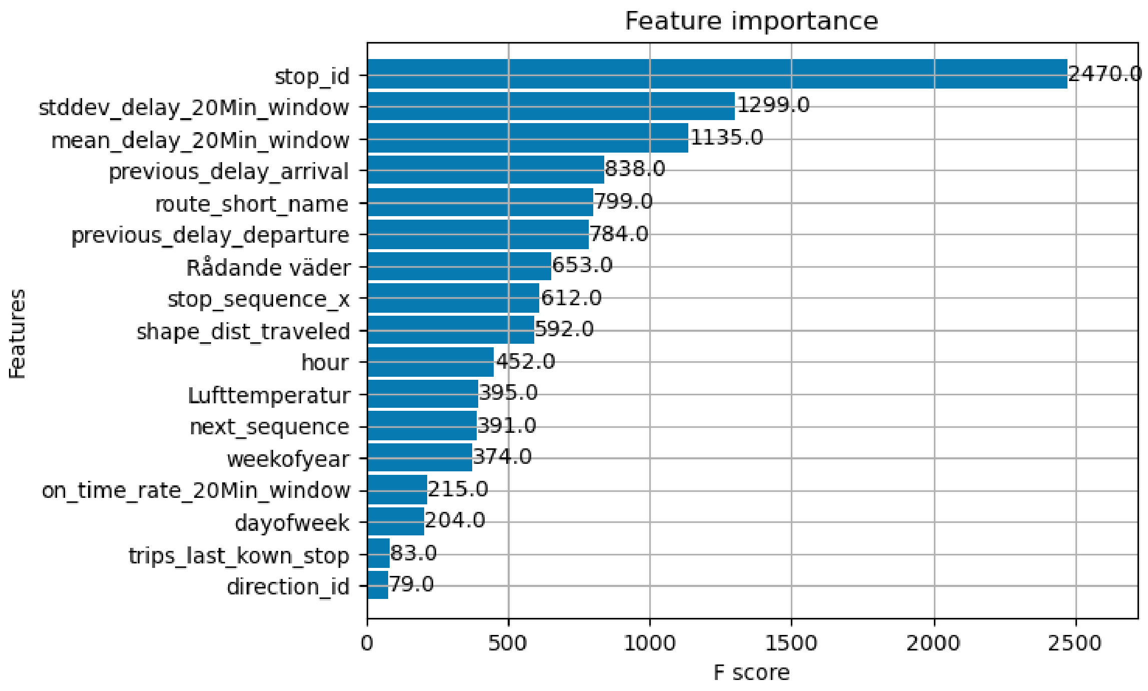


Figure 6: Feature importance of the latest trained xgboost model

The stop id is proeminent and associates delay to specific train stations. As a constraint of building a training dataset was to only retain delays that had been observed with certainty, and that this certainty could only be established at certain stations, many stations absent in the training data. The case of an uknown stop id will be a common case in production setting. Fortunately, the robustness of a GBT model named earlier comes in play at this point. The model can handle this missing value, and use other features to provide the estimation. Yet this observation over this specific feature it not to be minimized.

Generally, the features representing the network state are ranking high, which comforts me as them being the backbone of the model. Also, the additional features for weather have significant importance, likewise comforting the choice of including weather data in the model. Finally, positional features seem to have slightly more importance than the temporal ones, which are yet not meaningless either, especially the hour of the trip.

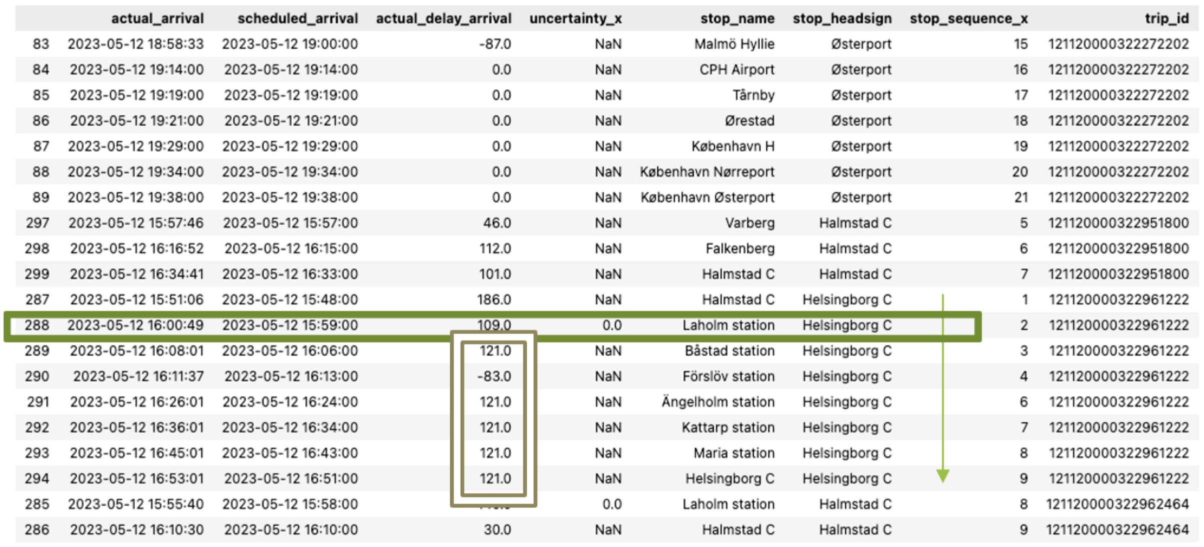
2.5 Further testing and MLops

While using the historical dataset is a good ground for approaching a solid parametrization and build a first functioning model, having built a second pipeline for making inference on streamed data opens for rearticulating the testing over newer data.

The error can be measured and collected over new samples. Also, the predictions of the current model can be compared to the values proposed by the inbuilt model.

The collection of new samples could approach what had first been intended with KodA, ressembling a collection of historical data. Once enough data is collected, the model could be trained again at regular intervals. Weighting of new samples could also be implemented.

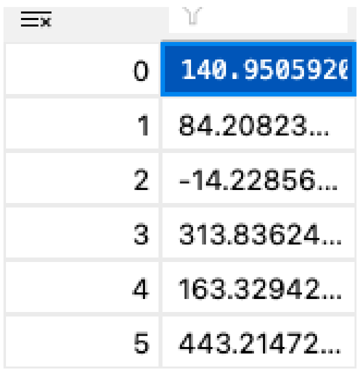
The table therunder represent streamed data.



	actual_arrival	scheduled_arrival	actual_delay_arrival	uncertainty_x	stop_name	stop_headsign	stop_sequence_x	trip_id
83	2023-05-12 18:58:33	2023-05-12 19:00:00	-87.0	NaN	Malmö Hyllie	Østerport	15	121120000322272202
84	2023-05-12 19:14:00	2023-05-12 19:14:00	0.0	NaN	CPH Airport	Østerport	16	121120000322272202
85	2023-05-12 19:19:00	2023-05-12 19:19:00	0.0	NaN	Tårnby	Østerport	17	121120000322272202
86	2023-05-12 19:21:00	2023-05-12 19:21:00	0.0	NaN	Ørestad	Østerport	18	121120000322272202
87	2023-05-12 19:29:00	2023-05-12 19:29:00	0.0	NaN	København H	Østerport	19	121120000322272202
88	2023-05-12 19:34:00	2023-05-12 19:34:00	0.0	NaN	København Nørreport	Østerport	20	121120000322272202
89	2023-05-12 19:38:00	2023-05-12 19:38:00	0.0	NaN	København Østerport	Østerport	21	121120000322272202
297	2023-05-12 15:57:46	2023-05-12 15:57:00	46.0	NaN	Varberg	Halmstad C	5	121120000322951800
298	2023-05-12 16:16:52	2023-05-12 16:15:00	112.0	NaN	Falkenberg	Halmstad C	6	121120000322951800
299	2023-05-12 16:34:41	2023-05-12 16:33:00	101.0	NaN	Halmstad C	Halmstad C	7	121120000322951800
287	2023-05-12 15:51:06	2023-05-12 15:48:00	186.0	NaN	Halmstad C	Helsingborg C	1	121120000322961222
288	2023-05-12 16:00:49	2023-05-12 15:59:00	109.0	0.0	Laholm station	Helsingborg C	2	121120000322961222
289	2023-05-12 16:08:01	2023-05-12 16:06:00	121.0	NaN	Båstad station	Helsingborg C	3	121120000322961222
290	2023-05-12 16:11:37	2023-05-12 16:13:00	-83.0	NaN	Förslöv station	Helsingborg C	4	121120000322961222
291	2023-05-12 16:26:01	2023-05-12 16:24:00	121.0	NaN	Ängelholm station	Helsingborg C	6	121120000322961222
292	2023-05-12 16:36:01	2023-05-12 16:34:00	121.0	NaN	Kattarp station	Helsingborg C	7	121120000322961222
293	2023-05-12 16:45:01	2023-05-12 16:43:00	121.0	NaN	Maria station	Helsingborg C	8	121120000322961222
294	2023-05-12 16:53:01	2023-05-12 16:51:00	121.0	NaN	Helsingborg C	Helsingborg C	9	121120000322961222
285	2023-05-12 15:55:40	2023-05-12 15:58:00	[illegible]	0.0	Laholm station	Halmstad C	8	121120000322962464
286	2023-05-12 16:10:30	2023-05-12 16:10:00	30.0	NaN	Halmstad C	Halmstad C	9	121120000322962464

Figure 7: A comprehensive tabular view of a streamed data used for inference

One trip is highlighted. At the moment of the stream, or in other words at the snapshot's time, we can see that the train had reached Laholm station with a delay of 109 seconds. Having a known delay (Uncertainty_x = 0) and aggregates about the network status, the model is able to predict the future delays on the route, see figure under. Note that aggregates are not visible on the figure, which is aimed at comprehensive human reading.



0	140.9505920
1	84.20823...
2	-14.22856...
3	313.83624...
4	163.32942...
5	443.21472...

Figure 8: The xgboost model's prediction of delay (s) on remaining stations on the trip

Each of these 6 values can compare both with the inbuilt model which proposed an array of delays as following: 121, -83 , 121,121,121,121. Furthermore as the snapshots are continually loaded, the train will reach stations where uncertainty is 0, and those points can be compared to the model's predictions. Such certain datapoints with a comparable target can be stored for further training of the model. Actual_delay_arrival with an uncertainty_x of NaN is in fact an inbuilt prediction.

Finally, any structural change of the network should be reflected in the model. An awareness of the actuality about the network is essential, so no works that have long term consequence about the network performance should be forgotten when they happen.

IT-HÖGSKOLAN

Här startar din IT-karriär.

3 Results and discussion

3.1 Results

Training the model with the selected parameters and making inference on the test dataset resulted in a MAE of 126.63 seconds. The score is not as good on large delay prevision as it is down to 306,87 seconds. Yet, it is an improvement of around 50 seconds regarding large delays compared to the first test processed on the unweighted dataset.

3.2 Discussion and future directions

While the MAE on previsions of greater delays could be largerly improved, the main objective of building a realistic model for predicting delays on commuter train traffic in region Skåne is reached. Time limitations have restrained an in-depth bettering of the model using historical data. In fact, the model has only been trained on two different sets of best parameters from cross validation, and on one attempt of weight assignation over training samples. It is planned to further conduct extensive testing of the model, both on historical data and using streaming data as mentionned in the MLops section. Such planning for further training is allowed by the completion of the work's second objective of building a pipeline for making inference on streamed data.

Further feature engineering is also possible. Among the ideas for features, is a cross relational metric among the routes. Where delays on routes directly linked to other through knots could be weighted in relation to eachother. The idea is that delays on a connected route could affect the other routes differently, and allow to gain precision compared to a general aggreate over all the network.

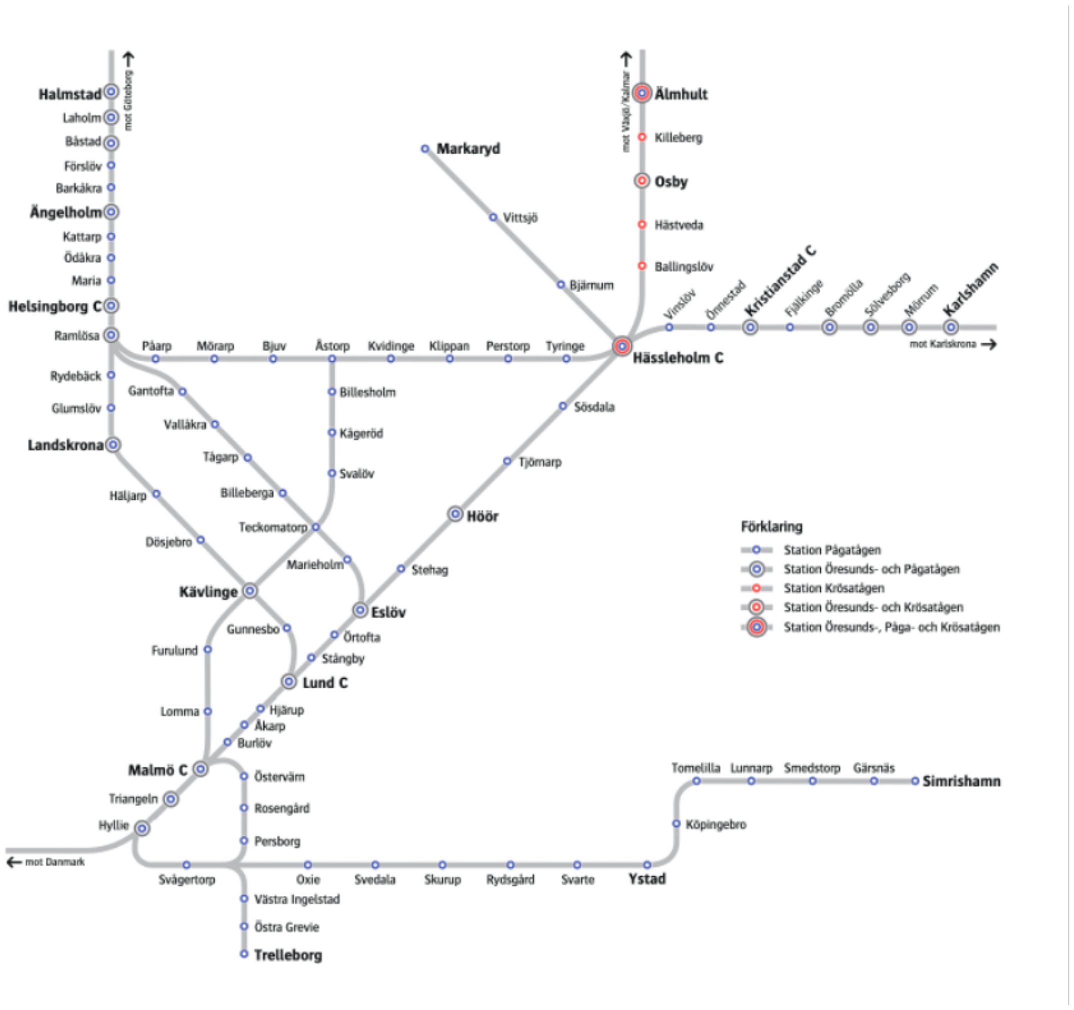


Figure 9: Skånes commuter train network is a combination of interconected routes

Among the many possibilities of enriching the model, It is feasible to couple historical data for traffic disruption announcements and gps positioning.

4 Conclusion

At the early stage of choosing a topic for my final work, I was certain that I wanted to address transportation related problems. Having worked consistently with time series and machine learning algorithms under two internships at the AI consultancy firm Tenfifty, and with the certitude that AI can bring much value in the field of transport; working on train delays naturally became the topic of choice.

The original intent was to quickly begin to work on machine learning and eventually deep learning models, using available data. Then, there was no such thing as available data. Realising the complexity of acquiring appropriate data and preparing it in such a way to address my problem correctly transformed the focus of the work dramatically.

I had to get used to whole new file formats and protocols, as well as problems of storage and a time dimensionality that I had not encountered until the beginning of this work. Also, my priority was to be able to demonstrate the functioning of the model over streamed data at the time of this work's presentation. Therefore, the pipelining amounted for roughly two third of the available time for this work.

Nevertheless, it has been the occasion to demonstrate capability of focusing on the appropriate solutions regarding data science. By trying to apply the right concepts and tools while constructing features and the xgboost model; a functioning solution to my problem using AI was developed in a limited amount of time.

This work in a sense could picture a working environment reality. Data acquisition challenges and need for MLops is related to the role of a versatile AI developer, stretching further than strict the boundaries between data engineering, data science, and ML engineering. The courses I had taken under my AI developer education at ITHS, have with certitude prepared me to this field "decontenairization". The professional experience in data science acquired under my internships has comforted it.

Now, consequent work remains in bettering the model. Furthermore, the streaming pipeline will be built on the cloud as a first step, so that it can be constantly rolling and that the model can ultimately be used by an application. The development of a test application for commuter train user in Skåne is at the heart of the long-term development focus.

5 Acknowledgements

There are some individuals that I would like to adress sincere gratefullness to.

to Johannes Öhlin, my supervisor for both this work for and my internship at Tenfifty. Thank you for your constant availability for discussing concepts and for giving me a direction for this work every time I needed one. It was needed more than once!

to Kokchun Giang, head teacher and initiator at the AI program at ITHS. Thank you for your conseling and tips regarding just this work. Also, I am gratefull and humbled about how you managed to engage and educate your students over such a large AI stack with less than two years in your hands. It is your doing that I can call myself with confidence an AI developper.

Finally to Valentina Ivanova and Per Bröms at Rise for their kind support and availability to answer my questions and make the best of the KoDa API.

References

Abboud Ado, F. S. (2023). *Punktlighet på järnväg 2022.* https://www.trafa.se/globalassets/statistik/bantrafik/punktlighet-pa-jarnvag/2023/punktlighet-pa-jarnvag-2022.pdf. (Accessed on May 11, 2023)

Gomér, J. (2021). *Persontågsstrategi strategi för utveckling av den regionala tågtrafiken i skåne 2020–2040.* https://www.skane.se/namndshandlingar/2654903/. ("Accessed on 2023-05-25)

Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? In *Thirty-sixth conference on neural information processing systems datasets and benchmarks track.* Retrieved from https://openreview.net/forum?id=Fp7__phQszn

Jarrett, C. (2023, 02). *Categorical features in xgboost without manual encoding.* https://developer.nvidia.com/blog/categorical-features-in-xgboost-without-manual-encoding/. ("Accessed on 2023-05-27)

Nybogård, L. (2022, 8). *Skånetrafiken: Resandet tillbaka på samma nivåer som innan pandemin.* https://www.svt.se/nyheter/lokalt/skane/resandet-tillbaka-pa-samma-nivaer-som-innan-pandemin. ("Accessed on 2023-05-25)

Stenström, C., Famurewa, S., Aditya, P., & Galar, D. (2012, 09). Impact of cold climate on failures in railway infrastructure.. Retrieved from https://www.diva-portal.org/smash/get/diva2:1003719/FULLTEXT01.pdf

6 annexes

Guide on github for associating static and real-time gtfs using python: visit the page