From Black-Box to Explainable Insights: Forecasting Passenger Demand using APC Data

Fra Black-Box til indsigter:
Prædiktion af passagerer ved brug af APC data

Aarhus University

Department of Mathematics
BSc in Data Science

Authors: Andreas Hyldegaard Hansen, 202106123
Andreas Skiby Andersen, 202107332

Primary Supervisor: Bezirgen Veliyev, Associate Professor
Department of Economics

External Supervisor: Tobias Grønbæk Andersen, Data Analyst
Midttrafik

Submission Deadline: 13:00 June 15th 2024
Submitted: June 14th 2024

AARHUS UNIVERSITET

Abstract

Implementation of Automatic Passenger Counting (APC) systems in public transportation allows public transport providers to collect and analyze Big Data, which is crucial for modern demand planning and scheduling. APC data enables time series forecasting of future passenger demand. Danish research, regarding the analysis of bus passenger behavior and time series forecasting of bus passenger demand, is very limited, which highlights the importance of research.

The field of time series is rapidly expanding with "black-box" machine learning and neural network methods being favored above traditional linear methods. "Black-box" methods may increase accuracy, but limits transparency in the training and inference process, resulting in models with learning patterns that cannot fully be explained.
This project evaluates a Seasonal Naive model (SNaive), an autoregressive Lasso, Random Forest Regression (RF), XGBoost (XGB), Vanilla Reccurent Neural Network (RNN) and Long Short-Term Memory (LSTM) on Midttrafik's APC data from June 2021 to December 2023 in Aarhus, Denmark. Data is aggregated on 60-, 30- and 15-minute intervals.
We show that Fourier Transformations are efficient for analyzing seasonal patterns, and find that including daily lags for the past week is optimal for all non-differenced models. We also show that the Diebold-Mariano test is effective for testing the statistical significance of performance between forecasts from different models.

Our results show that XGBoost has the highest accuracy, between 85.4%-90.5% nMAE, with a 2 day forecast horizon on the test data, for 60-minute aggregation level, compared to 76.6%-84.8% nMAE for the benchmark.
While XGBoost is shown to be the most robust method for lower aggregation levels, we find that increasing the forecast horizon from 2 days to 2 weeks only gave up to around 26% relative increase in nMAE for Random Forest, XGBoost, Vanilla RNN and LSTM, and a 40% relative increase for the benchmark model.
A compelling aspect of the project is centered around explaining "Black-Box" models using Performance Based Shapley Values (PBSV); a new variant of SHAP values. For XGBoost, PBSV shows that including daily and weekly lags, information about peak hour and type of bus schedule has the highest importance, while weather variables have no significant importance.

Resumé

Implementering af Automatisk Passagertællings (APC) systemer i offentlig transport giver offentlige transportudbydere muligheden for at opsamle og analysere Big Data, hvilket er afgørende for moderne efterspørgselsplanlægning. APC-data åbner op for muligheden for at lave tidsserieprædiktioner af fremtidig passagerefterspørgsel. Dansk forskning, vedrørende analyse af mønstre blandt buspassagerer og tidsserie-prædiktioner af efterspørgslen blandt buspassagerer, er meget begrænset, hvilket understreger vigtigheden af ny forskning.

Feltet inden for tidsserier er i kraftig vækst, hvor brugen af "Black-Box" maskinlæring- og neurale netværk-metoder favoriseres over traditionelle lineære modeller. "Black-Box" metoder kan forøge præcisionen, men begrænser gennemsigtigheden under trænings- og inferens processen, hvilket resulterer i modeller med læringsmønstre, som ikke kan forklares fuldt ud.
Dette projekt evaluerer en Periodisk Naiv model (SNaive), en autoregressiv Lasso, Random Forest, XGBoost, Vanilla Recurrent Neural Network (RNN) og Long Short-Term Memory (LSTM) på Midttrafiks APC data fra juni 2021 til december 2023 i Aarhus, Danmark. Data er aggregeret til 60-, 30- og 15-minutters intervaller.
Vi viser, at Fourier Transformationer er effektive til at analysere periodiske mønstre og finder, at inklusionen af lags fra den tidligere uge er optimale for alle ikke-differencede modeller. Vi viser

også at Diebold-Mariano testen er effektiv til at teste den statistiske signifikans af prædiktionsevnen mellem forskellige modeller.

Vores resultater viser, at XGBoost har den højeste præcision, mellem 85.4%-90.5% nMAE, med en 2-dags prædiktions horisont på test data for 60-minutters aggregeringsniveau, sammenlignet med 76.6%-84.8% nMAE for benchmark modellen.
Mens XGBoost viser sig at være den mest robuste metode på lavere aggregeringsniveauer, så har vi fundet ud af, at en udvidelse af prædiktionshorisonten fra 2 dage til 2 uger kun giver op til omkring 26% relativ forøgelse i nMAE for Random Forest, XGBoost, Vanilla RNN og LSTM, og 40% relativ forøgelse i nMAE for benchmark modellen.
Et særligt interessant aspekt af projektet er centreret omkring at forklare "Black-Box" modeller, ved at bruge Performance Based Shapley Values (PBSV); en ny variant af SHAP-værdier. PBSV viser for XGBoost, at inkludering af daglige og ugentlige lags, information om myldretid og hvilken type busplan der bruges har størst betydning, mens vejrvariable ikke har en signifikant betydning.

1 Introduction

Public transportation offers a method of transportation, accessible to a majority of the population. It aims to be an effective transportation style, that offers daily commuters and other passengers, affordable and well timed trips, within and between cities. Public transportation is regarded as a method to relieve traffic congestion and pollution, which is crucial for a city's infrastructure and net-zero ambitions.

Danish public transportation is facing challenges with increasing costs, resulting in increased ticket prices and cutbacks in routes, departures and stops, which overall lowers the quality for passengers [1]. On the contrary, research has shown that expanding public transportation in urban areas decreases car ownership in the respective areas [2].

To overcome these challenges, collecting Big Data becomes essential, since it allows for data mining and analytics, which are used for tracking busses and optimizing routes and schedules. Automatic Passenger Counting (APC) allows public transportation to offer passenger demand forecasting as a service. It allows passengers to evaluate which departure time to choose in advance, given their personal preferences and needs e.g. convenience, privacy, social anxieties and disabilities.
Furthermore, it is valuable for public transportation planners to know when a route will become overcrowded, as well as to gain insight into correlated relationships between exogenous variables, such as weather and passenger demand.

When implementing forecasting models, it is important to choose a good aggregation level of the data and a good forecast horizon (number of days to forecast ahead). Whether the aggregation level is an interval of 15 minutes, an hour or a whole day determines the structure of the data. Choosing to forecast many days ahead gives public transportation planners a lot of time to plan ahead, but the accuracy of the model will significantly decrease for longer forecast horizons.
Several studies have looked at the effect of different features, especially the impact of weather, as severe rain and wind can affect the amount of cyclists and pedestrians [3][4], and lags, as public transportation can be heavily influenced by seasonal patterns, such as daily commuters [5].

Many different types of forecasting models have been developed throughout the years, ranging from classic statistical models to modern machine learning and deep learning models. It is crucial to find models with optimal features, and be able to compare them fairly, to find the model that

strikes the perfect balance between accuracy, complexity and estimation/inference time. To find optimal lags, patterns can be searched for using the Fast Fourier Transform algorithm (FFT), while a statistical comparison of model performance can be performed using the Diebold-Mariano test.

Highly advanced models can offer better accuracy for regression and classification tasks, but comes at the trade-off of reduced interpretability in the training process and reduced explainability of predictions in the inference process. Explainable AI aims to make the decision making transparent. It can be divided into two types of methods. The first type is model specific methods, which are tailored to explain the specific structure of the model, whereas the second type, model agnostic methods, can be applied to any model, regardless of model structure. Agnostic methods offer flexibility and fair comparison of different models, where specific methods may offer deeper insight [6].

SHapley Additive exPlanations (SHAP) is a framework for evaluating feature contribution, with the purpose of discovering patterns learned by uninterpretable "black-box" models. Performance Based Shapley Values (PBSV) is a newly proposed extension of SHAP, that measures the feature contribution to the loss, rather than the contribution to the raw predictions [7].

In this project, we focus on applying and evaluating statistical models, machine learning models and neural networks for forecasting bus passenger demand accurately for the next 2 days ahead in Aarhus, Denmark. Notably, we apply PBSV for evaluating feature importance and contribution. We will go in depth with the following:

1. Exploring the structure of passenger data, by identifying exogenous variables correlated with passenger demand and searching for patterns.
2. Compare the performance of Lasso, Random Forest, XGBoost, Vanilla RNN and LSTM for forecasting bus passenger demands, against a Seasonal Naive benchmark.
3. Evaluate Performance Based SHAP Values to gain important insight into patterns and feature importance learned by "black-box" models.
4. Providing results for Danish passenger forecasting, which can be used as references for future implementations and research.

2 Literature review

Academic literature has studied the field of forecasting public transportation ridership throughout many years. Over the years a wide array of forecasting models have emerged, starting from traditional linear models, such as OLS, Lasso, ARIMA and SARIMA, and transitioning to advanced machine learning methods, such as Support Vector Regression (SVR), K-Nearest Neighbor Regression (KNN), Decision Trees (RF/XGB) and Recurrent Neural Networks (RNN/GRU/LSTM); see table 1 for an overview.

The literature agrees that short term passenger flow forecasting is a highly nonlinear, nonstationary time series problem. The literature agrees that forecasting models can be significantly improved by including seasonal components and exogenous variables. In recent years, time series forecasting has increasingly shifted towards the use of machine learning and neural networks, especially LSTM, due to the ability to model nonlinear relationships. There is generally a scarcity of Danish litterature on the analysis of bus passenger behavior and time series forecasting of bus passenger demand.

Passenger demand is an integer-valued number, thereby it is possible to use integer-valued forecasting methods. The literature does not mention using this approach for forecasting the passenger demand, but instead, they use real-valued forecasting methods and round the output off to the nearest integer [8]. Consequently, we will also focus on real-valued forecasting, which will be rounded off to the nearest integer.

Weather conditions are often assumed to influence public transportation passenger demand. Adverse weather can have a negative impact on the use of public transportation, in situations where the weather will worsen the quality of the travel. Adverse weather might cause travelers to stay at home or it might cause partial or complete cancellations. Wei (2022) found that weather conditions did not affect adult passenger flow during morning peak hours, observed on 30-minute intervals. For off-peak hours, rain and wind was found to significantly decrease adult passenger flow in Brisbane, Australia, observed on 30-minute intervals. Wei argues that in morning peak hours, travelers need to travel to work and school, where afternoon leisure travelers, that travel for entertainment and shopping, are more flexible [3].
Some methods of transportation are more susceptible to adverse weather e.g. walking and cycling. Adverse weather can also affect weather-susceptible commuters, to instead favor public transportation. Farahmand, et. al. (2023) observed that rainfall and strong wind increased passenger flow at morning and afternoon peak hours, on weekdays in the Netherlands. This caused the prediction accuracy to decrease, as all models underestimated passenger flow on days with heavy rain and strong wind [4].
The direction and magnitude of how adverse weather influences public transportation varies depending on where the study is conducted, although the literature consistently agrees that weather conditions influences individuals' decision to change travel plans [9].

Forecasting passenger demand using the demand in previous time steps is important to consider, due to the nature of passenger behavior. Lagged variables gives forecasts the ability to take seasonal effects into account, which can stem from repeated travel patterns, e.g. many commuters on workdays and few commuters on weekends. Cheng et. al (2022) concludes that including lagged passenger demands, of up to seven days, significantly improved the forecasting ability of their proposed model [5].

Farahmand et al. (2023) used smart-card data from Twente, Netherlands, and applied 3 different MLP models. One without weather information, one with same time weather information and one with one hour ahead weather forecasts in combination with holiday and football parameters. They showed that including weather and holiday information significantly improved forecasting, although their model still showed high errors in morning peak hours with heavy rain and strong wind. They also showed that the football schedule had a low impact on forecast performance [4].

Tang et al. (2020) applies Gradient-Boosted Decision Trees (GBDT) to the bus network in Changsha, China, to be able to understand the origin-destination and travel patterns of passengers. They utilize the properties of the GBDT architecture to measure the relative importance of their variables, using the frequency of the features used across all trees [9].

Cheng et al. (2022) applied MLP, SVM, RBFN and LSTM to forecast passenger flow for three bus routes in Taiwan, from January 2018 to October 2019. They calculated 80 models, and selected a weighted model of the top three best performing models, which were three lagged LSTM models [5].

Monje et al. (2022) used an XAI approach (AI + exogenous variables) to forecast bus passenger numbers in Madrid. They used an architecture consisting of LSTM blocks, and evaluated the

model's performance against different machine learning models. They argued that adding exogenous variables to the otherwise completely "black-box" model, increases the interpretability of the model. Furthermore, they use surrogate rules as their method to interpret feature contribution [10].

Liyanage et al. (2022) applied LSTM and bi-LSTM on Smart Card data in Melbourne, Australia, and compared with five conventional neural network methods. They tested 15-, 30- and 60-minute forecast horizons and showed that bi-LSTM achieved an accuracy above 80% and outperformed all models [11].

Halyal et al. (2022) compared an LSTM model against a traditional SARIMA model, for forecasting public transit passengers in Dharwad, India. They found that deep-learning based models generally outperform traditional linear models in describing the stochastic and non-linear nature of passenger demand, but in return requires larger datasets and more computing time. They test the models on 15-, 30-, 45- and 60-minute intervals and found that LSTM on 60-minute aggregation level had the best performance [12].

Yan & Zhou (2023) applied a Bayesian Optimized XGBoost model on bus passenger flow in Guangzhou, China, and compared against SVM, KNN and Random Forest models. To interpret the contribution of a variable, on the output of their model, they applied SHapley Additive exPlanations (SHAP). From this, they found that temperature and wind has a large impact on forecasts. BO-XGBoost outperformed all models, but had longer computing time [13].

Jiao et al. (2021) proposed an improved hybrid STL-LSTM model, which is a combined model of seasonal trend decomposition by loess regression and LSTM, to improve bus forecasting abilities during COVID-19 in Beijing, China. They used passenger data from November 2019 to August 2020, and achieved an accuracy increase of 15% [14].

Ye et al. (2019) applied ARIMA models to predict passenger flow in Jiaozou, China, from January 2018 to March 2018. They showed that their ARIMA model had difficulties with the volatility and non-stationarity of passenger data, and argued that including seasonality, trend and weather variables, could have improved results [15].

This project will build upon the research conducted by Liyanage et al. (2022), Yan & Zhou (2023) and Farahmand et al. (2023). Based on the selected papers, the models that will be adopted is Random Forest Regression, XGBoost, Vanilla RNN, LSTM and an autoregressive Lasso with regularization rather than SARIMA.
Liyanage et al. (2022) propose that future research should look into the impact of weather and Yan & Zhou (2023) emphasised the idea of using SHAP values for interpreting feature importance. We decide to base our variables on the research of Farahmand et al. (2023) due to their emphasis on use of weather and the Neatherlands is the region in our literature review that's most similar to Denmark.
We will assess the applicability of the proposed models in Aarhus, and compare feature importance and accuracy.

Author, Year	Area	Application	Methods	Best	KPI's
Cheng et al. (2022)	Taiwan	Bus passenger forecasting	MLP, SVM, RBFN, LSTM	3 weighted LSTM's	RMSE, MAPE
Monje et al. (2022)	Madrid, Spain	Bus passenger forecasting	LSTM, SVM, XGBoost, Random Forest	LSTM	R^2
Farahmand et al. (2023)	Twente, Nether-lands	Bus passenger forecasting	MLP	MLP	MAE, EVS
Liyanage et al. (2022)	Melbourne, Australia	Bus passenger forecasting	bi-LSTM, LSTM, RNN, DLBP, MNN, RBF, GRNN	bi-LSTM	MAPE, MAE, RMSE
Halyal et al. (2022)	Dharwad, India	Public transit forecasting	LSTM, SARIMA	LSTM	RMSE, MAE
Yan M., Zhou K. (2023)	Guangzhou, China	Bus passenger forecasting	BO-XGBoost, KNN, SVM, Random Forest	BO-XGBoost	RMSE, MAPE, VAPE, Computing Time
Jiao et al. (2021)	Beijing, China	Bus passenger forecasting during COVID-19	ISTL-LSTM, STL-LSTM, LSTM, GRU, LinReg, KNN, XGBoost	ISTL-LSTM	MAE, MAPE, RMSE, VAPE
Ye et al. (2019)	Jiaozou, China	Bus passenger forecasting	ARIMA	ARIMA	SDE, MAD, MAPE

Table 1: Overview of papers from previous authors. Area of implementation, application context, different methods used, the best method and Key Performance Indicators (KPI's) are listed for each paper.

3 Data processing

Midttrafik is the regional public transportation administration in central Jutland. They have invested in equipping their busses with GPS and optical cameras, for Automatic Passenger Counting (APC), for two main reasons. They want to collect analytics concerning timeliness and passenger distributions. Analytics is used to adapt future routes and driving schedules, to improve the service offering within the given budget. Furthermore, they offer a real time tracking service to passengers. Passengers can, through an app, see the current location, delay and available capacity of most busses that are currently en route.

In this project, we focus on the A-bus routes in Aarhus (route 1A to 6A), which are the most in-demand intracity routes in Aarhus. The project is focused on developing and testing forecast methods, and not on developing a software system for forecasting all bus passenger demands in Aarhus municipality. Therefore, we limit ourselves to only a few, but high-demand bus routes,

since low-demand bus routes are usually not subject to the problem of being overcrowded. A visualization of the six bus routes can be seen in figure 1, while a heatmap showing the most congested routes and time periods can be seen in figure 2.

Figure 1: Route map of routes 1A to 6A in Aarhus, Denmark.

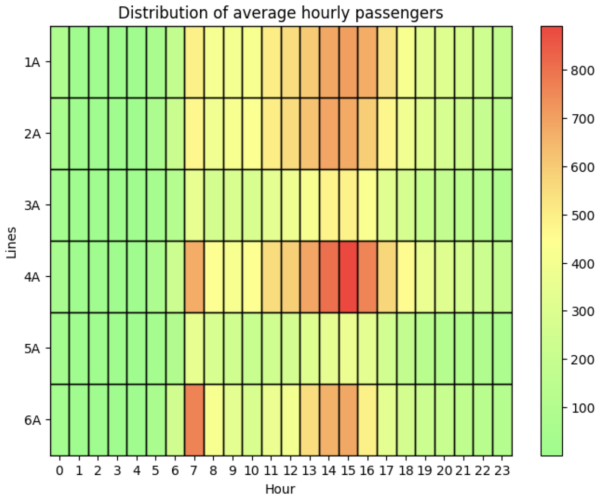

Figure 2: Heatmap of average hourly passengers on bus 1A-6A.

Midttrafik's APC database has an entry for each time a passenger has boarded or disembarked a bus with date, time, location and journey information. The method that Midttrafik uses to count the number of passengers, is by exclusively counting the number of passengers that board the bus, and then aggregating that number to different aggregation levels for interpretation.

We have aggregated the APC data to 15-, 30- and 60-minute intervals for each bus route, from the 27th of June 2021 (beginning of 2021 bus schedule) to the 31st of December 2023. Earlier data is heavily influenced by COVID-19, which drastically changed the travel behaviour of many individuals.
As APC data is collected automatically, there will be errors in the data. Errors may originate from occlusion, if a lot of passengers are standing near the camera's field of view, from bright sunshine's reflection in the windows or from the lack of light if a light bulb is defective. These scenarios do occur once in a while, but data quality is not a major concern, as the daily counting precision in this timeframe is tested to be above 99% on average for routes 1A-6A.

Midttrafik also provides calendar data with categorical information about date, year, quarter, month, week, weekday, vacation periods, and which bus schedule is used on the particular day. Bus schedules are divided into Sunday/holiday, which covers all Sundays and national holidays, Saturday, which covers all Saturdays, and workday, which covers all Mondays to Fridays except national holidays.
Weather data from Aarhus Municipality is obtained from Denmarks Meterological Institute's (DMI) public available Climate Data API. Historic climate data is quality controlled by meterologists, and interpolated for missing and erroneous observations [16]. DMI also supplies the Forecast API with 48 hour weather forecasts, which we have not used, since historical forecast data is not available, but it can be used in a practical implementation.
We include a boolean event variable, which is 1 if the hour is related to one or more special events in Aarhus. We chose the time interval two hours prior and two hours past the event to be included in the event variable, e.g. if an event is from 19:00 to 21:00, then hours 17-23 have a value of 1. We assume that public transportation is mainly affected within this time frame.

An overview of the columns of the full dataset, which is used throughout the report, can be seen in table 2.

Date	Observation date in local Europe/Copenhagen time zone.
Hour‡	Interval 05:00 (morning) - 00:59 (next day)
Minute	NA for 60-minute aggregation, 0 and 30 for 30-minute aggregation and 0, 15, 30, 45 for 15-minute aggregation.
Passengers Boarding	Number of boardings counted by APC.
Peak Hour	Boolean value of 1 in intervals 07:00 - 10:59 and 13:00 - 17:59 as defined by "Rejsekort". Travelling outside peak hour gives a 20% Rejsekort discount.
Quarters of the year	One-hot-encoded as Q1, Q2, Q3, Q4.
Month-, Date- and Weeknumber	Within the intervals 1-12, 1-31, 1-52 respectively.
Weekdays	One-hot-encoded as Mon, Tue, ..., Sun.
Plantype	Bus schedule type one-hot-encoded as workday, Saturday and Sunday/holiday.
Vacation periods	Public school vacations one-hot-encoded as summer, fall, Christmas, winter and Easter.
Accumulated precipitation	Accumulated precipitation in mm per hour.
Bright sunshine	Minutes of bright sunshine per hour.
Mean cloud cover	Cloud cover in % per hour.
Mean pressure	Mean pressure in hPa per hour.
Mean relative humidity	Mean relative humidity in % per hour.
Mean temperature	Mean temperature in Celsius per hour.
Mean wind speed	Mean wind speed in m/s per hour.
Snow depth†	Mean snow depth in cm per day.
Special event*	Binary variable with value of 1 if the hour contains a special event, else 0.

Table 2: Overview of columns in the dataset, consisting of passenger- and calendar data from Midttrafik, weather data from DMI and event data from the webpages of the different events.
‡ Hours 1, 2, 3 and 4 are discarded.
† Snow depth is manually measured once a day.
* Included events are; AGF Men Football matches, AGF Women Football matches, Grøn koncert - Aarhus (Music Festival), SPOT Festival (Music Festival), Northside (Music Festival), Aarhus Pride (Parade), Kapsejladsen (Regatta), Aarhus Food Festival, Royal Run - Aarhus (Running event), Aarhus Festuge (Festival week), Tivoli Friheden - Fed Fredag (Concert).

APC data is in local/Copenhagen time and DMI data is in Universal Time Coordinated (UTC) time. DMI data is converted to local/Copenhagen time, then APC and DMI data is joined onto the calendar data. Daylight savings time entails that one day per year has 25 hours and one day per year has 23 hours. We standardize each date to 24 hours by removing the 25th hour and linearly interpolate data from the missing 24th hour.
We discard hours 1, 2, 3 and 4, since our explorative analysis shows that these hours have an average passenger count of 0. It stems from the fact that these hours are not standard operating hours for route 1A-6A.

We have performed an 80%/10%/10% training, validation and test split of the data, and the test data has strictly not been included in any analysis before the final proposed models were to be applied.

4 Methodology

4.1 Process description

Our project flow is shown in figure 3.
First, data is collected from different sources, and processed to standardize the data into a suitable format, as described in section 3.
Next, we carried out an exploratory analysis, with the purpose of gaining familiarity with the data, exploring correlations between bus passengers and exogenous variables and mining potential patterns. We analyzed the data for seasonal patterns, trend patterns and checked stationarity.
We then entered the model training cycle. We used the validation data to optimize our models over multiple different combinations of exogenous variables, lags and hyperparameters. We checked the consistency of our models across all bus lines with the purpose of analysing the generalizability of their predictive power.
At last, we tested the predictive power of the proposed models on the test data and explained the results using SHAP values to gain an understanding of what patterns the "black-box" models had learned.

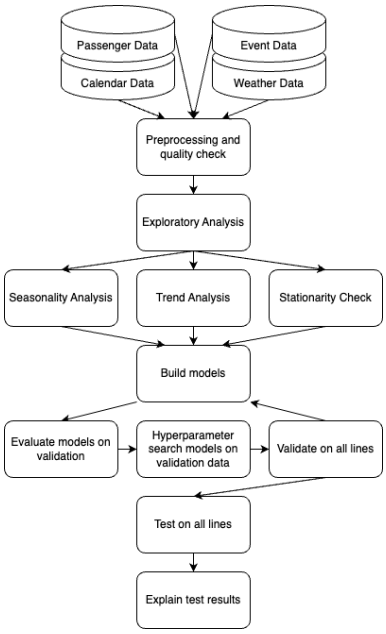

Figure 3: Overall flow of data processing, analysis, modelling and evaluation and explanation.

4.2 Recursive forecasting

$$y_{t+h} = \hat{y}_{t+h} + \epsilon_{t+h} \tag{1}$$
$$\hat{y}_{t+h} = f(X_{t+h}) \tag{2}$$

In time series forecasting, the purpose is to forecast future values of the target variable y_t. Forecasting h time steps into the future means that the ground truth value y_{t+h} is predicted as $\hat{y}_{t+h}$ and any forecast will always have an error ϵ_{t+h}. Predictions are calculated by a regression model $f(X_{t+h})$ which is any model f that takes an input X_{t+h} and produces an output $\hat{y}_{t+h}$.
X_{t+h} is the input vector at time step $t+h$, which contains the features used for regression, including exogenous binary variables, exogenous continuous variables and lagged values of the target.

$$E[\epsilon_{t+h}|X_{t+h}, X_{t+h-1}, X_{t+h-2}, ...] = 0 \Rightarrow E[\hat{y}_{t+h}] = y_{t+h} \tag{3}$$

Linear models build upon the zero conditional mean assumption (3), which states that the errors are expected to be centered around zero and uncorrelated with any explanatory variable. Machine Learning and Deep Learning models do not strictly depend on this assumption, yet zero-centered residuals are still desired to prevent bias and ensure consistency of predictions [17].

Recursive forecasting is used to handle forecasting with lagged values. Forecasting h time steps into the future using a lag l, the nature of the dependency on the lagged value changes based on whether the lag l is greater than the horizon h.
If $l > h$, then the prediction $\hat{y}_{t+h}$ depends on the ground truth lagged value y_{t+h-l}. Concretely,

X_{t+h} in equation (2) will contain the lag value y_{t+h-l}.
If $l < h$, then a recursive forecast method is necessary, as the prediction $\hat{y}_{t+h}$ depends on the lagged prediction $\hat{y}_{t+h-l}$. Concretely, X_{t+h} in equation (2) will contain the predicted lag value $\hat{y}_{t+h-l}$, instead of the ground truth lagged value y_{t+h-l}.

The re-estimation of models can use three different window approaches. The first approach is a fixed window approach, where the model is estimated on the training data once and predictions are made recursively without re-estimation. The second approach is an expanding window approach, similar to fixed window, but the model is re-estimated after each h predicted time steps into the future, which yields an expanding size of the training data. The third and last approach is a rolling window approach, similar to expanding window, except that the size of the training data remains constant and thereby creates a rolling effect, and is again re-estimated after each h predicted time steps into the future.

4.3 Discrete Fourier Transformation (DFT)

The Fourier Transformation is a widely used tool, often used to analyze the frequency representation of a signal and for filter design.

$$Y_f = \frac{1}{T} \sum_{n=0}^{T-1} y_t e^{-2\pi j f t} \tag{4}$$

The discrete Fourier Transform analysis equation is show in (4). It transforms a discrete signal in the time domain y_t into its representation in the frequency domain Y_f. If the time domain signal has T observations, then DFT operates by describing the signal as a weighted sum of T complex sinusoids. The magnitude of the frequency representation $|Y_f|$ is used to measure how strong the sinusoidal oscillation, with frequency f, is represented in the time domain data. The larger the magnitude is, the more dominant the sinusoid is in the time domain data. The mean of the time domain signal is denoted as the DC offset, and is represented as frequency 0 in the Fourier Transform.
In practice, the Discrete Fourier Transform is efficiently computed by the Fast Fourier Transform algorithm (FFT) [18].

4.4 Stationarity

A strictly stationary time series $\{y_t\}_{t=1}^{t=T}$ is a time series whose joint probability distribution of $(y_{t_1}, y_{t_2}, ..., y_{t_r})$ is time invariant for $\forall r \in \mathbb{N}$. All individual random variables have the same marginal distribution $\forall t \in T : y_t \sim F_Y$, which is independent of time. It is often hard to ensure strict stationarity as the probabilistic distribution of the real data is often unknown.
Covariance stationarity is a weaker form of stationarity, which does not assume that the distribution is time invariant, but that the mean, variance and autocorrelation is time invariant [19].
A time series is covariance stationary if

- Mean is real valued and does not depend on time $\forall t \in T : E[y_t] = \mu < \infty$
- Variance is real valued and does not depend on time $\forall t \in T : Var(y_t) = \sigma^2 < \infty$
- Covariance between y_t and lagged value y_{t-l} depends only on l and is independent of time $\forall t \in T : Cov(y_t, y_{t-l}) = \gamma_l < \infty$

4.4.1 Augmented Dickey Fuller test (ADF)

The ADF test builds upon the Dickey Fuller test (DF), that tests for a unit root.

With the DF test, we consider the following AR(1) regression model

$$y_t = \beta_0 + \beta_1 y_{t-1} + \epsilon_t \tag{5}$$

In equation (5), the DF test tests the null hypothesis $H_0 : \beta_1 = 1$ (stochastic) against the alternative hypothesis $H_A : |\beta_1| < 1$ (stationary). Under the null, y_t is equal to a sum of error terms, which makes it a unit root.

Both the DF test and ADF test uses the OLS t-statistic as test statistic, which is written as

$$t = \frac{\bar{y} - \mu_{y,0}}{SE(y)} \tag{6}$$

Equation (6) shows that the t-statistic is the standardized sample average, consisting of the difference between the sample mean and the hypothesized population mean, divided by the standard error [17].

To be able to use the t-statistic, the null hypothesis should be that the variable is equal to 0. Hence, y_{t-1} is subtracted on both sides in equation (5), to be able to rewrite it as

$$\begin{aligned} y_t - y_{t-1} &= \beta_0 + (\beta_1 - 1) y_{t-1} + \epsilon_t \\ &\Updownarrow \\ \Delta y_t &= \beta_0 + \delta y_{t-1} + \epsilon_t \end{aligned} \tag{7}$$

Where $\delta = \beta_1 - 1$ and Δy_t is y_t first-differenced, e.g. $\Delta y_t = y_t - y_{t-1}$.

In equation (7), the DF test tests the previous mentioned hypotheses $H_0 : \beta_1 = 1$ and $H_A : |\beta_1| < 1$, by testing the null hypothesis $H_0 : \delta = 0$ (stochastic) against the alternative hypothesis $H_A : \delta < 0$ (stationary).

The ADF test expands on this concept from the DF test and instead tests for a unit autoregressive root. Consider the following AR(p) regression model with trend term at

$$\Delta y_t = \beta_0 + at + \delta y_{t-1} + \gamma_1 \Delta y_{t-1} + \gamma_2 \Delta y_{t-2} + \ldots + \gamma_p \Delta y_{t-p} + \epsilon_t \tag{8}$$

In equation (8), the ADF test tests the null hypothesis $H_0 : \delta = 0$ (stochastic) against the alternative hypothesis $H_A : \delta < 0$ (stationarity). Under the null, y_t is equal to a sum of error terms with the trend term at and a sum of previous unit autoregressive roots $\gamma_1 \Delta y_{t-1} + \gamma_2 \Delta y_{t-2} + \ldots + \gamma_p \Delta y_{t-p}$, which makes it a unit autoregressive root [17].

4.4.2 Kwiatkowski-Phillips-Schmidt-Shin test (KPSS)

The KPSS test is used to test the null hypothesis that a time series is stationary around a deterministic trend, which can be both a constant mean or a linear trend.

$$y_t = \xi t + r_t + \epsilon_t \tag{9}$$

Time series $\{y_t\}_{t=1}^{T}$ is broken up into a linear trend, stochastic trend and error ϵ_t, which is assumed to be stationary. r_t is a random walk, where $r_t = r_{t-1} + u_t$, r_0 is a fixed intercept and u_t are i.i.d. with zero mean and variance σ_u^2.

The null hypothesis is $H_0 : \sigma_u^2 = 0$. As ϵ_t is assumed to be stationary, then y_t is stationary around the deterministic trend ξt under the null hypothesis H_0 [20].

The KPSS test uses the Lagrange Multiplier test as test statistic.

$$LM = \sum_{t=1}^{T} \frac{S_t^2}{\hat{\sigma}_\epsilon^2} \tag{10}$$

Equation (10) shows the test statistic, where $S_t = \sum_{i=1}^{t} \epsilon_i$ is the partial sum of residuals and $\hat{\sigma}_\epsilon = \frac{1}{T}\sum_{t=1}^{T} \epsilon_t^2$ is the estimated error variance. Residuals are defined as $\epsilon_t = y_t - \hat{y}_t$ for $t = 1, ..., T$ [20].

4.5 Models

4.5.1 Seasonal Naive

$$\hat{y}_t = y_{t-140} \tag{11}$$

Equation (11) shows our seasonal naive model, where the predicted value has the same value as the same hour last week. Note that 140 hours corresponds to the same hour 7 days ago, as each day is 20 hours long after discarding hour 1 to 4 as described in section 3.

4.5.2 Lasso

$$\hat{y}_t = w_0 + \sum_{i=1}^{k} w_i X_{t,i} \tag{12}$$

Equation (12) shows a standard linear regression, where $X_{t,i}$ is the regression features and w_i is regression coefficients. If one or more features are lagged values of the target y_t, then the linear regression becomes an autoregressive AR(p) model.
Least Absolute Shrinkage and Selection Operator (Lasso) is an estimator that utilizes L1 regularization for coefficient shrinkage and variable selection. The loss function of Lasso extends sum of squared errors from OLS with L1 regularization.

$$w_{Lasso} = arg\min_{w} \left(\sum_{t=1}^{T} (\hat{y}_t - y_t)^2 + \lambda \sum_{i=1}^{p} |w_i| \right) \tag{13}$$

The coefficients of Lasso are found by minimizing the loss function (13), where $\lambda > 0$ is a hyperparameter for L1 regularization. Larger values of λ gives larger penalties to coefficients, and thereby shrinks the coefficients closer towards 0, where coefficients also can be set to 0.
(13) has no general closed form solution, as it involves absolute values of the regression coefficients. The loss function is instead minimized by a numerical optimization algorithm. In our case, this will be Coordinate Descent [21].

4.5.3 Random Forest Regression

Random Forest is a supervised ensemble learning method, which builds multiple weak decision trees and combines the outputs to create a strong prediction.

The input data is sampled with bootstrapping, which means that data is drawn at random with replacement and forms bootstrapped samples. Each tree is build by a bootstrapped sample, which ensures that each tree is build independently from all other trees, and hence the outputs are independent.

The output is an example of Bootstrap Aggregation (Bagging), where all outputs are averaged to create a strong prediction with lower variance than each individual tree.
Ordinary bagging considers all features when evaluating splits, which can result in a high correlation between trees. Random Forest aims to increase tree diversity and tackles this by feature bagging, where only a random subset of features are considered for a split. The process of using feature bagging, ensures high diversity among trees and low correlation between trees. It ensures high accuracy and helps prevent performance instability, as each tree only sees a random sample of the input.

Random Forest is considered a black box model, as it is difficult to interpret which factors cause what behaviour in the predictions.

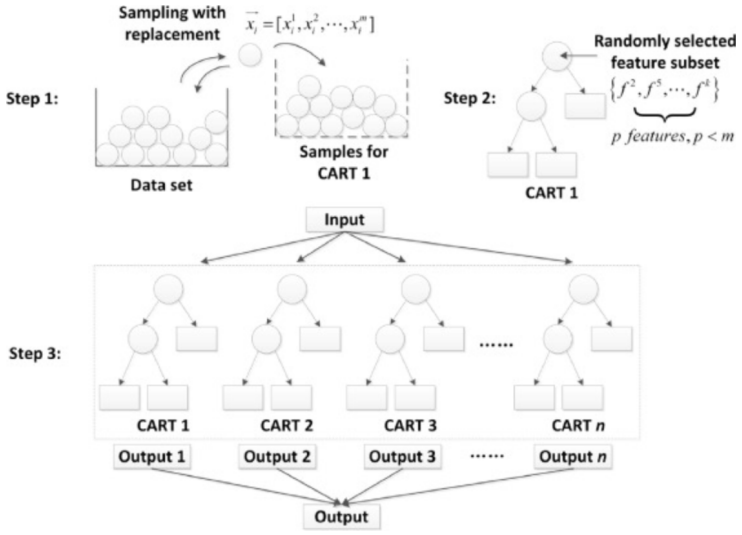

Figure 4: Methodology for Random Forest. Illustration obtained from Lin et. al. 2017 [22]

Figure 4 shows the process of training Random Forest.
In the first step, the input data is bootstrapped to obtain K bootstrapped samples $X_t^{(k)}$ for $k = 1,...,K$, where all samples contains all m input features.
In the second step, a decision tree $\hat{h}_k$ is built, using the random subspace technique, feature bagging. At each node, a set of p input features are randomly picked where $p < m$, and the best split is chosen among the p features. A split is defined as the i'th feature, having a value above a threshold $X_{t,i}^{(k)} > q$ or below a threshold $X_{t,i}^{(k)} < q$. This step is repeated iteratively, until no splits further improve the tree, and is repeated for all trees.
In the third step, aggregation makes the final prediction, and is here produced by an unweighted average of all individual predictions $\hat{y}_t = \frac{1}{K}\sum_{k=1}^{K}\hat{h}_k(X_t^{(k)})$. The Law of Large Numbers ensures that the output converges towards the ground truth, as the number of trees increases [23].

4.5.4 XGBoost

Extreme Gradient Boosting (XGBoost) is a supervised gradient tree boosting method proposed by Chen & Guestrin (2016). It builds multiple decision trees, similar to Random Forest, but differs in how the trees are built and combined. XGBoost is not based on bootstrapping data. Instead trees are built sequentially in a boosting fashion, where each tree has the objective of correcting the errors of the previous tree.

As XGBoost is a decision tree, it can be good at modelling nonlinearity, which often yields high accuracy predictions. XGBoost can use of L1 or L2 regularization, which helps prevent overfitting and improve generalizability.

$$\begin{aligned}
\hat{y}_t^{(0)} &= 0 \\
\hat{y}_t^{(1)} &= f_1(X_t) = \hat{y}_t^{(0)} + f_1(X_t) \\
\hat{y}_t^{(2)} &= f_1(X_t) + f_2(X_t) = \hat{y}_t^{(1)} + f_2(X_t) \\
&\dots \\
\hat{y}_t^{(K)} &= \sum_{k=1}^{K} f_k(X_t) = \hat{y}_t^{(K-1)} + f_K(X_t)
\end{aligned} \tag{14}$$

Equation (14) shows the principle of additive learning (boosting), that starts with a weak initial tree estimator, where all subsequent trees then have to correct the errors performed by the previous tree. At the final iteration, K, the final prediction $\hat{y}_t^{(K)}$ of the target variables is a sum of all tree outputs. f_k is the k'th tree and $f_k(X_t)$ is the output of the k'th tree given input variables X_t.

$$L^{(j)} = \sum_{t=1}^{n} l(y_t, \hat{y}_t^{(j)}) + \sum_{k=1}^{j} \Omega(f_k) = \sum_{t=1}^{n} l(y_t, \hat{y}_t^{(j-1)} + f_j(X_t)) + \Omega(f_j) \tag{15}$$

$$\Omega(f) = \gamma T + \frac{\lambda}{2} \sum_{j=1}^{T} w_j^2 \tag{16}$$

XGBoost is a boosting method, which means that after the $j-1$'th iteration, the j'th tree is chosen as the tree f_j that minimizes the loss in equation (15). l is the loss function and $\Omega(f)$ in equation (16) is regularization for tree f. T is the number of leaves in the tree and w is a vector of leaf scores. Penalizing the number of leaves and leaf scores in a tree contributes to preventing too complex trees being built.

$$L_{split} = \frac{1}{2} \left(\frac{G_{left}^2}{H_{left} + \lambda} + \frac{G_{right}^2}{H_{right} + \lambda} - \frac{(G_{left} + G_{right})^2}{H_{left} + H_{right} + \lambda} \right) - \gamma \tag{17}$$

Building the decision trees requires a method for evaluating which splits to perform. It is impossible to iterate over all possible trees and pick the best. Therefore each tree is optimized one step at a time, by evaluating the loss reduction of a split. It uses an approximate greedy split finding algorithm, to determine the loss reduction of a split candidate in equation (17), where the score is the sum of the first order loss gradient squared G^2 divided by the sum of the second order loss gradient H and regularization λ. γ is the minimum loss reduction of a split. If the loss reduction is less than γ, then there is no reason to add a new branch. This regularization technique is known as tree pruning [24].

4.5.5 Vanilla RNN

The vanilla Recurrent Neural Network (RNN) is a type of Neural Network, which implements loops that persist information and can act as memory.

At the first time-step $t = 1$, the RNN takes information X_1 and outputs the hidden state H_1 to the output layer, as well as sending H_1 back to itself again.
At the next time-steps $t > 1$, the RNN takes information X_t and the previous time-step's hidden state H_{t-1} and again outputs the hidden state H_t to the output layer and sends H_t back to itself. This can also be seen in figure 5.

$$H_t = \tanh(W_{xh}X_t + W_{hh}H_{t-1} + b_h) \qquad (18)$$

In equation (18), it is shown how the hidden state is computed at each time-step t, where W represents the weights of the input and the hidden state, and where b represents the bias.

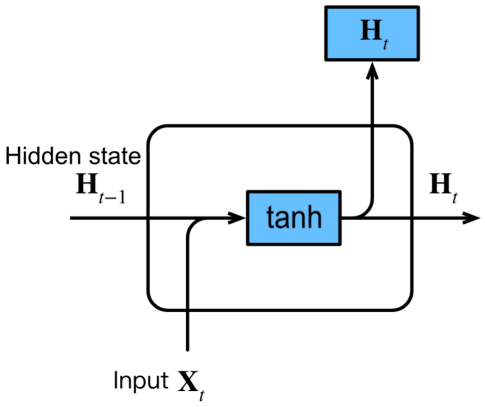

Figure 5: Structure of RNN module.

In equation (18), it is also shown that the model uses the nonlinear activation function, hyperbolic tangent, as activation function, which introduces nonlinearities into the model. Without nonlinearities the models are limited to represent linear transformations of the input data, which might be insufficient for capturing the complexities present in real-world data. This might then be considered an advantage of Neural Networks as opposed to previous introduced models [25].

The vanilla RNN can be trained using the Gradient Descent algorithm, which for Recurrent Neural Networks utilizes backpropagation through time (BPTT), to calculate the gradient of the loss function, with respect to the weights of the network [26].

4.5.6 LSTM

While vanilla RNN's theoretically can persist long term information through the implementation of loops, it is in practice better at extracting recent information, and performs worse when trying to extract long-term information.

Long Short-Term Memory (LSTM) is a different type of recurrent neural network, that is an enhancement of the vanilla RNN. As opposed to the vanilla RNN, the LSTM module persists information not only via the hidden state, but also includes the implementation of a memory cell.

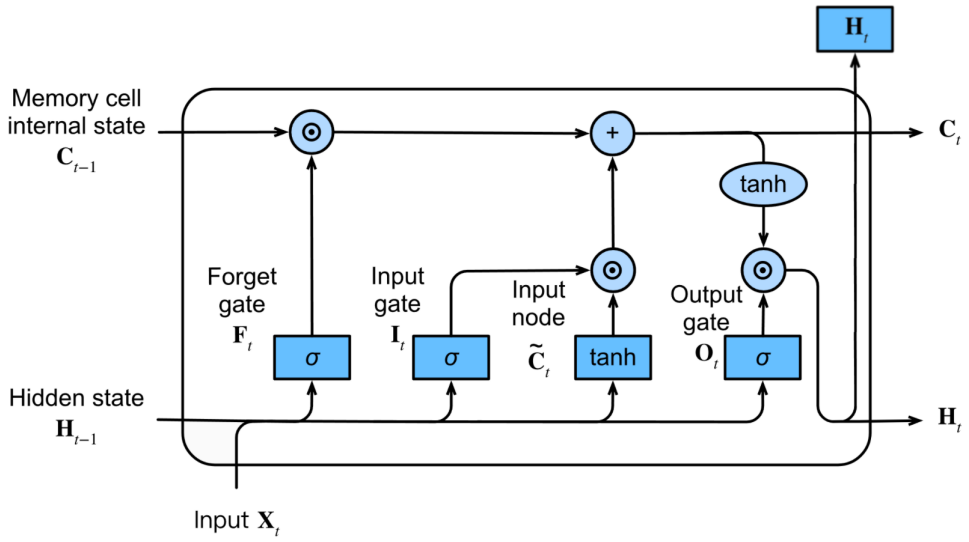

Figure 6: Structure of LSTM module.

At the first time-step $t = 1$, the LSTM takes information X_t as input and creates the memory cell internal state C_t, to be used to compute the hidden state H_t. It then outputs the hidden state H_t to the output layer and sends H_t and C_t back to itself again.
At the next time-steps $t > 1$, the LSTM takes information X_t, the previous time-step's hidden state H_{t-1} and the previous time-steps memory cell internal state C_{t-1} as input. First it updates the memory cell internal state to C_t, to again be able to compute the new hidden state H_t. Afterwards, it outputs the hidden state H_t to the output layer, and again sends the H_t and C_t back to itself.

To be able to integrate and update the memory cell internal state in the module, the extension from vanilla RNN into LSTM has introduced gates to the module structure, which can be seen in figure 6.

$$\tilde{C}_t = \tanh(W_{xc}X_t + W_{hc}H_{t-1} + b_c) \tag{19}$$

In equation (19), it is shown how the input node $\tilde{C}_t$ is calculated, which corresponds to the vanilla RNN's method of computing the hidden state in equation (18).

In equation (20), it is shown how the different gates are computed.

$$\begin{aligned} I_t &= \sigma(W_{xi}X_t + W_{hi}H_{t-1} + b_i) \\ O_t &= \sigma(W_{xo}X_t + W_{ho}H_{t-1} + b_o) \\ F_t &= \sigma(W_{xf}X_t + W_{hf}H_{t-1} + b_f) \end{aligned} \tag{20}$$

σ denotes the sigmoid activation function.

The first gate that is introduced is the input gate I_t. This gate determines how much of the input node's value $\tilde{C}_t$ should be added to the current memory cell internal state C_t. The output gate O_t determines how much the memory cell should influence the current computed hidden state H_t. The forget gate determines whether to keep the current value of the memory cell, to surpress the current value of the memory cell, or to discard it once it has learned, that the stored information is out of date and not of use anymore. For all the gates, a value closer to 0 means that less information is passed through and a value closer to 1 means that more information is passed through.
The hidden state and cell state is calculated as shown in equation (21) and (22).

$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t \tag{21}$$
$$H_t = O_t \odot \tanh(C_t) \tag{22}$$

$\odot$ denotes the Hadamard elementwise product operator [25].

The bias of the forget gate b_f is usually initialized with negative values, whereas the bias of the input b_i and output gates b_o are usually initialized with positive values. This makes the LSTM module use most of the information at the first iterations of the training phase, before it then starts to learn to forget [27].

4.6 Key Performance Indicators (KPI)

It is important to evaluate the forecasting ability of the different models using Key Performance Indicators (KPI).

$$MAPE = \frac{1}{T}\sum_{t=1}^{T}\frac{|\hat{y}_t - y_t|}{y_t} \tag{23}$$

Equation (23) is the Mean Absolute Percentage Error (MAPE). It is scale dependent and good at capturing outliers, but it is sensitive to small demand, especially when demand is 0.

$$nMAE = \frac{\frac{1}{T}\sum_{t=1}^{T}|\hat{y}_t - y_t|}{\frac{1}{T}\sum_{t=1}^{T} y_t} \tag{24}$$

Equation (24) is the normalized Mean Absolute Error, known as nMAE or MAE%. It tackles the issue of scale dependence in MAE, by dividing with the mean of the ground truth. In contrast to MAPE, it can lose some information about outliers due to the aggregation before scaling.

$$nRMSE = \frac{\sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}}{\frac{1}{T}\sum_{t=1}^{T} y_t} \tag{25}$$

Equation (25) is the normalized Root Mean Squared Error, known as nRMSE or RMSE%. Similar to nMAE, it addresses the issue of scale dependency for RMSE.

4.7 Diebold-Mariano Test

For time series $\{y_t\}_{t=1}^{T}$ and two separate forecasts $\{\hat{y}_t^{(1)}\}_{t=1}^{T}, \{\hat{y}_t^{(2)}\}_{t=1}^{T}$, the Diebold-Mariano test is a two sided test used to test if the difference in accuracy between forecasts is statistically equal.
Let forecast errors be $\epsilon_t^{(i)} = \hat{y}_t^{(i)} - y_t$ for $i = 1, 2$, and let loss function $l(\epsilon_t^{(i)})$ be squared error. Let $d_t = l(\epsilon_t^{(1)}) - l(\epsilon_t^{(2)})$ be the loss differential.
The null hypothesis states that both forecasts have equal accuracy $H_0 : \forall t : E[d_t] = 0$. Hence, the mean of the loss differential is 0. The alternative hypothesis is that forecast accuracy is not equal $H_A : E[d_t] \neq 0$.
The Diebold-Mariano test assumes stationarity of the loss differential d_t, but is robust to forecast errors being non-gaussian, non-zero mean and serially correlated.

$$DM = \frac{\bar{d}}{\sqrt{\frac{2\pi\hat{f}_d(0)}{T}}} \tag{26}$$

The test statistic is shown in equation (26), and $DM \sim N(0,1)$ under the null hypothesis. $\bar{d}$ is the mean of the loss differential and $\hat{f}_d(0)$ is the spectral density of the loss differential, calculated as a weighted sum of the sample auto-covariance [28].

4.8 SHapley Additive exPlanations (SHAP)

SHapley Additive exPlanations (SHAP) is a model agnostic framework for interpreting the local feature contribution of "black-box" models. It adds a layer of explainability to models that alone, are too large and complex for human interpretation [29].

As pointed out by Borup et. al. (2022), SHAP values are reasonable to use for cross-sectional data, where only one model is fitted on the training data and used to predict the test data. For time series contexts, where forecasts may by performed in an expanding or rolling window fashion, there is a sequence of models to interpret.

$$\hat{\phi}_p(X_t, W_i, h) = \sum_{Q \subseteq S \setminus \{p\}} \frac{|Q|!(P - |Q| - 1)!}{P!} [\hat{f}_{Q \cup \{p\}}(X_t, W_i, h) - \hat{f}_Q(X_t, W_i, h)] \tag{27}$$

SHAP values are based on coalition game theory and have the purpose of fairly assigning a contribution to players given their contribution to the total prediction.
Time consistent Shapley values $\hat{\phi}_p(X_t, W_i, h)$ are shown in equation (27) for feature $p \in S$, over window W_i and horizon h.
Q is a subset (coalition) of features and $S \setminus \{p\}$ is the set of all coalitions with $P - 1$ predictors excluding feature p. $\hat{f}_{Q \cup \{p\}}(X_t, W_i, h)$ is the predicted value of the models conditioned on coalition $Q \cup \{p\}$. The feature contribution from feature p is the difference between the predicted value conditioned on coalition $Q \cup \{p\}$ and Q. All marginal feature contributions are aggregated as a combinatorial weighted sum over all possible coalitions excluding p.

$$\hat{f}(X_t, W_i, h) = \hat{\phi}_{\emptyset}(X_t, W_i, h) + \sum_{p \in S} \hat{\phi}_p(X_{t,j}, W_i, h) \tag{28}$$

SHAP values have the properties of linearity, symmetry, dummy and, in equation (28), efficiency. The efficiency property states that the prediction $\hat{f}(X_t, W_i, h)$ at time t is fully decomposed as a base contribution and a sum of all feature contributions (SHAP values).
We measure global feature importance using Performance Based Shapley Values (PBSV), where (27) is altered to measure feature p's contribution to the global out-of-sample accuracy, rather than the raw prediction. We also use the Out-of-Sample Shapley Values to understand the local contribution of each feature [7].

5 Empirical analysis and results

5.1 Experimental setup

Our project code is programmed in Python and data has primarily been handled with the Pandas library.
Forecasting with Lasso, Random Forest and XGBoost is performed using the skforecast library. It is a scikit-learn compatible library that handles recursive forecasting, and easily allows for the use of first-differencing and scalar transformations.
Scikit-learn models have been estimated using an Apple M3 Pro CPU.

Vanilla RNN and LSTM are implemented in PyTorch using the Adam optimizer [30], MSE loss function and a one cycle learning rate scheduler [31]. We found no suitable package for recursive forecasting using PyTorch models, hence this part has been implemented by ourselves. The pseudo code for the implementation of recursive forecasting in PyTorch can be seen in appendix in section 8.1.
PyTorch training is accelerated using an NVIDIA GeForce RTX 3050 Laptop GPU with 4GB VRAM. Hyperparameter search is performed using an NVIDIA GeForce RTX 3060 GPU with 12GB VRAM.

$$X_{t,p}^{(scaled)} = \frac{X_{t,p} - \hat{\mu}_p}{\hat{\sigma}_p} \tag{29}$$

Each feature p has been transformed by the standard scalar in equation (29) before being input to any model. $\hat{\mu}_p$ and $\hat{\sigma}_p$ are the sample mean and standard deviation of the features.

For all results of forecasting below, the fixed window estimation approach was used, due to the long estimation time of RNN and LSTM. This approach made it computationally feasible to calculate a large number of results, while it has been tested that expanding and rolling window approaches provide very little to no relative improvement. Additionally, a forecast horizon of 40 was used (equivalent to 2 days), for evaluating forecast accuracy across all results below, with the exception of figure 16.

5.2 Analysis

5.2.1 Seasonality

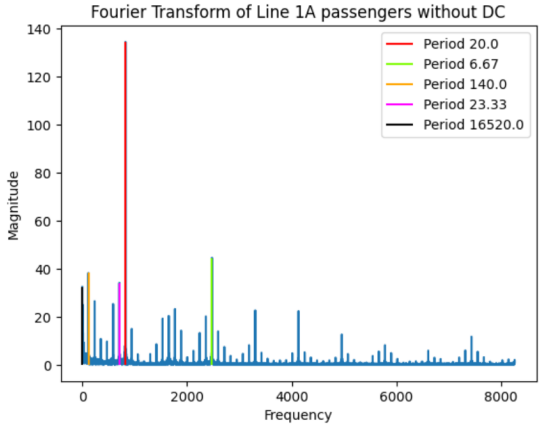

Figure 7: Frequency spectrum for Line 1A (60min agg. level), where the five highest magnitudes are highlighted.

Line	Periods				
1A	20	6.67	140	23.33	16520
2A	20	140	6.67	23.33	70
3A	20	6.67	140	23.33	70
4A	20	6.67	140	23.33	10
5A	20	6.67	140	10	70
6A	20	6.67	140	10	4

Figure 8: Periods in hours corresponding to the five frequencies with the highest magnitude for all lines (60min agg. level).

Figure 7 shows the result of the Fourier Transformation on line 1A on 60-minute aggregation level. The 5 highest magnitudes are the spikes at frequency 826, 2478, 118, 708 and 1, which corresponds to the periods 20 (24 hours), 6.67 (8 hours), 140 (1 week), 23.33 and 16520.
The Fourier analysis has been repeated for all bus lines in the table in figure 8, which shows a similar pattern across all lines. The frequencies corresponding to periods 20, 6.67 and 140 have the highest magnitudes across all six lines. It clearly indicates that bus passenger data contains a high degree of hourly, daily and weekly seasonality.

Figure 9 shows bus passenger data for line 1A for a specific week. Seasonality is shown as repeated temporal patterns, where each day sees two spikes at peak hours and few passengers in late night/early morning hours. The pattern repeats for Monday to Friday and breaks in the weekend. The observed seasonality is consistent with the table in figure 8.

Some weeks, where the seasonal pattern is less dominant, is in vacation periods. All vacation periods see a sharp drop in passenger demand across all bus lines as shown in figure 10. Winter, Easter, fall and Christmas vacations last 1-2 weeks and generally see very abrupt drops, while the Summer vacation lasts 6-7 weeks and has a gradually decreasing passenger count.

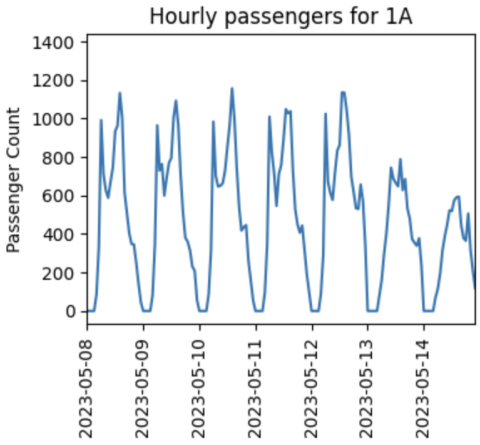

Figure 9: 1 week of hourly passenger count on line 1A from May 5th 2023 (Monday) to May 14th 2023 (Sunday).

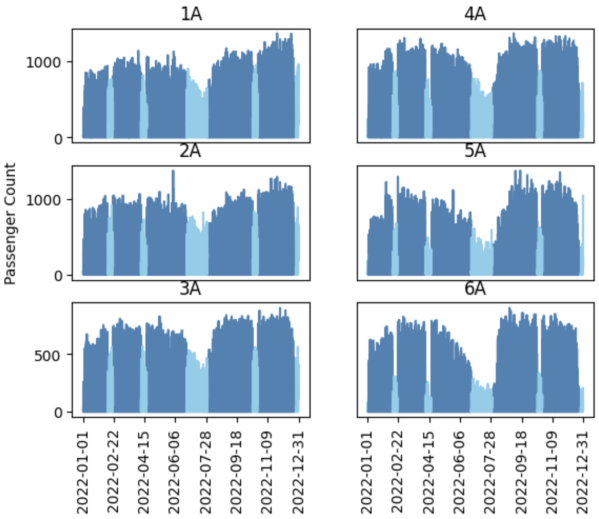

Figure 10: Hourly passengers during 2022. Vacation periods are highlighted with light blue.

5.2.2 Trend and Stationarity

There are different methods that can be applied in order to try to achieve stationarity. A simple method is to detrend the data, by computing the linear trend with OLS and subtracting each data point of the trend from the original data. The results of detrending can be seen in figure 11.

Another commonly used method to achieve stationarity is to first-difference the data, where the current data point is subtracted from the previous data point for all data points in the original data. The results from first-differencing can be seen in figure 12.

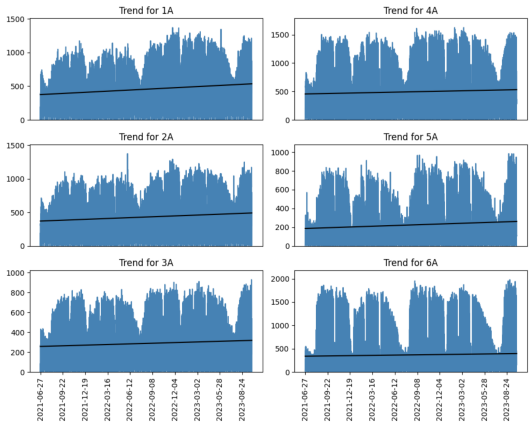

Figure 11: Plot of linear trend for all lines.

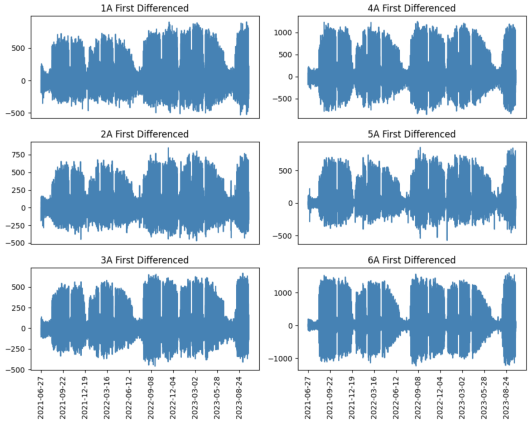

Figure 12: Plot of first-differenced data for all lines.

To check whether the original data is stationary or if the methods manage to make the data stationary, we use the ADF test and the KPSS test on the original data, the detrended data and the first-differenced data. Both the ADF test and the KPSS test is applied on all three aggregation levels and on all the six different bus lines.

Across all aggregation levels and bus lines for both the original data, the detrended data and the differenced data, the result of the ADF test suggests that the null hypothesis of non-stationarity can be rejected.

In contrast, the KPSS test on the original data suggests that the null hypothesis of stationarity can be rejected across all aggregation levels and bus lines. This means that the two tests yield conflicting results regarding stationarity on the original data.

For the detrended data, the result of the KPSS test suggests that the null hypothesis of stationarity can be rejected for line 1A on 15- and 30-minute aggregation levels and for line 1A, 2A and 3A on 60-minute aggregation level. For all other bus lines, the null hypothesis of stationarity cannot be rejected. This means that there is an agreement between the two tests, on that stationarity has been achieved by detrending, for some of the data.

For the first-differenced data, the results of the KPSS test instead suggests that the null hypothesis of stationarity cannot be rejected across all aggregation levels and bus lines. This means that there is an agreement between the two tests, that stationarity has been achieved by first-differencing, for all of the data.

We think that the dominant seasonality in the data is causing the stationarity disagreement between the ADF and KPSS test. When the original data is decomposed using a Multiple Seasonal-Trend decomposition using Loess (MSTL) with seasonal periods 6, 7, 20 and 140 hours, then both ADF and KPSS agrees that the residuals are stationary across all bus lines and all aggregation levels. This reinforces the hypothesis that seasonality could be causing the non-stationarity of the original data.

5.3 Modelling

5.3.1 Hyperparameter Search

Model	Diff	Exog	Lags	Hyperparameters
Lasso	d=0	C, W	20, 40, 60, 80, 100, 120, 140	$\alpha = 0.005$
Lasso	d=1	C, W	1, 10, 20, 140	$\alpha = 0.005$
RF	d=0	C, W	20, 40, 60, 80, 100, 120, 140	$estimators = 250$
RF	d=1	C, W	20, 40, 60, 80, 100, 120, 140	$estimators = 250$
XGB	d=0	C	20, 40, 60, 80, 100, 120, 140	$estimators = 100$, $\alpha = 0.3$, $\eta = 0.1$
XGB	d=1	C	1, 20, 140	$estimators = 250$, $\alpha = 0.2$, $\eta = 0.05$
RNN	d=0	C	20, 40, 60, 80, 100, 120, 140	bi-directional=True, hidden size=100, layers=3, dropout=0.1, max lr=0.0001, epochs=100, batch size=32
RNN	d=1	C	1, 4, 8, 20, 140	bi-directional = False, hidden size = 100, layers = 3, dropout = 0.5, max lr = 0.0001, epochs = 100, batch size = 32
LSTM	d=0	C	20, 40, 60, 80, 100, 120, 140	bi-directional = True, hidden size = 100, layers = 2, dropout = 0.5, max lr = 0.001, epochs = 75, batch size = 32
LSTM	d=1	C	20, 40, 60, 80, 100, 120, 140	bi-directional = True, hidden size = 40, layers = 2, dropout = 0.25, max lr = 0.001, epochs = 75, batch size = 32

Table 3: Optimal hyperparameters for line 1A. C denotes including calendar variables and W denotes including weather variables.

Optimal model hyperparameters have been estimated using a hyperparameter search on the validation data. Lasso, Random Forest and XGBoost all have low estimation times (0.01 second, 55 seconds and 0.7 seconds respectively on CPU), which allows us to perform a hyperparameter grid search over many combinations of parameters and lags and testing models with or without

first-differencing, calendar variables and weather variables.
RNN and LSTM have long estimation times (3-5 minutes with GPU accelerated training) and many hyperparameters, which results in a very large grid search. Instead we use an iterative hyperparameter search, where only one hyperparameter is being searched at a time, while using the best found hyperparameters up to that point. The hyperparameter search is thereby not searching through all combinations to find a set of optimal hyperparameters, but an approximation of the optimal set of hyperparameters.

Table 3 shows the optimal hyperparameters for forecasting on the validation data. Note that for non-differenced data, all models find lags 20, 40, 60, 80, 100, 120, 140 to be optimal lags.
XGBoost is found to have a smaller model size than Random Forest with 100 estimators compared to 250 estimators. Similarly, for LSTM and RNN, LSTM finds a hidden size of 100 with only 2 layers to be optimal, compared to 3 layers for RNN. LSTM is also found to need 75 epochs of training compared to 100 for RNN.
XGBoost and LSTM are similar, in the sense that they have more advanced model architectures than Random Forest and RNN respectively, which can explain why they can perform equally good or better using a smaller model.

Table 6 and table 7 in the appendix contain all results from the hyperparameter search for line 1A. It is worth noting that using daily lags gives a relative improvement in nMAE of between 58.6%-62.3% across all models. Surprisingly, including weather variables does not cause a major reduction in forecast error and in some cases, for XGBoost, RNN and LSTM, it increases the forecasting error when combined with lags.
For the non-differenced validation data, the benchmark has a low error of 13.4% nMAE, while Lasso slightly beats the benchmark with 12.6% nMAE, Random Forest and XGBoost performs similar with 10% and 9.9% nMAE, while RNN and LSTM achieves 9.7% and 9.6% nMAE.
For the first-differenced data, the benchmark has a slightly higher error of 14.8% nMAE, while Lasso achieves 17.2% nMAE, Random Forest 17.5% nMAE, XGBoost 16.8% nMAE, RNN 18% nMAE and LSTM 18.3% nMAE. The first-differenced benchmark here proves harder to beat, as no model that uses the first-differenced data achieves better performance. This may be caused by the fact that first-differenced predictions have to be inverse-differenced before accuracy is evaluated, which leads to an accumulation of prediction errors.

5.3.2 Forecasting Results

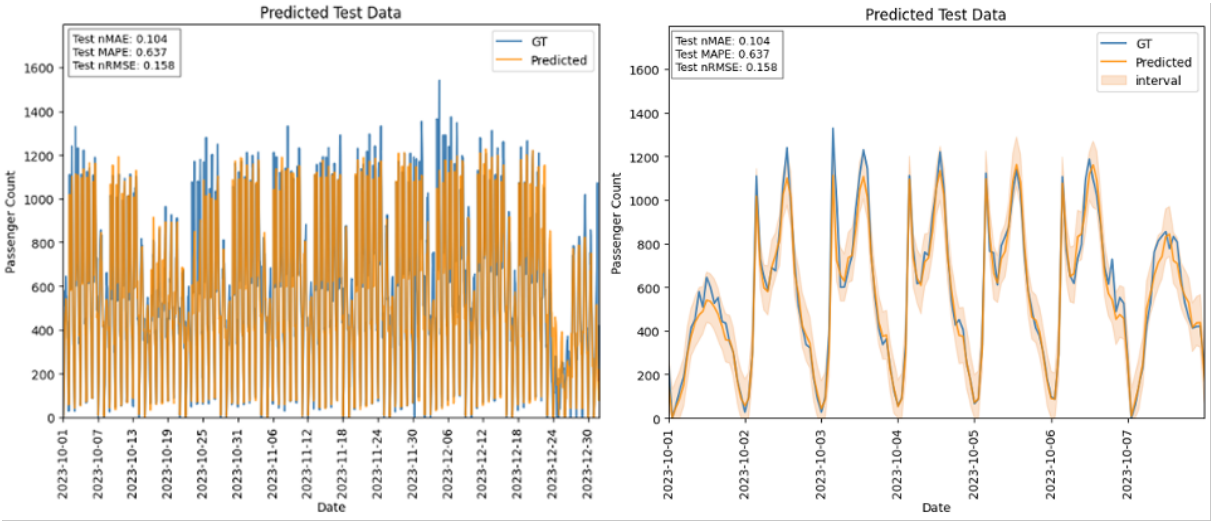

Figure 13: Plot of predictions from XGBoost on line 1A, using 40 hour horizon for all of the test data, as well as a plot of a magnified view of the first 7 days with a 95% confidence interval, from October 1st 2023 to October 7th 2023.

To evaluate the forecasting ability of our models, using the hyperparameters found from the hyperparameter search, we test the best version of each on the test data, to check for consistency. Figure 13 shows XGBoost's predictions of line 1A on the full test data (left) and a magnified plot of the first week in the test data (right). We can inspect and see that XGBoost has captured trends and seasonalities, e.g. it can effectively model both morning hours and peak hours, but it fails to successfully predict some very high peaks, that randomly occur during peak hour in some weeks. It has also successfully captured the decrease in passengers during fall vacation, October 16th - 20th, but struggles to recover after the vacation, which is likely rooted in the use of lags from the previous week, which is a vacation period.
The magnified plot shows that XGBoost has successfully captured the dominant daily and weekly seasonal patterns, by following the ground truth closely. It does successfully identify peaks at peak hours, but in some cases it fails to reach the peak values. The predictions also contain a bootstrapped 95% confidence interval, with a coverage of 88.59%, which is slightly lower than the expected 95%. The confidence interval is wider during peak hour and weekends, which indicates that those periods have higher uncertainty than the remaining hours of the day.

Table 4 shows the performance of all models across all bus lines using the optimal hyperparameters on a 60-minute aggregation level. Both the training, validation and test performance is shown.
For validation data, LSTM is the best performing model on most KPI's for line 1A, 2A, 3A, 4A and 6A. Random Forest is the best performing model on line 5A.
For test data, Vanilla RNN is the best performing model for line 1A, 2A and 3A on most KPI's, LSTM for line 4A, Random Forest for line 5A and XGBoost for line 6A, but the differences are marginally different in most cases.

			Training			Validation			Test		
Line	Diff	Model	nMAE	MAPE	nRMSE	nMAE	MAPE	nRMSE	nMAE	MAPE	nRMSE
1A	d=0	SNaive	0.170	0.229	0.278	0.134	0.150	0.193	0.156	0.955	0.269
		Lasso	0.146	0.255	0.204	0.126	0.233	0.172	0.143	0.905	0.212
		RF	0.038	0.054	0.054	0.100	0.120	0.146	0.107	0.675	0.163
		XGB	0.089	0.125	0.123	0.099	0.123	0.143	**0.104**	0.637	0.158
		RNN	0.101	0.142	0.139	0.098	0.125	0.140	**0.104**	**0.629**	**0.154**
		LSTM	0.091	0.118	0.126	**0.096**	**0.115**	**0.137**	0.105	0.645	0.159
2A	d=0	SNaive	0.161	0.288	0.270	0.122	0.141	0.181	0.152	0.380	0.261
		Lasso	0.138	0.375	0.193	0.115	0.290	0.162	0.141	0.399	0.206
		RF	0.036	0.062	0.051	0.093	0.111	0.141	0.102	**0.180**	0.153
		XGB	0.083	0.175	0.114	0.091	0.124	0.136	**0.101**	0.224	0.151
		RNN	0.093	0.201	0.129	0.090	0.145	0.133	**0.101**	0.285	**0.148**
		LSTM	0.087	0.143	0.122	**0.087**	**0.106**	**0.131**	0.102	0.196	0.156
3A	d=0	SNaive	0.171	0.252	0.295	0.142	0.168	0.207	0.172	0.308	0.301
		Lasso	0.148	0.340	0.213	0.138	0.275	0.193	0.160	0.365	0.236
		RF	0.038	0.057	0.054	0.108	0.138	0.158	0.111	**0.164**	0.168
		XGB	0.087	0.147	0.121	0.105	0.136	0.152	0.109	0.177	**0.165**
		RNN	0.096	0.167	0.136	0.103	0.140	**0.148**	**0.108**	0.186	**0.165**
		LSTM	0.090	0.140	0.127	**0.102**	**0.130**	**0.148**	0.114	0.188	0.174
4A	d=0	SNaive	0.161	0.316	0.290	0.123	0.156	0.180	0.152	0.437	0.282
		Lasso	0.141	0.394	0.212	0.127	0.339	0.174	0.146	0.466	0.223
		RF	0.033	0.070	0.050	0.093	**0.125**	0.137	0.098	**0.177**	0.151
		XGB	0.076	0.216	0.108	0.090	0.144	0.129	**0.095**	0.274	0.145
		RNN	0.086	0.230	0.123	0.091	0.170	0.127	0.096	0.353	0.147
		LSTM	0.079	0.184	0.112	**0.087**	0.165	**0.122**	**0.095**	0.238	**0.142**
5A	d=0	SNaive	0.230	0.334	0.395	0.183	0.225	0.274	0.234	0.436	0.396
		Lasso	0.200	0.457	0.295	0.181	0.364	0.260	0.211	0.487	0.321
		RF	0.050	0.080	0.077	0.136	**0.179**	**0.206**	**0.142**	**0.225**	**0.227**
		XGB	0.115	0.211	0.169	**0.135**	0.185	0.207	0.146	0.228	0.232
		RNN	0.135	0.237	0.202	0.138	0.209	0.208	0.146	0.244	0.235
		LSTM	0.128	0.216	0.189	0.137	0.185	0.207	0.146	0.228	0.236
6A	d=0	SNaive	0.199	0.256	0.427	0.139	0.167	0.248	0.200	0.342	0.417
		Lasso	0.194	0.397	0.325	0.172	0.361	0.270	0.199	0.452	0.333
		RF	0.038	0.052	0.067	0.105	**0.130**	0.185	0.111	0.180	0.183
		XGB	0.087	0.142	0.134	0.104	0.142	0.174	**0.110**	**0.176**	**0.179**
		RNN	0.103	0.163	0.170	0.104	0.154	0.165	0.115	0.194	0.209
		LSTM	0.094	0.139	0.150	**0.099**	0.132	**0.162**	0.111	0.180	0.201

Table 4: Performance of best models on all lines. Metrics marked red indicates performance worse than benchmark, and metrics in bold have best performance for the respective line.

Both Random Forest, XGBoost, Vanilla RNN and LSTM achieve similar performance across all lines. In testing, they achieve a 33% – 45% relative nMAE improvement compared to the benchmark, where Lasso only achieves 0.5% – 9.8% relative nMAE improvement compared to the benchmark.
Additionally, Lasso consistently has a worse MAPE than the benchmark. In some cases Lasso achieves a lower validation nMAE than training nMAE, which indicates that it underfits the data and is not able to learn the underlying patterns.
Random Forest consistently has a much lower training nMAE compared to validation and test nMAE, which indicates overfitting, but it does not appear to hurt forecasting performance.
LSTM outperforms most other models in validation, but not in testing, which suggests that it does not generalize well, which can be caused by the vast amount of hyperparameters and the complex model architecture.

The performance of the different models was also recorded for the other aggregation levels on the validation data. Here it should be noted that the hyperparameters that were found to be best on the 60-minute aggregation level was used. This was done, since the hyperparameter search was already very computationally expensive, and thereby took an extensive amount of time to compute, which would have more than doubled, if it was computed on a lower aggregation level.
Noteable from the results, on the other aggregation levels on the validation data, is that lower aggregation levels result in a significant drop in forecasting performance for both the benchmark and all other models. This supports using the 60-minute aggregation level data for forecasting.
Where LSTM and Vanilla RNN were some of the best performing models on the 60-minute aggregation level validation data, they were outperformed by XGBoost on the 30-minute aggregation level validation data, and they started to perform worse than the Seasonal Naive model on the 15-minute aggregation level validation data, where XGBoost again performed best. These results can be seen in appendix in table 9 and table 10. It further supports the argument that it is difficult to ensure consistency and generalizability for LSTM, as well as for Vanilla RNN.

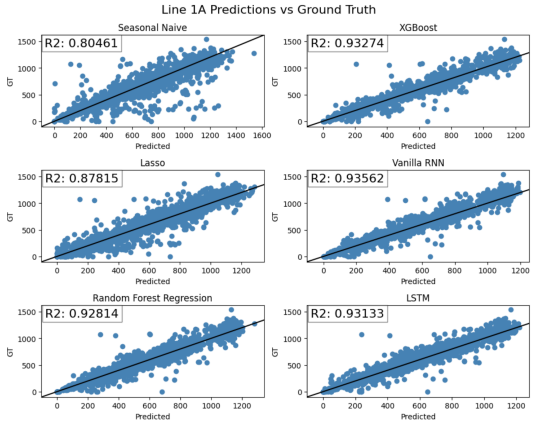

Figure 14: Plot of predictions against ground truth test data for all the models on line 1A.

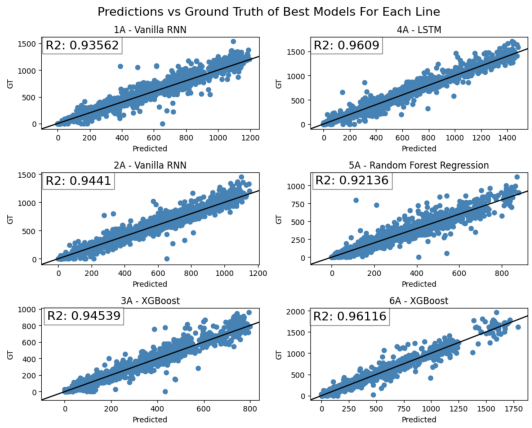

Figure 15: Plot of predictions against ground truth test data for the best model, using R^2 score, on each line.

Figure 14 shows the predicted values plotted against the ground truth test data for all models on line 1A, where the black line indicates a perfect forecast. The R^2 score for the fit of the black line is calculated and used as an indicator of forecast precision. The benchmark model achieves the lowest R^2 score of 0.805, while the vanilla RNN has the highest R^2 score of 0.936. This suggests that the Vanilla RNN is the best performing model on line 1A, which is consistent with the results from table 4.
Figure 15 shows the R^2 score of predictions against the ground truth test data for the best model on each bus line.

5.3.3 Evaluation of Forecasts

The goal for the project is to be able to accurately forecast 2 days ahead, as forecasting long horizons will in most cases significantly decrease forecasting accuracy. Furthermore, 48 hours was initially chosen, as practical implementations would need to use weather forecasts from DMI's Forecast API, which is only available for the next 2 days ahead. We are still interested in analysing how the forecast horizon impacts accuracy, to analyze the robustness of each model's forecasting ability.

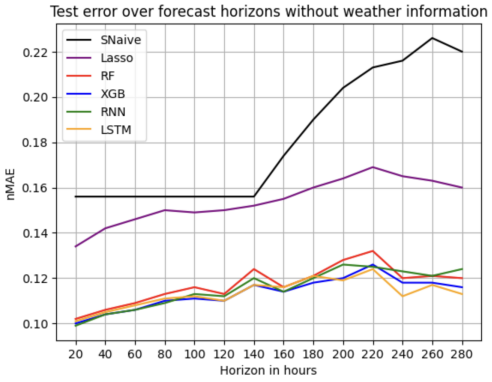

Figure 16: nMAE of forecasts over increasing forecast horizon on test data for line 1A. Models are fitted without weather information.

Figure 16 shows the persistence of nMAE over increasing forecast horizons for all models on the test data. The test data is forecasted using a forecast horizon ranging from 20 hours (1 day) to 280 hours (14 days), while the data does not include weather variables, since these would be unknown for forecast horizons larger than 40 hours (2 days).

The benchmark uses a lag of 140, and therefore the performance is constant for horizons 20-140. For horizons larger than 140 hours, the error starts deterring quickly. Random Forest, XGBoost, RNN and LSTM all achieve a similar performance, but XGBoost and LSTM consistently perform slightly better than Random Forest and RNN. There is a 26% and 23% increase of relative nMAE for XGBoost and LSTM respectively. The horizon-accuracy trade off shows that longer forecast horizons can be used with only a minor decrease in accuracy.

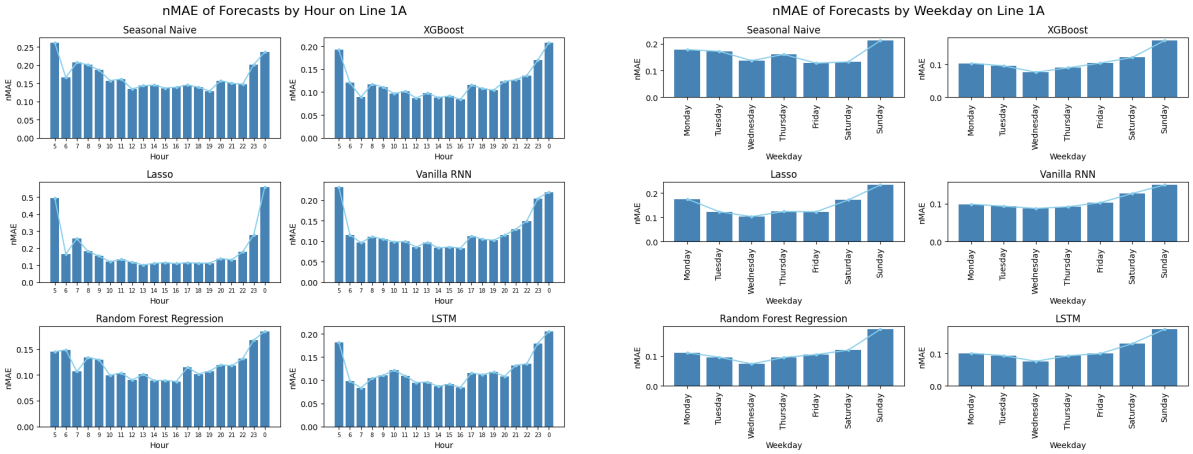

Figure 17: nMAE of forecast residuals grouped by hour (left) and day (right) for all models on the test data of line 1A.

The nMAE of grouped residuals can be used to analyze which time periods are the most difficult to forecast. Figure 17 (left) shows the nMAE of the forecast residuals for all models grouped by hour. It shows that early morning and late night hours have the highest relative error across all models. Lasso has significantly worse performance with nMAE of 50%, which is twice as high as 25% for the benchmark in hour 5. At peak hours, all models encounter slightly lower errors, which is likely caused by the large amount of passengers. All remaining hours have a stable error,

evenly spread throughout the day.
Figure 17 (right) shows the nMAE of the forecast residuals grouped by weekday. The nMAE is highest on Sundays. On Saturdays the nMAE is slightly higher compared to weekdays across all models except the benchmark. It suggests that difference in travel patterns between weekdays and weekends slightly worsens the predictive power in the weekends.

Many forecasts have been obtained, and it has been shown that there is only a small difference in predictive accuracy between many of the models. Therefore, the significance of differences in forecast accuracy between models is evaluated using the Diebold-Mariano test, where the full table of results can be found in appendix in table 11. The two-sided Diebold-Mariano test is used to determine a significant or insignificant difference in forecast accuracy, between every combination of two models, using a significance level of $\alpha = 0.05$.

Line	Best performing models sequentially
1A	XGB = RNN > LSTM > RF > Lasso > SN
2A	RF = XGB = RNN > LSTM > Lasso > SN
3A	RF = XGB = RNN > LSTM > Lasso > SN
4A	XGB = RNN = LSTM > RF > Lasso > SN
5A	RF = XGB > RNN = LSTM > Lasso > SN
6A	RF = XGB = LSTM > RNN > Lasso > SN

Table 5: Best performing models on test data according to two-sided Diebold-Mariano test. Results derived from table 11 in appendix.

Table 5 shows a summary of the full table 11 in appendix. The test shows that all models have significantly better forecasts than the Seasonal Naive model across all lines. It is also significant, across all lines, that Lasso performs worse than all other models, except for the benchmark. The results also show that LSTM has the best performance on two different lines, Random Forest and Vanilla RNN has the best performance on four different lines and XGBoost has the best performance on all six lines, having better or same forecast accuracy as the other models across all lines. This again shows that there are only small differences between the forecast accuracy of the different models, but XGBoost performs best on most the test data of the different bus lines, with an accuracy between 85.4%-90.5% nMAE, derived from table 4.

5.4 Feature Importance

We evaluate the feature importance using the time consistent Performance Based Shapley Values (PBSV), that estimate the feature importance in terms of accuracy contribution. We also estimate ordinary SHAP Values, where feature importance is estimated as the mean absolute feature contribution, as they cannot measure accuracy contribution.

Figure 18 shows the anatomized PBSV for XGBoost on the test data. It shows that lags 140 and 20 contribute most to the accuracy, 14% and 12% respectively. Peak hour, workday plan and Sunday/holiday plan are calendar variables that also significantly contribute to the accuracy, 5%, 2% and 2.5% respectively. Besides lags 40, 60, 80, 100 and 120, all other variables only have a minor impact between $0\% - 0.5\%$. Snow depth is the highest contributing variable among weather variables, which may be caused by the fact, that the test data spans over October-December, where snow occurs. Nonetheless, it is clear that no weather variables have a significant impact on accuracy.
The ordinary SHAP values in figure 19 shows the mean absolute feature contribution, which traditionally is the evaluation method for SHAP feature importance. It shows a similar ordering of feature importance as PBSV, but lag 20 has the highest contribution towards the predictions. Ordinary SHAP values estimate the contribution of weather variables and the event variable to be higher, where PBSV estimate that they have no or little contribution to the accuracy.

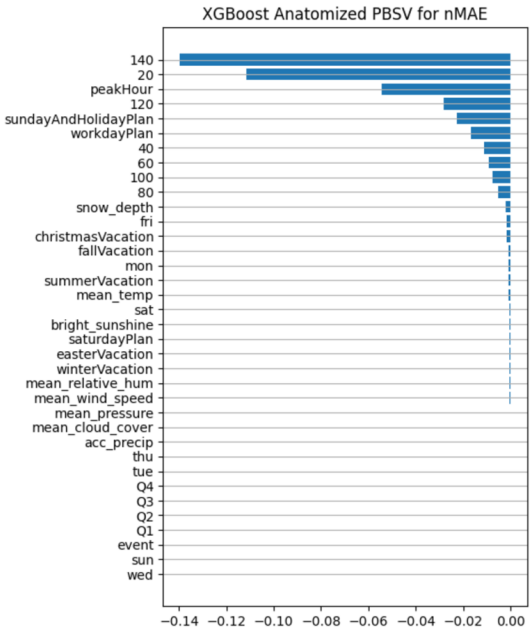

Figure 18: PBSV feature importance in terms of nMAE reduction for XGBoost on test data using daily lags and calendar/weather variables.

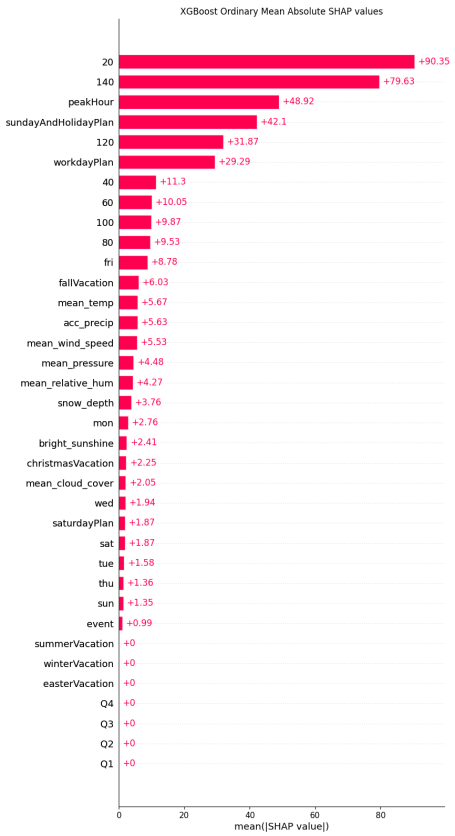

Figure 19: Feature importance from mean absolute value of ordinary SHAP values for XGBoost on the test data.

Anatomized PBSV for Lasso and Random Forest can be found in appendix in figure 22 and figure 23. They similarly show that lag 140 and 20 have the highest contribution. Peak hour, workday plan and Sunday/holiday plan are the highest contributing calendar variables, and all other variables besides lags have minor to no contribution at all.

We have not anatomized PBSV for RNN and LSTM as the computation time depends exponentially on the model training and inference time. Computing it with GPU accelerated training would take many months, which makes PBSV computationally infeasible in practice for deep learning models. The computation time of PBSV for Lasso, Random Forest and XGBoost was 10 minutes, 6 hours and 30 minutes respectively.

Inspecting the distribution of SHAP values helps understand what patterns the model has captured and utilized in forecasting the test data.

Figure 20 shows the distribution of SHAP values for each feature. The SHAP values are colorized based on

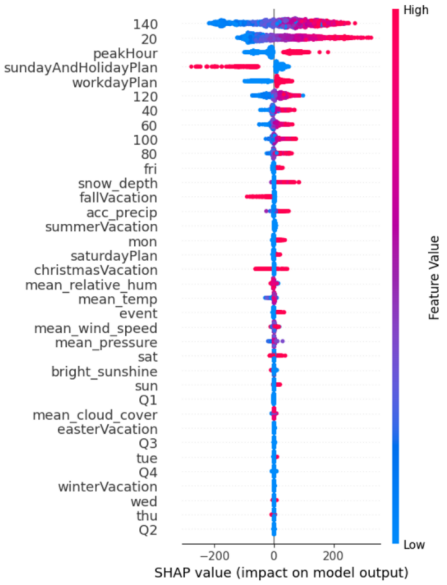

Figure 20: Feature contribution of out-of-sample time Shapley Values for XGBoost on the test data.

the value of the feature.
For all lag-features, if the lagged value is high, then it increases the value of the prediction, and decreases the value of the prediction if the lagged value is low. Predictions are also increased in peak hours, on workdays, at events and when snow and rain is present. Predictions are decreased on Sundays and holidays and in the fall vacation.
Christmas vacation yields both increasing and decreasing contributions, which may be caused by high travel demand before Christmas and New Years Eve and low travel demand in the following days. Summer, Winter and Easter vacation yields no contribution since the test data spans over a different time period.
XGBoost has captured patterns that are consistent with the seasonal, weekend and vacation patterns we found in the exploratory data analysis.

6 Discussion

6.1 Variable Selection

In forecasting bus passengers, we have seen that calendar features and historic seasonal patterns are crucial for the ability to forecast. We have purposely not included the amount of busses per hour as a feature.
The relationship between amount of busses and passengers is illustrated in figure 21, as a simultaneous equation model. The amount of busses can directly affect the amount of passengers and thereby impose an artificial constraint, e.g. if only one bus is driving during peak hours where demand is much higher. Conversely, when a sufficient amount of busses are supplied to comfortably accommodate all passengers, then that will be a saturation point, where an increase in busses is no longer proportional to the amount of passengers.

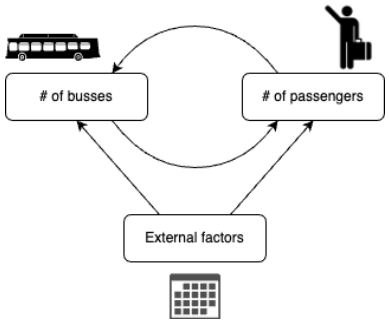

Figure 21: Simultaneous equation model of bus passengers, amount of busses and external factors.

The amount of passengers can affect the amount of busses, when public transportation providers observes or expects to observe high demand. They may schedule additional busses, or at low capacity they may schedule fewer busses.

Including amount of busses as a feature may lead to the false interpretation, that an increase in the amount of busses causes an increase in passengers. In broad perspectives, expanding public transportation in an area can increase the population's ability and willingness to use public transportation [32]. Regardless, passenger demand is inherently driven by external factors, such as travelling to school, work and social events.

Given these considerations, we could have experimented with amount of busses as a feature, as they are expected to be strongly correlated. We chose to focus on exogenous variables that may have a direct impact, e.g. weather and related calendar variables, to ensure accuracy and reliability. An example would be that Sunday/holiday plan is caused by Sundays, the alteration of passenger behavior is also caused by Sundays, but the passenger demand doesn't cause Sundays.

A different consideration that was adressed, was the consideration of multicollinearity, when estimating Lasso. We took measures to ensure that we would not get perfect multicollinearity, such that the model could be estimated, by removing the Q1 variable for the quarters and the Monday variable for the weekdays.
We are aware that there still can be strong multicollinearity between some of the remaining

variables, but the effect of removing them has been tested, which results in worse performance of the model.

6.2 Stationarity

In our analysis of trying to achieve stationarity, we used ADF test and KPSS test, and found that the original data was not stationary, while detrending helped and first-differencing did make the data stationary.

Stationarity means that the statistical properties do not change over time, which generally makes forecasting easier. It is still possible to forecast a non-stationary time series, but the results may be strongly misleading, whereas stationarity makes forecasts more reliable, especially for long horizons [17].

The ADF test concluded that the original data was stationary and the KPSS test concluded that it was not stationary. Nevertheless, an MSTL decomposition of the original data was performed, to test the hypothesis that seasonality could potentially be affecting the statistical tests for stationarity. On the residuals of the decomposed data, where seasonality had been removed, both tests then agreed upon stationarity.

From the results of the hyperparameter search on the original and first-differenced data, it was noticed that the forecasts on the original data had higher accuracy than the first-differenced data, despite the original data not being stationary. These results can be seen in table 6 and table 7 in appendix. As previously mentioned, the MSTL decomposition indicates that seasonality has a large effect on the stationarity of the data, which may interfere with ADF and KPSS test. Thereby, it could be that the original data may reasonably be assumed to be stationary, when accounting for seasonality by using lags and dummy variables for each day, peak hour, holidays etc.

The reason for worse forecasting accuracy using first-differenced data is also highly likely to stem from the accumulation of errors, as the predictions must be inversed differenced before accuracy is evaluated.

6.3 Modelling

In our search for hyperparameters, we settled on the plan to only run hyperparameter search for optimal hyperparameters on the 60-minute aggregated data from line 1A. It would have been optimal if we could have implemented a grid search for each model, on each line, on each aggregation level, but this was not possible due to the vast computation time of searching through each combination of all the many hyperparameters.
Due to this vast computation time, we also had to perform a less effective hyperparameter search than the grid search, for Vanilla RNN and LSTM, to be able to find an approximation of the optimal hyperparameters.

We are aware that this might have the consequence of us not having found the most optimal hyperparameters for each line and for each aggregation level. We can for example see that the performance of all models, and especially Vanilla RNN and LSTM, drops by a large margin, when switching from the 60-minute aggregation level to the lower aggregation levels of data, which is likely due to non-optimal hyperparameters.
Across each line, a drop in performance, due to non-optimal hyperparameters, looks to be less significant. Figure 15 shows that the performance on the other lines is roughly the same and sometimes better, when only using the best found hyperparameters for line 1A. Still, it is possible

that a new hyperparameter search for each line could yield even better results.

After having found hyperparameters for each model, we tested the models on the validation data, where many of the models seemed to have similar performance, while LSTM achieved the best performance by a small margin. Then the models were tested on the test data, where we found that many of the models also achieved similar performance. This made us test the significance of difference between forecast performance, using Diebold-Mariano test, where it was shown that Random Forest Regression, XGBoost, Vanilla RNN and LSTM indeed had similar performance, but in the end XGBoost, with other models as well, was the top performing model across all lines.

On the test data, we saw that the forecast from XGBoost effectively adapts to the fall vacation, but struggles to recover to the normal passenger flow in the following week, in figure 13. This problem may be solved by more advanced feature engineering, e.g. including a binary variable if a week is the first week after a vacation period and potentially including a special lag with the passenger count before the vacation started.
Alternately, a very different approach may be used, where the time series is split into four different series, one for weekday plan, one for Saturday plan, one for Sunday/Holiday plan and one for vacations. This approach removes the dependency between days that have structurally different travel patterns.

We have shown that XGBoost is a top performing model on the test data across all lines and is robust across longer forecast horizons with only a minor decrease in performance. It is robust to changes in aggregation levels, as the performance drop is much lower than LSTM and consistently better than the benchmark. Combined with a fast estimation time of 0.7 seconds, and only few hyperparameters that must be tuned, XGBoost is an excellent choice for balancing accuracy, estimation time and robustness.

6.4 Feature Importance

After having found the models that performed best, it was important to understand what variables were important for their ability to model the data, and whether they could be further improved. Hence, the Performance Based Shapley Values (PBSV) for the models were computed, and compared to the ordinary SHAP values for the models. It is important to note that these results are purely correlations and have not been tested for being causal effects. For example, it is shown in figure 20, that when the busses are following the Sunday/holiday plan, then there is a decrease in passenger demand. The decrease in passenger demand is clearly not due to the fact that the plan has changed. We are instead seeing the effect of it being Sunday, and people are not taking the bus to school and work.

We only found small differences in the importance ordering of features for PBSV and ordinary SHAP values, even though PBSV should be more accurate, as they are developed using time consistent SHAP values. Despite the similar ordering of features, we still argue that PBSV gives a better insight into global feature importance, as it measures the feature contribution to the accuracy, rather than the feature contribution to the raw prediction. Interpreting feature importance from ordinary SHAP values is based on the mean absolute feature contribution, but a large contribution does not necessarily imply a high importance to the overall accuracy.

For Vanilla RNN and LSTM, it was computationally infeasible to calculate PBSV, while it for Random Forest Regression took 6 hours to compute. For XGBoost, the computation only took 30 minutes, which is more reasonable, but the computation only took 1 minute for the ordinary SHAP values on XGBoost.
Hence, if the goal is efficiency, then computing the ordinary SHAP values is more efficient, but

the results are going to be an approximation of the actual accuracy contribution. Alternatively, if the goal is accuracy, then it would be optimal to compute PBSV, if it is computationally feasible. However, it is reassuring for the Deep Learning models, that the difference between the importance orderings of features is similar for Lasso, Random Forest and XGBoost.

6.5 Comparison of results

We have found that lowering the aggregation level from 60 minutes to 30 and 15 minutes decreases forecasting accuracy, which is the same conclusion found by Halyal et. al. (2022). We also found that all machine learning models and recurrent neural networks yield a 33%-45% relative nMAE improvement, compared to the Seasonal Naive model, where Halyal et. al. (2022) found LSTM to have the best performance, with a 27.46% relative nMAE improvement, compared to SARIMA.
We found that the linear model, Lasso, performs better than the benchmark, but worse than non-linear models, similar to the findings of Halyal et. al. (2022) for SARIMA.

We have shown that all weather-related variables have no significant contribution to the accuracy of forecasting bus passengers in Aarhus. Wei (2022) found that Australian passenger demand was resilient to weather in morning peak hours, but impressionable in the afternoon. Our results may suggest that Danish commuters are more resilient to rain and wind, which may be caused by the high frequency of weather fluctuations in Denmark.
In a practical implementation of forecasting passenger demand, it should be recalled from section 3 that weather forecasts from DMI's Forecast API would need to replace actual weather data. This adds additional uncertainty to forecasted passenger counts, as it builds upon weather forecasts. From our results, seeing that weather data has no significant impact on the performance, shows that the additional uncertainty of including weather forecasts instead will not be of great concern. Alternatively, weather can be discarded, allowing for forecast horizons larger than 2 days, e.g. for a forecast horizon of 14 days, we only saw a relative nMAE increase of up to 26% compared to the forecast horizon of 2 days.

Farahmand et al. (2023) included holiday calendar information, one hour lagged weather, as well as the match schedule of the major football club in the city. They found that holiday calendar information significantly improved their model. Including weather and one hour lagged weather also improved the model, while the football schedule had low impact. From our results, we found that including the different bus plans were of large importance to the performance of the models, while including vacation periods also slightly increased performance, as seen in figure 18. Weather was found to have no significant importance in our experiments, while modelling XGBoost with rain and wind speed lagged one hour was tried, which still yielded no performance change. In our implementation, we included many different events, taking place in the area surrounding the chosen bus routes. This was done to make the data less sparse, than had we only included football matches. Still, we also found that they had low impact, and additionally that they did not improve the performance of the model.

We have, through our seasonality analysis with Fourier Transformation in figure 7, shown that a thorough inspection of lags can significantly increase forecasting accuracy in highly seasonal time series, similar to the results from Cheng et. al (2022).

7 Conclusion

In this project, the performance of five models, selected on the basis of prior research, has been compared, in the setting of forecasting bus passenger data from the most in-demand buslines in

Aarhus, Denmark.

Using the most optimal hyperparameters for appropriately modelling the nature of bus passenger demand, all the proposed models were able to outperform the Seasonal Naive benchmark model on most KPI's, for the 60- and 30-minute aggregation levels, on most of the buslines.
Fourier Transformations proved to be efficient at discovering seasonal patterns. Including lags for the same hour every day for the past week was optimal for all non-differenced models, and gave a relative nMAE improvement between 58.6%-62.3%.
It was also discovered, that while we set out to forecast only the next 48 hours ahead, it was possible to forecast the next 2 weeks ahead with only up to around 26% relative nMAE increase, when not including weather variables.

The best performing model turned out to be XGBoost, which had an accuracy between 85.4%-90.5% nMAE, and achieved between 33%-45% relative nMAE improvement compared to the Seasonal Naive benchmark model. Additionally, XGBoost was found to be the best performing model across all chosen bus lines, according to the statistical significance of the Diebold-Mariano test.
It was discovered that XGBoost was more generalizable to the 15- and 30-minute aggregation levels of the data as well, especially when compared to Vanilla RNN and LSTM.

While being able to forecast accurately is a crucial task for optimizing public transportation, one of the compelling aspects of the project has been the application of time consistent Performance-Based Shapley Values (PBSV), to gain insight into feature importance and feature contribution to each individual prediction.

From PBSV, it was concluded, that all weather variables, except snow depth, had little to no contribution to the accuracy of the model. The features contributing the most to the accuracy of the model was the daily lags, whether it was peak hour and the different types of bus schedules. These results were compared to the estimation of feature importance from the ordinary SHAP values, where the importance ordering of the features often were similar, although the ordinary SHAP values credited the weather variables with a little higher relative importance. Despite similar feature ordering, we contend that PBSV provides better insight into global feature importance, by measuring contributions to accuracy, rather than the feature contribution to the raw prediction.

While the study of PBSV has provided valuable insights, it remains infeasible to calculate PBSV for the models utilizing Recurrent Neural Networks. Hence, it may be a significant aspect for further exploration, to implement a method to more efficiently estimate PBSV.
Another area ripe for future work, is to do a more in-depth hyperparameter search. Further exploring new features that describe passenger behavior may lead to new combinations of hyperparameters, lags and exogenous variables, that better explains the randomness involved in passenger behavior, and thereby improve forecasting of bus passenger demand.

7.1 Use of Generative AI

Generative AI (ChatGPT) has been used for inspiration when writing the project. It has been used to reformulate specific sentences and evaluate paragraphs and the coherence of the report.
Generative AI has not been used to generate text to be inserted into the project report, and generated text has not been altered/made changes to, with the purpose of being inserted into the project report.
Everything is written by Andreas Hyldegaard Hansen and Andreas Skiby Andersen.

References

[1] Midttrafik. *Budget 2024 Bilag 1 - Hovednotat*. 2023. URL: https://www.midttrafik.dk/media/30819/b-2024-bilag-1-hovednotat-aod.pdf. accessed: 08-02-2024.

[2] I. Mulalic and J. Rouwendal. "Does improving public transport decrease car ownership? Evidence from a residential sorting model for the Copenhagen metropolitan area". In: *Regional Science and Urban Economics* 83 (2020). URL: https://www.sciencedirect.com/science/article/pii/S0166046219302157.

[3] M. Wei. "How does the weather affect public transit ridership? A model with weather-passenger variations". In: *Journal of Transport Geography* 98 (2022). URL: https://www.sciencedirect.com/science/article/pii/S0966692321002957.

[4] Z. Farahmand, K. Gkiotsalitis, and K. Geurs. "Predicting bus ridership based on the weather conditions using deep learning algorithms". In: *Transportation Research Interdisciplinary Perspectives* 19 (2023). URL: https://www.sciencedirect.com/science/article/pii/S2590198223000805.

[5] C. Cheng, M. Tsai, and Y. Cheng. "An Intelligent Time-Series Model for Forecasting Bus Passengers Based on Smartcard Data". In: *Applied Sciences* 12 (9) (2022). URL: https://doi.org/10.3390/app12094763.

[6] R. Marcinkevics and J. Vogt. "Interpretable and explainable machine learning: A methods-centric overview with concrete examples". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 13 (3) (2023). URL: https://wires.onlinelibrary.wiley.com/doi/full/10.1002/widm.1493.

[7] D. Borup, P. Coulombe, D. Rapach E. Schütte, and S. Schwenk-Nebbe. "The Anatomy of Out-of-Sample Forecasting Accuracy". In: *FRB Atlanta Working Paper No. 16* (2022). URL: https://doi.org/10.29338/wp2022-16.

[8] M. Ifraz, A. Aktepe, S. Ersoz, and T. Cetinyokus. "Demand forecasting of spare parts with regression and machine learning methods: Application in a bus fleet". In: *Journal of Engineering Research* 11 (2) (2023). URL: https://www.sciencedirect.com/science/article/pii/S2307187723000585.

[9] T. Tang, R. Liu, and C. Choudhury. "Incorporating weather conditions and travel history in estimating the alighting bus stops from smart card data". In: *Sustainable Cities and Society* 53 (2020). URL: https://www.sciencedirect.com/science/article/pii/S2210670719321274.

[10] L. Monje, R. Carrasco, C. Rosado, and M. Sánchez-Montañés. "Deep Learning XAI for Bus Passenger Forecasting: A Use Case in Spain". In: *Mathematics* 10 (9) (2022). URL: https://www.mdpi.com/2227-7390/10/9/1428.

[11] S. Liyanage, R. Abduljabbar, D. Hussein, and P. Tsai. "AI-based neural network models for bus passenger demand forecasting using smart card data". In: *Journal of Urban Management* 11 (3) (2022). URL: https://www.sciencedirect.com/science/article/pii/S2226585622000280.

[12] S. Halyal, R. Mulangi, and M. Harsha. "Forecasting public transit passenger demand: With neural networks using APC data". In: *Case Studies on Transport Policy* 10 (2) (2022). URL: https://www.sciencedirect.com/science/article/pii/S2213624X22000633.

[13] M. Yan and K. Zhou. "Bus Passenger Flow Prediction Using BO-Optimized XGBoost". In: *2nd International Conference on Big Data, Information and Computer Network (BDICN), Xishuangbanna, China* (2023), pp. 134–141. URL: https://ieeexplore.ieee.org/abstract/document/10109905.

[14] F. Jiao, L. Huang, R. Song, and H. Huang. “An Improved STL-LSTM Model for Daily Bus Passenger Flow Prediction during the COVID-19 Pandemic”. In: *Sensors* 21 (17) (2021). URL: https://www.mdpi.com/1424-8220/21/17/5950.

[15] Y. Ye, L. Chen, and F. Xue. “Passenger Flow Prediction in Bus Transportation System using ARIMA Models with Big Data”. In: *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery* (2019), pp. 436–443. URL: https://ieeexplore.ieee.org/document/8945914.

[16] DMI. *DMI Open Data Documentation*. 2024. URL: https://opendatadocs.dmi.govcloud.dk/DMIOpenData. accessed: 29-02-2024.

[17] C. Hanck, M. Arnold, A. Gerber, and M. Schmelzer. *Introduction to Econometrics with R*. 2024. Chap. 3.5, 6.4, 14.7. URL: https://www.econometrics-with-r.org/ITER.pdf.

[18] P. Bloomfield. *Fourier Analysis of Time Series: An Introduction*. 2nd ed. John Willey Sons, Inc, 2000. Chap. 4.2, 5.

[19] E. Zivot. *Introduction to Computational Finance and Financial Econometrics with R*. Springer, 2016. Chap. 4.1. URL: https://bookdown.org/compfinezbook/introcompfinr/Stochastic-Processes.html.

[20] D. Kwiatkowski, P. Phillips, P. Schmidt, and Y. Shin. “Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?” In: *Journal of Econometrics* 54 (1-3) (1992), pp. 159–178. URL: https://www.sciencedirect.com/science/article/pii/030440769290104Y.

[21] R. Tibshirani. “Regression Shrinkage and Selection via the Lasso”. In: *Journal of the Royal Statistical Society* 58 (1) (1996), pp. 267–288. URL: https://www.jstor.org/stable/2346178?seq=1.

[22] L. Lin, W. Fang, X. Xiaolong, and Z. Shissheng. “Random forests-based extreme learning machine ensemble for multi-regime time series prediction”. In: *Expert Systems with Applications* 83 (2017), pp. 164–176. URL: https://www.sciencedirect.com/science/article/pii/S0957417417302488.

[23] R. Genuer and J. Poggi. *Random Forests with R*. 2020. Chap. 1, 3. URL: https://link.springer.com/book/10.1007/978-3-030-56485-8.

[24] T. Chen and C. Guestrin. “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016). URL: https://arxiv.org/abs/1603.02754.

[25] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. *Dive into Deep Learning*. Cambridge University Press, 2023. Chap. 9, 10. URL: https://d2l.ai/index.html.

[26] P. J. Werbos. “Backpropagation through time: what it does and how to do it”. In: *Proceedings of the IEEE* 78 (10) (1990), 1550–1560. URL: https://ieeexplore.ieee.org/document/58337.

[27] F. A. Gers, J. Schmidhuber, and F. Cummins. “Learning to forget: continual prediction with LSTM”. In: *International Conference on Artificial Neural Networks ICANN* 2 (1999), pp. 850–855. URL: https://ieeexplore.ieee.org/document/818041.

[28] F. Diebold and R. Mariano. “Comparing Predictive Accuracy”. In: *Journal of Business Economic Statistics* Vol. 13 (1995). URL: https://www.tandfonline.com/doi/abs/10.1198/073500102753410444.

[29] L. S. Shapley. “A Value for n-Person Games”. In: *Contributions to the Theory of Games* Vol. 2 (1953), pp. 307–317. URL: https://www.degruyter.com/document/doi/10.1515/9781400829156-012/pdf.

[30] D. P. Kingma and J. L. Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations* (2014). URL: https://arxiv.org/abs/1412.6980.

[31] L. N. Smith and N. Topin. “Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates”. In: *Artificial intelligence and machine learning for multi-domain operations applications* Vol. 11006 (2018), pp. 369–386. URL: https://arxiv.org/abs/1708.07120.

[32] M. Gascon, O. Marquet, E. Gràcia-Lavedan, A. Ambròs, T. Götschi, A. de Nazelle, L. Int Panis, R. Gerike, C. Brand, E. Dons, U. Eriksson, F. Iacorossi, I. Ávila Palència, T. Cole-Hunter, and M. J. Nieuwenhuisjen. “What explains public transport use? Evidence from seven European cities”. In: *Transport Policy* 99 (2020), pp. 362–374. URL: https://www.sciencedirect.com/science/article/pii/S0967070X20303000.

8 Appendix

8.1 Pseudo code for recursive forecasting

```
1  def predictHorizonSteps(valData, preds, model, horizon):
2      for _ in range(horizon):
3          firstRow = //Take the first row from valData//
4          prediction = model.predict(firstRow)
5          preds = //Add prediction to preds//
6
7          valData = //Add prediction to the lag-columns of valData//  # This allows the
                       model to use the prediction to predict future values
8
9          valData = //Delete firstRow from valData//
10
11     return (valData, preds)
12
13 def forecast(windowStrategy, valData, target, model, horizon):
14     preds = []
15     while(length of valData is not 0):
16         valData, preds = predictHorizonSteps(valData, preds, model, horizon)
17
18         valData = //Where predictions were added (line 7); Replace the rows of the
                       lag-columns with the GT values from target//
19
20         target = //Delete the GT values that were added to the lag-columns
                              of the valData (line 18)//
21
22         # Reestimate model according to windowStrategy
23         model = windowStrategy(valData, model, horizon)  # Fixed, Rolling or Expanding
24
25     return preds
```

8.2 Hyperparameter search without difference

				Training			Validation		
line	level	diff	model	nMAE	MAPE	nRMSE	nMAE	MAPE	nRMSE
1A	60	d=0	SNaive	0.170	0.229	0.278	0.134	0.150	0.193
			F-Lasso-W	0.441	1.184	0.552	0.422	1.523	0.541
			F-Lasso-C	0.357	0.976	0.439	0.362	1.191	0.445
			F-Lasso-CW	0.313	0.784	0.394	0.333	0.763	0.419
			F-Lasso-L	0.161	0.261	0.238	0.138	0.239	0.190
			F-Lasso-LW	0.161	0.258	0.237	0.137	0.247	0.188
			F-Lasso-LC	0.147	0.256	0.204	0.128	0.236	0.174
			F-Lasso-LCW	0.146	0.255	0.204	0.126	0.233	0.172
			F-RF-W	0.128	0.325	0.170	0.423	1.415	0.548
			F-RF-C	0.339	0.902	0.417	0.351	1.171	0.434
			F-RF-CW	0.086	0.201	0.115	0.295	0.590	0.389
			F-RF-L	0.048	0.071	0.073	0.116	0.144	0.171
			F-RF-LW	0.047	0.070	0.070	0.115	0.144	0.168
			F-RF-LC	0.040	0.054	0.057	0.101	0.121	0.148
			F-RF-LCW	0.038	0.054	0.055	0.100	0.121	0.146
			F-XGB-W	0.389	1.037	0.494	0.409	1.451	0.526
			F-XGB-C	0.342	0.942	0.422	0.350	1.249	0.436
			F-XGB-CW	0.258	0.650	0.329	0.295	0.746	0.384
			F-XGB-L	0.110	0.168	0.155	0.118	0.163	0.171
			F-XGB-LW	0.114	0.183	0.160	0.121	0.187	0.172
			F-XGB-LC	0.089	0.129	0.122	0.099	0.124	0.143
			F-XGB-LCW	0.096	0.152	0.130	0.099	0.144	0.143
			F-RNN-W	0.421	1.119	0.529	0.411	1.530	0.525
			F-RNN-C	0.351	0.888	0.433	0.345	1.178	0.419
			F-RNN-CW	0.301	0.732	0.379	0.308	0.870	0.398
			F-RNN-L	0.130	0.195	0.192	0.116	0.164	0.167
			F-RNN-LW	0.141	0.207	0.208	0.123	0.173	0.172
			F-RNN-LC	0.101	0.143	0.139	0.097	0.130	0.140
			F-RNN-LCW	0.107	0.160	0.147	0.102	0.131	0.144
			F-LSTM-W	0.419	1.110	0.526	0.414	1.538	0.534
			F-LSTM-C	0.347	0.940	0.426	0.338	1.288	0.410
			F-LSTM-CW	0.275	0.654	0.350	0.286	0.771	0.367
			F-LSTM-L	0.120	0.179	0.175	0.115	0.161	0.167
			F-LSTM-LW	0.127	0.196	0.182	0.118	0.187	0.167
			F-LSTM-LC	0.091	0.118	0.126	0.096	0.115	0.137
			F-LSTM-LCW	0.090	0.129	0.121	0.101	0.120	0.148

Table 6: Line 1A 60min non differenced grid search results

F denotes fixed window, L denotes including lags, C denotes including calendar variables and W denotes including weather variables.

8.3 Hyperparameter search with difference

				Training			Validation		
line	level	diff	model	nMAE	MAPE	nRMSE	nMAE	MAPE	nRMSE
1A	60	d=1	SNaive	0.207	0.312	0.314	0.148	0.210	0.203
			F-Lasso-W	0.690	1.009	0.881	0.660	0.155	0.847
			F-Lasso-C	0.597	0.855	0.772	0.601	1.106	0.782
			F-Lasso-CW	0.603	0.974	0.775	0.587	1.263	0.770
			F-Lasso-L	0.226	0.347	0.324	0.187	0.273	0.253
			F-Lasso-LW	0.227	0.348	0.325	0.187	0.274	0.251
			F-Lasso-LC	0.212	0.351	0.300	0.172	0.312	0.241
			F-Lasso-LCW	0.212	0.351	0.300	0.172	0.312	0.241
			F-RF-W	0.249	0.435	0.324	0.729	1.436	0.938
			F-RF-C	0.598	0.958	0.783	0.558	1.321	0.733
			F-RF-CW	0.236	0.393	0.317	0.648	1.182	0.848
			F-RF-L	0.099	0.169	0.134	0.216	0.352	0.285
			F-RF-LW	0.096	0.167	0.131	0.188	0.311	0.253
			F-RF-LC	0.092	0.170	0.125	0.179	0.335	0.236
			F-RF-LCW	0.084	0.153	0.114	0.175	0.291	0.236
			F-XGB-W	0.630	0.971	0.806	0.638	1.228	0.850
			F-XGB-C	0.595	0.924	0.781	0.562	1.259	0.747
			F-XGB-CW	0.533	0.882	0.700	0.605	1.361	0.803
			F-XGB-L	0.213	0.356	0.288	0.194	0.366	0.254
			F-XGB-LW	0.206	0.355	0.277	0.194	0.368	0.257
			F-XGB-LC	0.192	0.316	0.260	0.158	0.345	0.210
			F-XGB-LCW	0.198	0.339	0.263	0.168	0.303	0.223
			F-RNN-W	0.657	1.035	0.839	0.636	1.125	0.831
			F-RNN-C	0.564	1.095	0.722	0.549	1.416	0.730
			F-RNN-CW	0.626	0.998	0.802	0.538	1.081	0.697
			F-RNN-L	0.225	0.353	0.325	0.194	0.268	0.260
			F-RNN-LW	0.241	0.400	0.347	0.192	0.304	0.258
			F-RNN-LC	0.237	0.410	0.314	0.180	0.348	0.248
			F-RNN-LCW	0.251	0.422	0.344	0.184	0.376	0.245
			F-LSTM-W	0.670	1.031	0.859	0.596	1.261	0.778
			F-LSTM-C	0.588	0.874	0.767	0.587	1.140	0.779
			F-LSTM-CW	0.692	1.040	0.884	0.587	1.284	0.771
			F-LSTM-L	0.224	0.401	0.304	0.181	0.338	0.240
			F-LSTM-LW	0.241	0.396	0.343	0.194	0.359	0.256
			F-LSTM-LC	0.193	0.319	0.257	0.183	0.319	0.245
			F-LSTM-LCW	0.223	0.367	0.296	0.197	0.316	0.259

Table 7: Line 1A 60min first-differenced grid search results

8.4 Models from search on all lines (60min agg. level)

				Training			Validation		
model	level	diff	line	nMAE	MAPE	nRMSE	nMAE	MAPE	nRMSE
SNaive	60	d=0	1A	0.170	0.229	0.278	0.134	0.150	0.193
			2A	0.161	0.288	0.270	0.122	0.141	0.181
			3A	0.171	0.252	0.295	0.142	0.168	0.207
			4A	0.161	0.316	0.290	0.123	0.156	0.180
			5A	0.230	0.334	0.395	0.183	0.225	0.274
			6A	0.199	0.256	0.427	0.139	0.167	0.248
F-Lasso-LCW	60	d=0	1A	0.146	0.255	0.204	0.126	0.233	0.172
			2A	0.138	0.375	0.193	0.115	0.290	0.162
			3A	0.148	0.340	0.213	0.138	0.275	0.193
			4A	0.141	0.394	0.212	0.127	0.339	0.174
			5A	0.200	0.457	0.295	0.181	0.364	0.260
			6A	0.194	0.397	0.325	0.172	0.361	0.270
F-RF-LCW	60	d=0	1A	0.038	0.054	0.054	0.100	0.120	0.146
			2A	0.036	0.062	0.051	0.093	0.111	0.141
			3A	0.038	0.057	0.054	0.108	0.138	0.158
			4A	0.033	0.070	0.050	0.093	**0.125**	0.137
			5A	0.050	0.080	0.077	0.136	**0.179**	**0.206**
			6A	0.038	0.052	0.067	0.105	**0.130**	0.185
F-XGB-LC	60	d=0	1A	0.089	0.125	0.123	0.099	0.123	0.143
			2A	0.083	0.175	0.114	0.091	0.124	0.136
			3A	0.087	0.147	0.121	0.105	0.136	0.152
			4A	0.076	0.216	0.108	0.090	0.144	0.129
			5A	0.115	0.211	0.169	**0.135**	0.185	0.207
			6A	0.087	0.142	0.134	0.104	0.142	0.174
F-RNN-LC	60	d=0	1A	0.101	0.142	0.139	0.098	0.125	0.140
			2A	0.093	0.201	0.129	0.090	0.145	0.133
			3A	0.096	0.167	0.136	0.103	0.140	**0.148**
			4A	0.086	0.230	0.123	0.091	0.170	0.127
			5A	0.135	0.237	0.202	0.138	0.209	0.208
			6A	0.103	0.163	0.170	0.104	0.154	0.165
F-LSTM-LC	60	d=0	1A	0.091	0.118	0.126	**0.096**	**0.115**	**0.137**
			2A	0.087	0.143	0.122	**0.087**	**0.106**	**0.131**
			3A	0.090	0.140	0.127	**0.102**	**0.130**	**0.148**
			4A	0.079	0.184	0.112	**0.087**	0.165	**0.122**
			5A	0.128	0.216	0.189	0.137	0.185	0.207
			6A	0.094	0.139	0.150	**0.099**	0.132	**0.162**

Table 8: Models using best hyperparameters from hyperparameter search on the 60-minute aggregation level of training and validation data.

8.5 Models from search on all lines (30min agg. level)

				Training			Validation		
model	level	diff	line	nMAE	MAPE	nRMSE	nMAE	MAPE	nRMSE
SNaive	30	d=0	1A	0.192	0.268	0.301	0.153	0.199	0.216
			2A	0.183	0.278	0.293	0.142	0.186	0.207
			3A	0.198	0.276	0.323	0.166	0.201	0.241
			4A	0.181	0.271	0.309	0.142	0.189	0.204
			5A	0.255	0.367	0.429	0.208	0.258	0.311
			6A	0.221	0.297	0.454	0.162	0.208	0.282
F-Lasso-LCW	30	d=0	1A	0.163	0.439	0.225	0.141	0.375	0.192
			2A	0.155	0.412	0.215	0.132	0.334	0.183
			3A	0.168	0.317	0.238	0.158	0.263	0.220
			4A	0.157	0.346	0.230	0.142	0.343	0.195
			5A	0.219	0.421	0.328	0.202	0.338	0.292
			6A	0.208	0.463	0.351	0.186	0.405	0.304
F-RF-LCW	30	d=0	1A	0.044	0.071	0.062	**0.116**	**0.168**	0.166
			2A	0.042	0.074	0.059	0.112	0.167	0.163
			3A	0.045	0.066	0.065	0.129	0.163	0.186
			4A	0.039	0.063	0.056	0.108	0.164	0.153
			5A	0.057	0.089	0.087	**0.158**	**0.205**	**0.241**
			6A	0.045	0.065	0.076	**0.124**	**0.165**	0.214
F-XGB-LC	30	d=0	1A	0.111	0.188	0.153	**0.116**	0.182	**0.164**
			2A	0.105	0.199	0.144	**0.108**	**0.162**	**0.157**
			3A	0.112	0.175	0.157	**0.126**	**0.160**	**0.181**
			4A	0.098	0.177	0.137	**0.107**	**0.162**	**0.149**
			5A	0.144	0.233	0.211	0.160	**0.205**	0.245
			6A	0.112	0.182	0.174	0.125	0.170	**0.210**
F-RNN-LC	30	d=0	1A	0.142	0.275	0.194	0.137	0.262	0.190
			2A	0.136	0.301	0.186	0.132	0.262	0.186
			3A	0.141	0.258	0.198	0.147	0.231	0.205
			4A	0.127	0.302	0.180	0.128	0.266	0.177
			5A	0.182	0.342	0.267	0.186	0.290	0.270
			6A	0.152	0.287	0.250	0.151	0.296	0.236
F-LSTM-LC	30	d=0	1A	0.135	0.228	0.187	0.132	0.213	0.185
			2A	0.128	0.228	0.179	0.129	0.201	0.185
			3A	0.135	0.220	0.192	0.142	0.194	0.202
			4A	0.120	0.213	0.174	0.122	0.186	0.171
			5A	0.173	0.300	0.256	0.187	0.274	0.277
			6A	0.141	0.245	0.230	0.145	0.251	0.231

Table 9: Models using best hyperparameters from hyperparameter search on the 30-minute aggregation level of training and validation data.

8.6 Models from search on all lines (15min agg. level)

				Training			Validation		
model	level	diff	line	nMAE	MAPE	nRMSE	nMAE	MAPE	nRMSE
SNaive	15	d=0	1A	0.233	0.304	0.335	0.183	0.229	0.255
			2A	0.217	0.298	0.328	0.176	0.220	0.251
			3A	0.238	0.327	0.368	0.206	0.261	0.295
			4A	0.211	0.292	0.341	0.175	0.227	0.247
			5A	0.297	0.445	0.479	0.249	0.332	0.370
			6A	0.254	0.354	0.489	0.197	0.255	0.326
F-Lasso-LCW	15	d=0	1A	0.187	0.361	0.254	0.165	0.304	0.223
			2A	0.180	0.337	0.245	0.158	0.277	0.217
			3A	0.199	0.345	0.278	0.188	0.304	0.263
			4A	0.180	0.342	0.259	0.167	0.309	0.230
			5A	0.251	0.469	0.370	0.233	0.373	0.340
			6A	0.231	0.456	0.383	0.210	0.397	0.339
F-RF-LCW	15	d=0	1A	0.053	0.078	0.074	**0.140**	0.196	0.197
			2A	0.052	0.076	0.072	0.137	0.180	0.197
			3A	0.057	0.083	0.081	0.160	0.218	0.231
			4A	0.048	0.070	0.069	**0.134**	0.187	0.189
			5A	0.069	0.113	0.103	**0.191**	0.274	**0.289**
			6A	0.055	0.080	0.088	0.151	0.205	0.250
F-XGB-LC	15	d=0	1A	0.141	0.212	0.193	**0.140**	**0.190**	**0.195**
			2A	0.136	0.202	0.186	**0.134**	**0.174**	**0.191**
			3A	0.149	0.220	0.208	**0.158**	**0.216**	**0.227**
			4A	0.127	0.189	0.177	**0.134**	**0.184**	0.187
			5A	0.183	0.302	0.268	0.192	**0.272**	0.292
			6A	0.144	0.219	0.224	**0.150**	**0.202**	**0.245**
F-RNN-LC	15	d=0	1A	0.248	0.574	0.329	0.236	0.495	0.315
			2A	0.229	0.484	0.304	0.222	0.428	0.298
			3A	0.247	0.453	0.334	0.247	0.415	0.339
			4A	0.232	0.497	0.319	0.227	0.435	0.312
			5A	0.289	0.539	0.418	0.296	0.458	0.448
			6A	0.280	0.562	0.452	0.277	0.488	0.464
F-LSTM-LC	15	d=0	1A	0.246	0.554	0.327	0.234	0.481	0.311
			2A	0.228	0.471	0.303	0.219	0.427	0.295
			3A	0.244	0.449	0.332	0.245	0.423	0.338
			4A	0.230	0.476	0.317	0.225	0.427	0.310
			5A	0.288	0.537	0.417	0.294	0.467	0.444
			6A	0.276	0.537	0.450	0.276	0.489	0.461

Table 10: Models using best hyperparameters from hyperparameter search on the 15-minute aggregation level of training and validation data.

	1A					2A				
† / *	SN	Las	RF	XGB	RNN	SN	Las	RF	XGB	RNN
Las	8e-10†	-	-	-	-	6e-10†	-	-	-	-
RF	3e-16†	1e-23†	-	-	-	5e-18†	8e-27†	-	-	-
XGB	4e-17†	1e-27†	0.003†	-	-	6e-18†	4e-29†	0.547	-	-
RNN	4e-17†	3e-24†	5e-04†	0.169	-	2e-18†	7e-30†	0.060	0.053	-
LSTM	2e-16†	1e-22†	0.141†	0.456	0.041*	1e-16†	8e-24†	0.250	0.055	0.005*
	3A					4A				
† / *	SN	Las	RF	XGB	RNN	SN	Las	RF	XGB	RNN
Las	1e-10†	-	-	-	-	5e-08†	-	-	-	-
RF	2e-18†	2e-24†	-	-	-	7e-15†	2e-22†	-	-	-
XGB	5e-19†	2e-28†	0.194	-	-	1e-15†	2e-24†	0.003†	-	-
RNN	5e-19†	5e-30†	0.523	0.748	-	2e-15†	2e-25†	0.079	0.459	-
LSTM	1e-18†	1e-20†	0.065	0.004*	0.031*	1e-15†	7e-22†	0.021†	0.371	0.322
	5A					6A				
† / *	SN	Las	RF	XGB	RNN	SN	Las	RF	XGB	RNN
Las	2e-12†	-	-	-	-	1e-07†	-	-	-	-
RF	1e-23†	3e-26†	-	-	-	9e-14†	1e-14†	-	-	-
XGB	2e-22†	4e-23†	0.189	-	-	4e-14†	3e-15†	0.152	-	-
RNN	1e-22†	2e-27†	0.028*	0.420	-	9e-14†	3e-19†	0.090	0.053	-
LSTM	7e-22†	3e-24†	0.046*	0.317	0.808	3e-14†	2e-20†	0.248	0.167	0.016†

Table 11: P-values from Diebold Mariano test on forecasts from the test data on 60-minute aggregation level. Significane level set at $\alpha = 0.05$ and the best model, in case of significant P-value, is determined using MAE.

8.7 Feature Importance

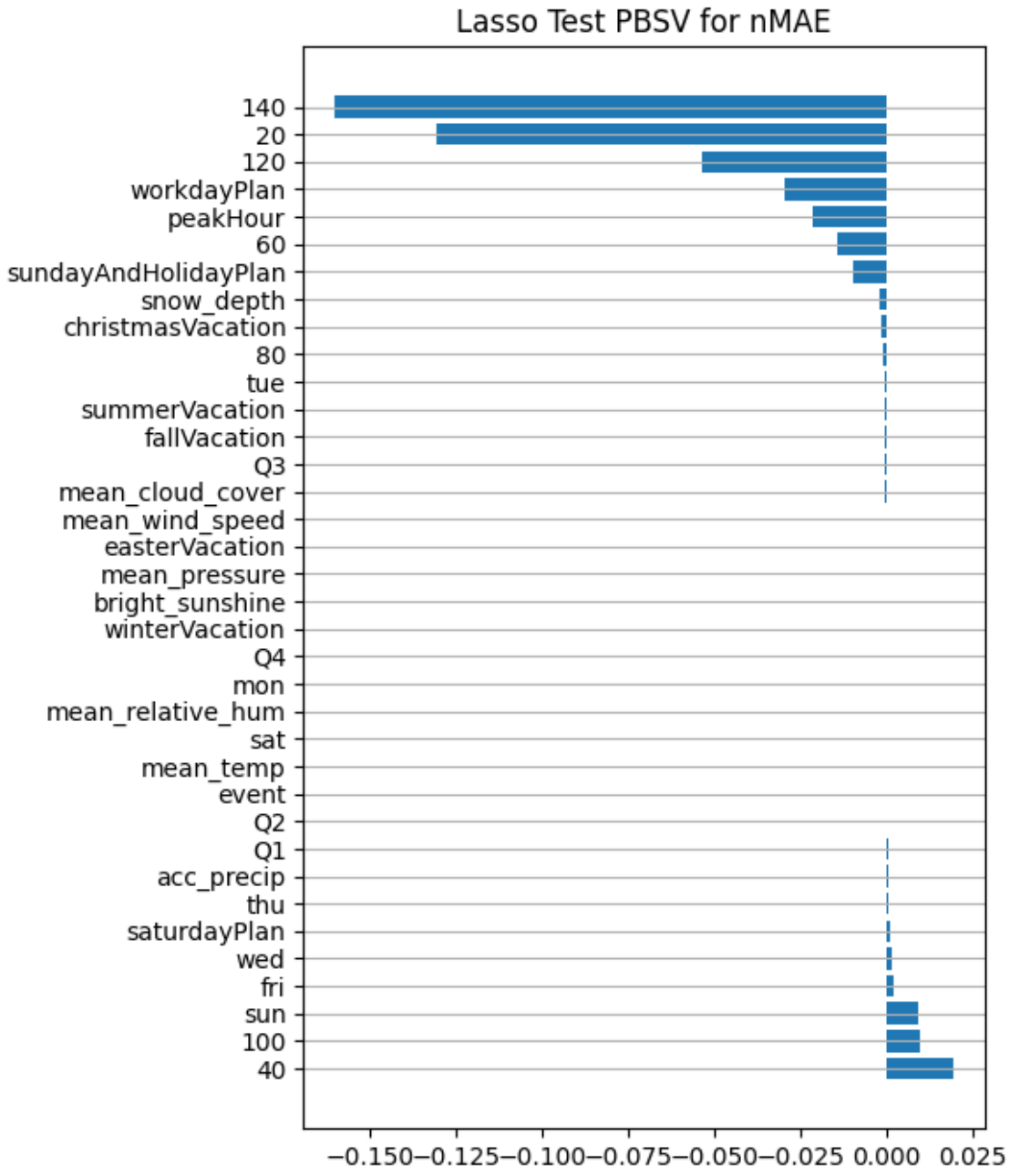

Figure 22: PBSV feature importance in terms of nMAE reduction for Random Forest on test data using daily lags and all calendar and all weather variables.

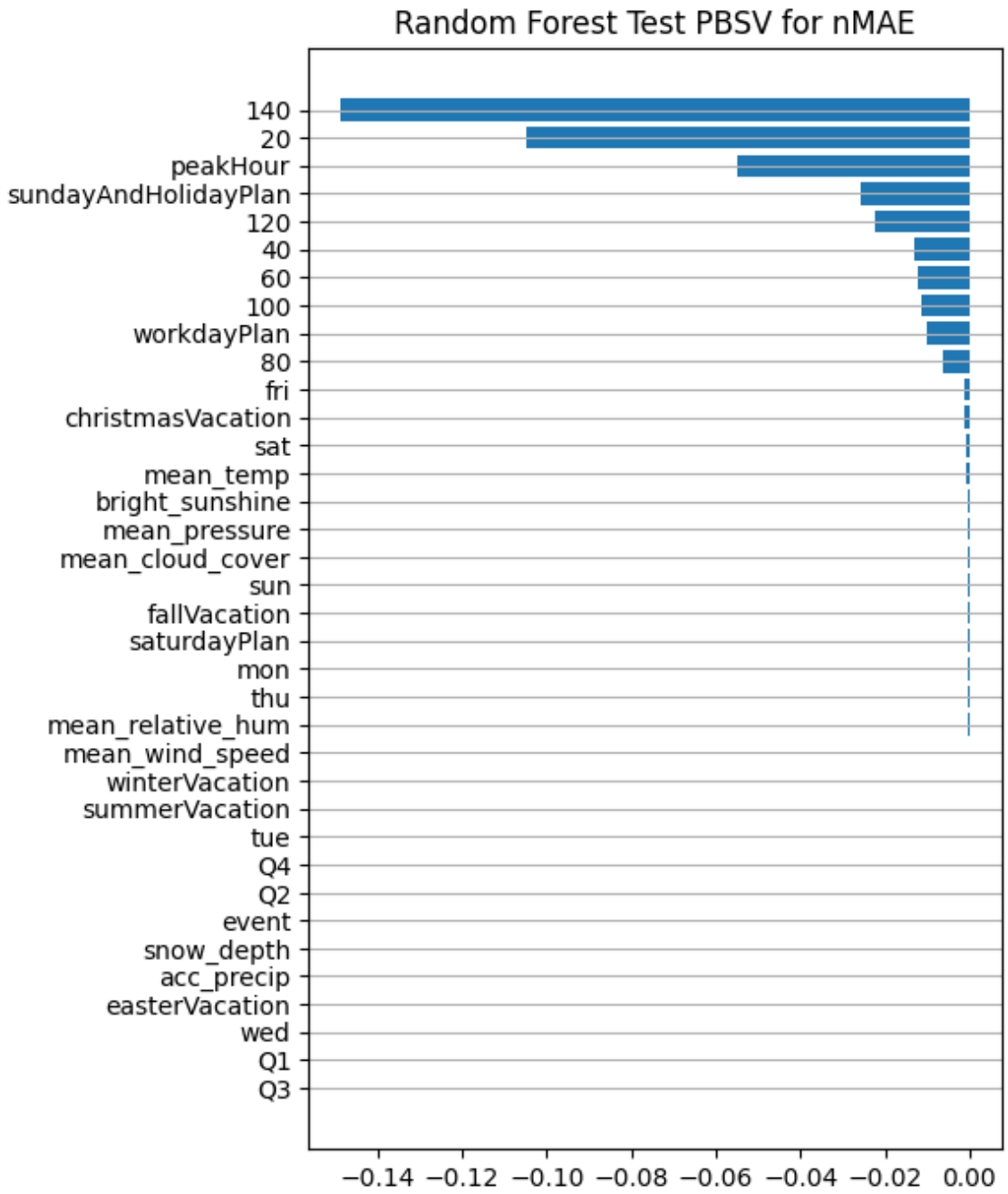

Figure 23: PBSV feature importance in terms of nMAE reduction for Random Forest on test data using daily lags and all calendar and all weather variables.

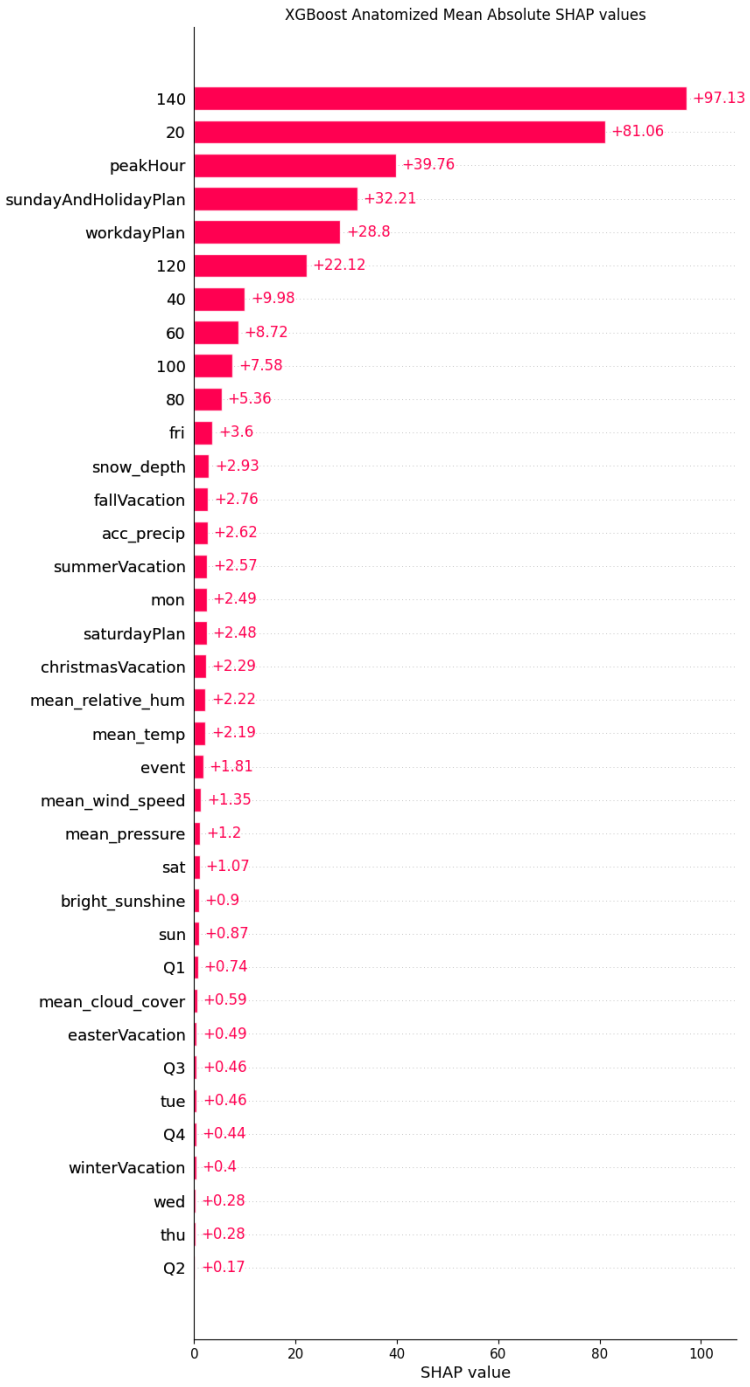

Figure 24: Mean absolute value of anatomized SHAP values for XGBoost on the test data.

Deklaration af Generative Artificial Intelligence (GAI) i projekter

Brug af GAI i projekter

I henhold til de nye retningslinjer på Aarhus Universitet omkring brug af Generative Artificial Intelligence (GAI) skal du inkludere en deklaration som bilag, hvis du har brugt GAI (som f.eks. ChatGPT, Microsoft Copilot (Bing), Google Gemini, custom GPTs etc..) i forbindelse med udarbejdelsen af dit projekt.

Eksamensopgaver bedømmes ud fra læringsmålene i kursusbeskrivelsen, og derfor vil brugen af GAI ikke i sig selv have indflydelse på bedømmelsen, forudsat at reglerne for brug af GAI overholdes.

Husk altid at undersøge reglerne for brug af GAI på studerende.au.dk.

Sådan udfylder du deklarationen

I din deklaration om brugen af GAI, skal du som minimum:

1. Tilkendegive at du har brugt GAI.
2. Specificere hvilken teknologi, der er anvendt, med angivelse af versionsnummer (fx ChatGPT 3.5 eller Llama 2).
3. Beskrive hvordan information er blevet genereret, beskrive de anvendte input, samt forklare hvordan outputtet blev anvendt i din projektrapport – se vejledning på Studypedia
4. Efter aftale med din vejleder kan du eventuelt også vedlægge et mere detaljeret bilag med prompts og de output, der er genereret.

Eksempler på brug af GAI

Nedenfor er oplistet en række eksempler på anvendelsesmåder af GAI i dit projekt. Listen er ikke udtømmende, og du må gerne angive andre måder, du har anvendt GAI på.
I deklarationen skal du både angive, hvilke værktøjer, du konkret har anvendt, samt beskrive kortfattet hvordan og med hvilket udbytte, du har anvendt dem hver især.

- Til at søge eller strukturere information
- Til programmeringsopgaver
- Til dataanalyse
- Til fremstilling af figurer
- Til at formatere tekst/formler i din opgave
- Til at få feedback/forbedringer/korrektur på tekst og formuleringer
- Til at generere tekst anvendt direkte i opgaven (vigtigt, se nedenfor)
    - Hvis du har omskrevet tekster, der er genereret af GAI, men ikke direkte citeret, skal du deklarere, hvordan du har brugt generativ AI.
    - Hvis du har citeret direkte fra tekst, der er genereret af GAI, skal det angives som citat med en reference til det GAI -værktøj, du har benyttet – se vejledning her AU Library

Følgende skema anvendes som deklaration (og udfyld gerne skemaet i samarbejde med din vejleder):

Deklaration af Generativ Kunstig Intelligens (GAI) i projekter

Navn: Andreas Skiby Andersen, Andreas Hyldegaard Hansen

Studienummer: 202107332, 202106123

Projekttype: Bachelorprojekt

Projekttitel: From Black-Box to Explainable Insights: Forecasting Passenger Demand using APC Data.
Fra Black-Box til indsigter: Prædiktion af passagerer ved brug af APC data

✔ **Jeg har benyttet generativ kunstig intelligens til udfærdigelse af dette projekt** *(sæt kryds)*

Beskriv hvilke GAI-værktøjer, du har benyttet (husk version):

ChatGPT-3.5 og ChatGPT-4o.

Jeg har brugt GAI på følgende vis:

Nogle mulige anvendelsesområder er angivet som inspiration – slet eller tilføj efter behov. For hvert relevant område forklares, hvordan GAI er benyttet. Beskriv f.eks. kortfattet hvordan informationen blev genereret, samt forklar hvordan outputtet er anvendt i din projektrapport.

- *til at søge eller strukturere information*
- *til programmeringsopgaver*
- *til dataanalyse*
- *til fremstilling af figurer*
- *til at formatere tekst/formler i din opgave*
- *til at få feedback/forbedringer/korrektur på tekst og formulering*
- *til at generere tekst anvendt direkte i opgaven*
- *andet*

- til programmeringsopgaver og figurer:
Hurtig og effektiv måde at huske kode syntax f.eks. til dataprocessering i Pandas og til at lave et flot heatmap.

- til at formatere tekst:
Hjælp til at oversætte specifikke ord, finde passende synonymer og inspiration når en kompliceret idé i vores hoved skulle skrives ned på en simpel og kortfattet måde.

- til at få feedback/forbedringer/korrektur på tekst og formuleringer:
Konkret feedback til specifikke sætninger og overordnet feedback/forbedringsforslag til større paragraffer og rapporten som helhed.
Ingen større tekststykker er direkte kopieret fra GAI.