Petelin *et al.*
European Transport Research Review (2023) 15:30
https://doi.org/10.1186/s12544-023-00600-6

European Transport Research Review

ORIGINAL PAPER

Open Access

Models for forecasting the traffic flow within the city of Ljubljana

Gašper Petelin[1,2*], Rok Hribar[1,2] and Gregor Papa[1,2]

Abstract

Efficient traffic management is essential in modern urban areas. The development of intelligent traffic flow prediction systems can help to reduce travel times and maximize road capacity utilization. However, accurately modeling complex spatiotemporal dependencies can be a difficult task, especially when real-time data collection is not possible. This study aims to tackle this challenge by proposing a solution that incorporates extensive feature engineering to combine historical traffic patterns with covariates such as weather data and public holidays. The proposed approach is assessed using a new real-world data set of traffic patterns collected in Ljubljana, Slovenia. The constructed models are evaluated for their accuracy and hyperparameter sensitivity, providing insights into their performance. By providing practical solutions for real-world scenarios, the proposed approach offers an effective means to improve traffic flow prediction without relying on real-time data.

Keywords Traffic modeling, Time-series forecasting, Traffic-count data set, Machine learning, Model comparison

1 Introduction

Traffic estimation and prediction systems tend to improve traffic conditions and reduce travel delays while facilitating better utilization of available capacity [1, 2]. Travel information affects users' route choice behavior by directing travelers to less congested routes and reducing congestion [3]. Traffic congestion is one of the most demanding transport problems in large urban environments [4], and therefore traffic flow prediction plays an essential role in today's transportation systems. Accurate predictions of traffic flow can assist with route planning, guide vehicle dispatching, and mitigate traffic congestion [5]. This problem is challenging due to the complicated and dynamic spatiotemporal dependencies between different regions in the road network. In general, traffic flow prediction is the task of making accurate predictions of flow of vehicles within a specific area at a particular future time interval based on historical traffic data. Successful traffic management is crucial in most modern urban areas. Therefore, predicting traffic flow and traffic patterns can provide some insight to traffic managers and is important to allow for the smooth flow of traffic and minimize congestion. In addition, the use of up-to-date traffic information can lead to a more reliable system as it helps to reduce the volatility of travel costs compared to a decision-making approach without information [6].

Many cities collect traffic data by various means, such as sensors and cameras. However, the availability of this data in real-time is often limited due to technical and logistical challenges. This severely limits the capabilities of existing traffic flow modeling methods that rely on up-to-date real-time data for prediction modeling. Most existing modeling methods require access to the latest traffic conditions that can be ingested into the model in real-time to produce new, up-to-date predictions. When this is not possible, finding alternative solutions that can accurately predict traffic flow without relying on real-time data is crucial for effective traffic management in

*Correspondence:
Gašper Petelin
gasper.petelin@ijs.si
[1] Computer Systems Department, Jožef Stefan Institute, Jamova cesta 39, Ljubljana, Slovenia
[2] Jožef Stefan International Postgraduate School, Jamova cesta 39, Ljubljana, Slovenia

urban areas where infrastructure does not support automated data collection.

The main objective of this paper is to propose a solution to the challenge of accurately predicting traffic flow in cities where real-time data collection is not feasible. In such scenarios, traffic flow prediction models cannot rely on the so-called "lag features" that capture the relationship between the current traffic flow and its past values. Therefore, the objective of this paper is to develop models that can accurately predict traffic flow without relying on the latest real-time data and with no lag features. To achieve this, we propose to mitigate some of the limitations through extensive feature engineering that does not require real-time traffic data. By developing models that do not rely on real-time data and using feature engineering techniques, we aim to develop practical solutions that can be used in real-world scenarios to improve traffic flow prediction.

Our contribution We introduce a new real-world data set of traffic patterns collected between the years 2013 and 2020 in Ljubljana, Slovenia. Using this data set, we construct models that, taking into account the data acquisition constraints, can combine historical traffic patterns with covariates such as weather data and public holidays to obtain accurate and robust forecasts of traffic patterns. The obtained models are further analyzed for their accuracy and hyperparameter sensitivity. We provide insight into when the models perform well and in which scenarios they become less reliable.

The rest of the paper is organized as follows. In Sect. 2 a related literature review is presented. In Sect. 3 we introduce a newly acquired traffic flow data set and its properties. In Sect. 4 we outline the problem, describe the experimental methodology for fair model comparison, and outline the machine-learning methods applied to the problem. In Sect. 5, we present the results of the data analysis and model comparison, accompanied by a related discussion. Limitations are exposed in Sect. 6, and concluding remarks are given in Sect. 7, followed by future directions of work.

2 Related work

The problem of traffic modeling has been extensively studied in the literature, and a wide range of approaches have been proposed to address this complex issue. It is a critical component of intelligent transportation systems and has been the subject of extensive research efforts. The main goal of traffic modeling is to accurately analyze various traffic characteristics, such as traffic flow, density, and speed, and leverage these properties and their interactions to predict future traffic trends. The ultimate goal is to improve the efficiency of the traffic network, support the planning and design of road facilities, and optimize traffic operations.

Our paper primarily studies traffic flow using historical data, taking into account the complex temporal and spatial dependencies involved in traffic forecasting [7, 8]. Temporal dependencies relate to periodic trends, such as rush hours or seasonal changes, while spatial dependencies describe how changes in traffic on one road may affect adjacent roads due to the topological structure of the road network. Successful modeling of traffic patterns requires the inclusion of both types of dependencies along with various covariates to improve the predictive power of algorithms. It has also been shown that multi-target models that capture dependencies between different targets and transfer information between them can also improve generalization [9, 10].

In the past, the need for accurate modeling and prediction of traffic flows encouraged the development of many different machine-learning approaches. The field of statistics, which focuses on univariate time series, offers a wide range of models such as the Moving Average (MA), the Auto-Regressive (AR), and the Auto-Regressive Integrated Moving Average (ARIMA) [11]. These models can perform extremely well when only a small amount of data is available. Many of these approaches were later extended to handle multivariate data and covariates, such as VARIMA [12], ARMAX, and ARIMAX [13], and were successfully applied for traffic modeling [14].

Given the ever-increasing amount of data, traditional statistical approaches are sometimes insufficient to effectively model complex time-dependent interactions [15, 16]. As a result, recent methodologies are increasingly turning towards more complex machine learning (ML) models. However, drawing a clear line of demarcation between statistical models and ML-based models is often vague and poorly-defined [17]. In our study, we establish a categorization scheme where methods are classified as statistical if they involve explicitly specifying the data-generating process. On the other hand, methods are considered ML-based if they allow learning data relationships.

The first approaches used classical machine learning methods where temporal dependencies were added by including lag features and treating problems as tabular problems [18–20]. Further improvements in accuracy have been achieved by incorporating more complex neural network-based models, such as RNN [21], where connections between neurons can be organized in a cycle. Such models are better suited for handling temporal dependencies and have been successfully used for modeling traffic [22, 23]. Later improvements in modeling temporal data, such as the introduction of LSTM cells [24], were also quickly adopted for the field of traffic

forecasting [25]. Almost in parallel with the advances in temporal modeling with LSTMs, convolutional neural networks (CNNs) [26] also became increasingly popular. Although originally developed for image classification, they were adapted for time series modelling [27] and successfully used for traffic forecasting [28–30]. More recently, there has been a focus on developing architectures that are more specialized for time series modeling. One of the popular approaches for univariate time series point forecasting is the N- BEATS [31] architecture, which is well suited for forecasting problems where large amounts of data are available. Similarly, DeepAR [32] is a popular forecasting neural network that uses LSTM cells to predict parameters of a probabilistic distribution and provides more information about the uncertainty of the model. It can handle multivariate time series with future and past covariates. Lately, transformer-based [33] neural networks such as Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting [34], have also been used for predicting freeway traffic speed [35]. Although deep learning techniques are widely used for forecasting, one should not ignore other approaches that have also been successfully used for accurate traffic forecasting [36, 37].

The previously mentioned related works focus on real-time data or continuous data acquisition, which limits their applicability to ideal scenarios. In the real world, where real-time data acquisition is not possible, alternative modeling techniques are needed. In this study, we attempt to develop techniques for such challenging scenarios where traditional modeling methods that rely on temporal components may not work. To our knowledge, no previous work has attempted to predict traffic flow without using any lag features.

With the ever-growing number of forecasting algorithms, it is crucial that they can be fairly compared in terms of performance. To achieve this, several data sets are available, with M Forecasting Competitions [38–40] being the most popular. If we focus specifically on traffic forecasting, two of the most popular data sets/benchmarks for evaluating and comparing traffic models are METR-LA and PEMS-BAY [28], which use loop detectors to collect traffic statistics. These benchmarks have been used extensively for model evaluation and benchmarking [41, 42]. With respect to our newly introduced data set, our data set covers a longer time period compared to the existing data sets.

3 Municipality of Ljubljana traffic data set (MOL-TR)

In this section, we describe the Municipality of Ljubljana traffic data set (MOL-TR) used in this study. The city of Ljubljana, Slovenia, is equipped with several traffic counting stations that are embedded in the roadway and detect vehicles as they pass over them. They are distributed throughout the city and are mainly located on roads/lanes with a high volume of vehicles. Over the years, the number of stations has changed from 57 at the beginning of 2013 to 47 at the end of 2020, mainly due to the reconstruction and merging. Figure 1 provides an overview of the locations within the city ring where the counters are placed.

The traffic recorder stores vehicle counts at 15-minute intervals (96 measurements per day for each individual measuring station). Vehicles that pass over the measuring station are assigned to one of 8 vehicle types (passenger car, motorcycle, bus, light/medium/heavy truck, semi-trailer truck, and tractor unit). The full data set contains 2,0410,686 vehicle count measurements collected at 59 stations during the 2013–2020 acquisition period. For a small glimpse of the data set, see Table 1. Each measurement instance contains an integer type variable `count` representing a count of vehicles of type `vehicle_type` that passed the measurement station with ID `station_ID` in the 15-minute time interval that ended at the time described by the `timestamp`.

To better understand traffic patterns, Fig. 2 shows the daily traffic distribution for the measuring station "1010-188" (note that other traffic measuring stations may have different patterns). The traffic volume usually follows a periodic pattern. Most noticeable are the daily changes, where there are two peaks during the morning and afternoon rush hour. Another pattern observed is a weekly periodicity, with less traffic on weekends compared to weekdays.

A similar breakdown is obtained when comparing vehicle types. Figure 3 breaks down the frequency of vehicles, with some types being more common than others. Regular passenger cars are the most frequent, about 460 times more than semi-trailer trucks. Vehicle types also have different distribution patterns, with passenger cars having distinct peaks in the morning and afternoon, while motorcycles typically only have a peak in the afternoon.

Missing data is a frequent occurrence in traffic modeling. If the measuring station fails, shuts down, or fails to collect data, there can be large gaps (even a few weeks or more) in data acquisition. Due to the design of measuring stations where two lanes are collected simultaneously, the failure of one measuring station usually means missing data for two lanes. Figure 4 shows the periods for which data are not available. For most periods between 2013 and 2020, the majority of stations are continuously operating and counting vehicles. The only exception is the year 2014, for which no data is available. There are a few stations that existed early on and were discontinued at some point, or stations that did not exist at the beginning

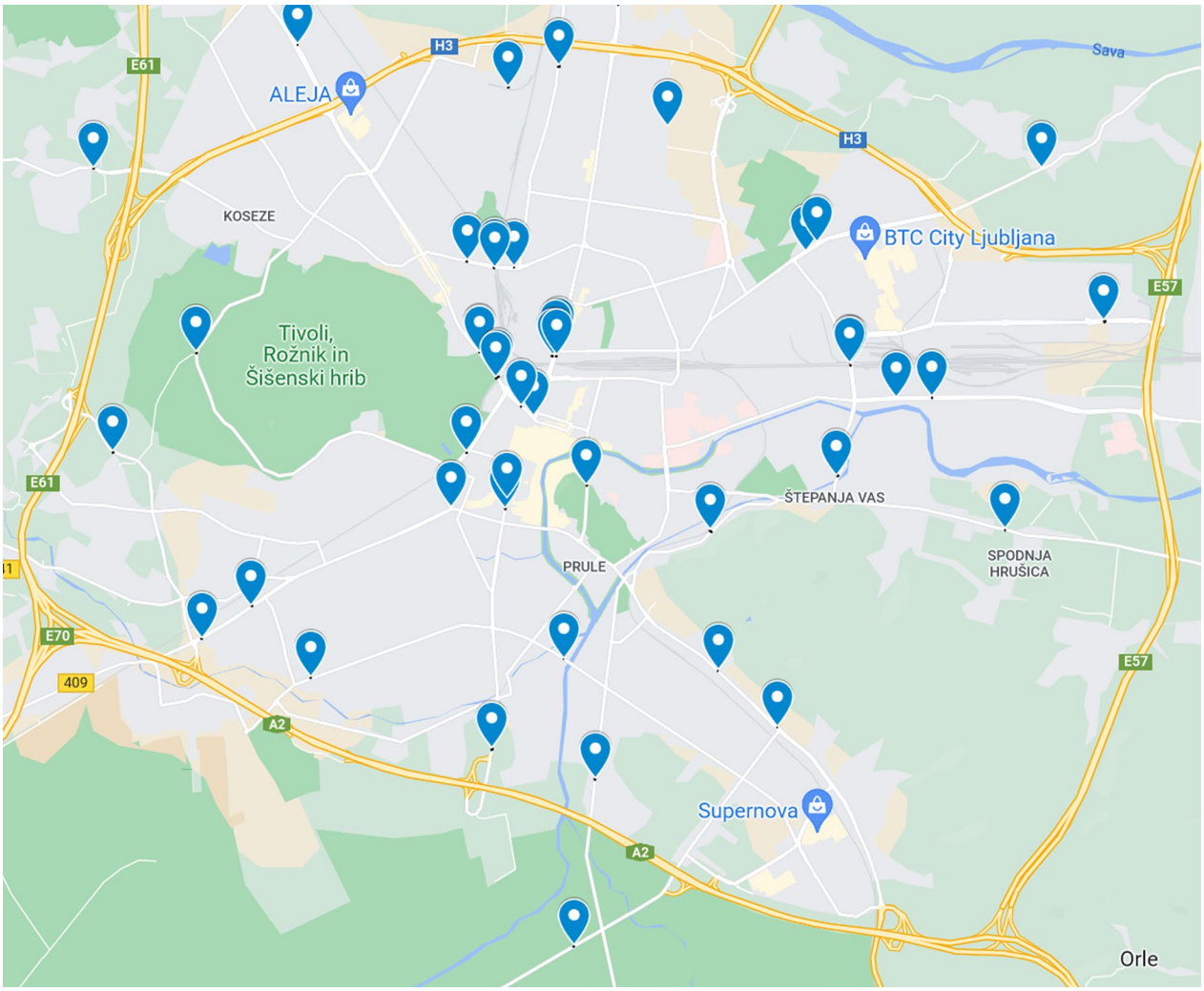

Fig. 1 Location of inductive loop traffic counters in the municipality of Ljubljana (MOL)

Table 1 A glimpse of the Ljubljana traffic flow data set shown as a four-column table with time variable `timestamp`, categorical variable `station_ID` of cardinality 59, categorical variable `vehicle_type` of cardinality 8 and integer type variable `count`

Timestamp	Station_ID	Vehicle_type	Count
⋮	⋮	⋮	⋮
2019-08-19 16:00	1019-186	Passenger car	95
2019-08-19 16:00	1019-186	Motorcycle	12
2019-08-19 16:15	1019-186	Passenger car	72
2019-08-19 16:15	1019-186	Motorcycle	16
2019-08-19 16:30	1019-186	Passenger car	57
2019-08-19 16:30	1019-186	Motorcycle	10
⋮	⋮	⋮	⋮

and became operational later. Taking measurements is also an error-prone process. Measurement stations may encounter problems such as reporting incorrect values due to a disturbance, or they may even stop reporting values if the disturbance is large enough or if a station is shut down for a period of time.

Relocations of measuring stations can cause some problems in predicting traffic. During the presented period, some of the measuring stations were temporarily or permanently relocated due to changes in road layouts and/or construction works. An example of this is transforming a specific lane that is only available for busses and prohibited for general traffic. This type of event completely changes the number of vehicles passing a measuring station and the distribution of traffic types.

Errors in the data acquisition process can cause problems in traffic modeling. Examples of such errors are values that cannot be stored due to an extremely

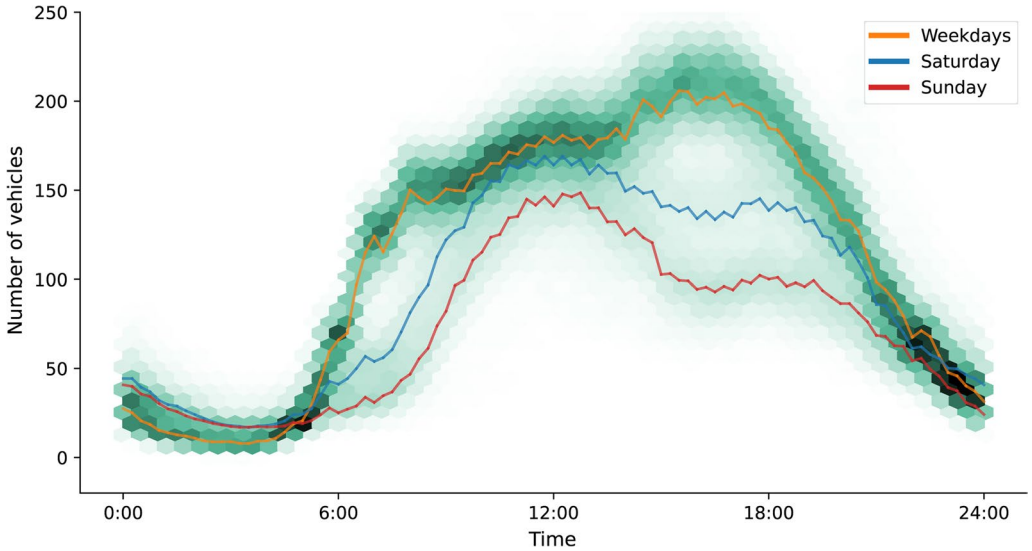

Fig. 2 Daily distribution in a number of vehicles for each 15-minute interval (green) for measuring station ID 1010-188. Additionally, colored lines show the mean traffic aggregated over the weekday, saturday, and sunday

high traffic volume (overflowing values). Such measurements are ignored and not considered in further modeling. Similarly, some measuring stations can automatically switch between standard and daylight saving time, while others cannot. This means that some measuring stations are effectively shifted in time by one hour. Unfortunately, it is not always clear which stations can automatically switch to daylight saving time.

Weather data is a factor that has to be considered when modeling traffic. Changes in weather can affect travel patterns, the types of vehicles used for transportation, and potential accidents. For this reason, weather information was added to the traffic data. The weather data consists of the daily average of the following: temperature, wind speed, cloud coverage, humidity, air pressure, precipitation, snowfall, sunshine, dew, glaze ice, and icy ground. While this daily resolution of coverage data may be too sparse for successful modeling of events such as sudden heavy rain that slows traffic, it can help capture impacts on a larger time scale. An example of such an event would be a multi-day snowstorm that can significantly alter traffic patterns.

Public holidays are another important factor affecting traffic patterns. During holidays, there is usually a significant decrease in traffic volume as well as a shift in traffic peaks. Additional data on public holidays is used, which includes all holidays between 2013 and 2020.

4 Methods

In this section, we first give a detailed description of the modeling framework and error metrics used to evaluate the prediction. Then we describe a method for making predictions with details on feature engineering and machine learning models.

4.1 Data acquisition process

In a problem where multiple predictions are required on related tasks, such as traffic flow prediction, selecting the right approach to accurately model traffic patterns is crucial. This section presents the modeling framework used for traffic forecasting. Figure 5 shows an overview of how data are collected and models are trained. Successfully collected measurements are stored in the measuring station. Since not all stations have an online connection to the traffic center, some stations have to be visited to manually transfer the data. This has a significant impact on how traffic modeling can be performed at this time, as the latest historical data patterns are not available. For example, when predicting traffic volumes, we cannot rely on traffic information from the most recent 15-minute interval.

Suppose there are three measuring stations where data acquisition occurs at time points $C1$, $C2$, and $C3$ for stations 1, 2, and 3, respectively. At that moment, we obtain all traffic data before the acquisition points. For some other stations, traffic data acquisition may occur at a different time point. This period of time when data is available and stored at a measuring station but has not yet been

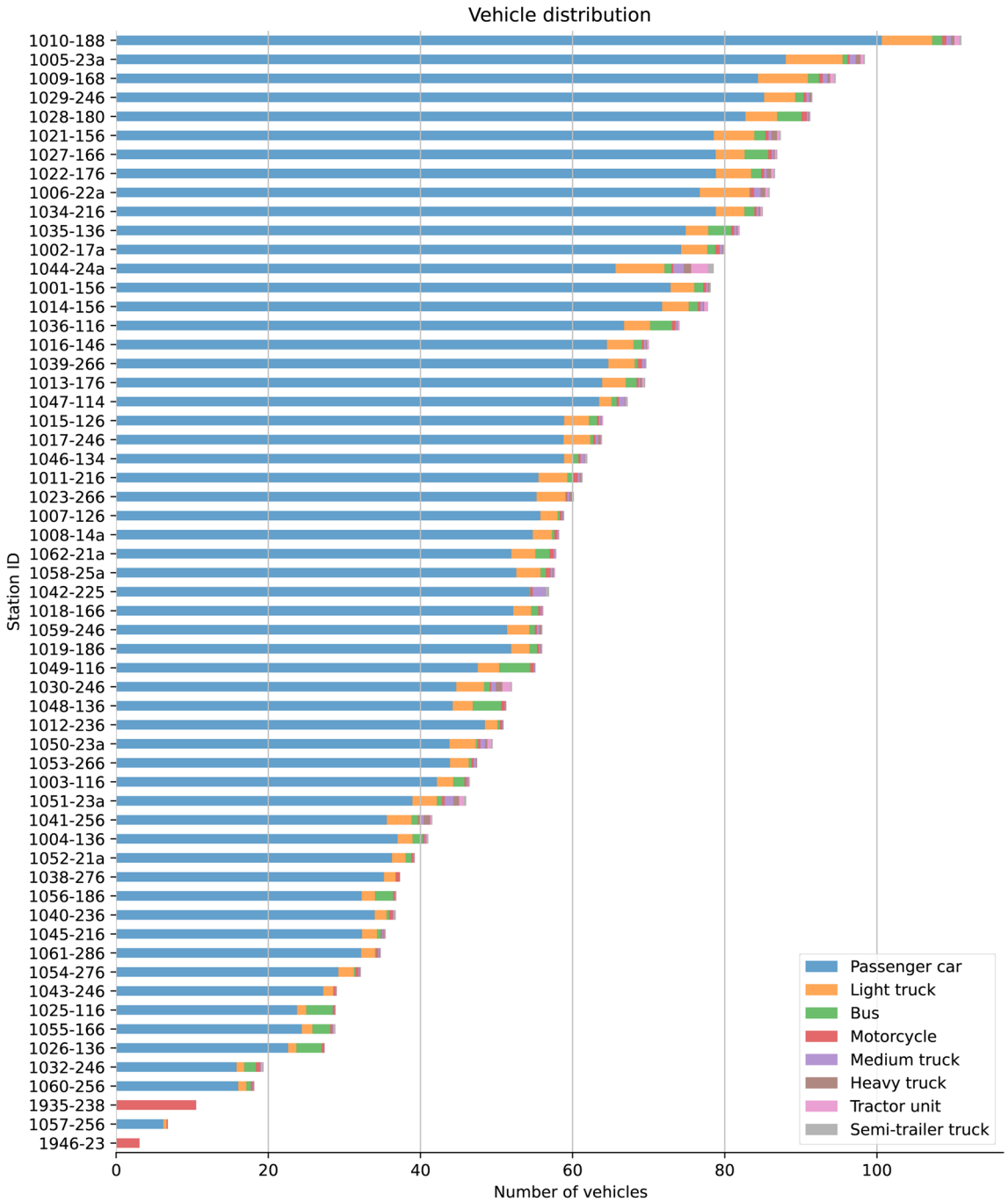

Fig. 3 Average number of vehicles (for 15-minute intervals) grouped by type and measuring station. Aggregation is performed over all the years of data acquisition

transmitted is symbolized by the gap between $C1$, $C2$, $C3$, and the model building phase. At a certain point in time, a model is constructed that attempts to model traffic behavior in the future based on the data collected up to that point. At the present time, predictions can be made about traffic patterns, for the next day. Note that besides missing data due to delayed acquisition, data may also be missing due to a malfunctioned measuring station. These missing data are indicated by the gaps in the red and green lines for the first two stations. Such a modeling framework also has some limitations. Due to delayed (e.g., manual) data transfer, the most recent data may be missing at the time the

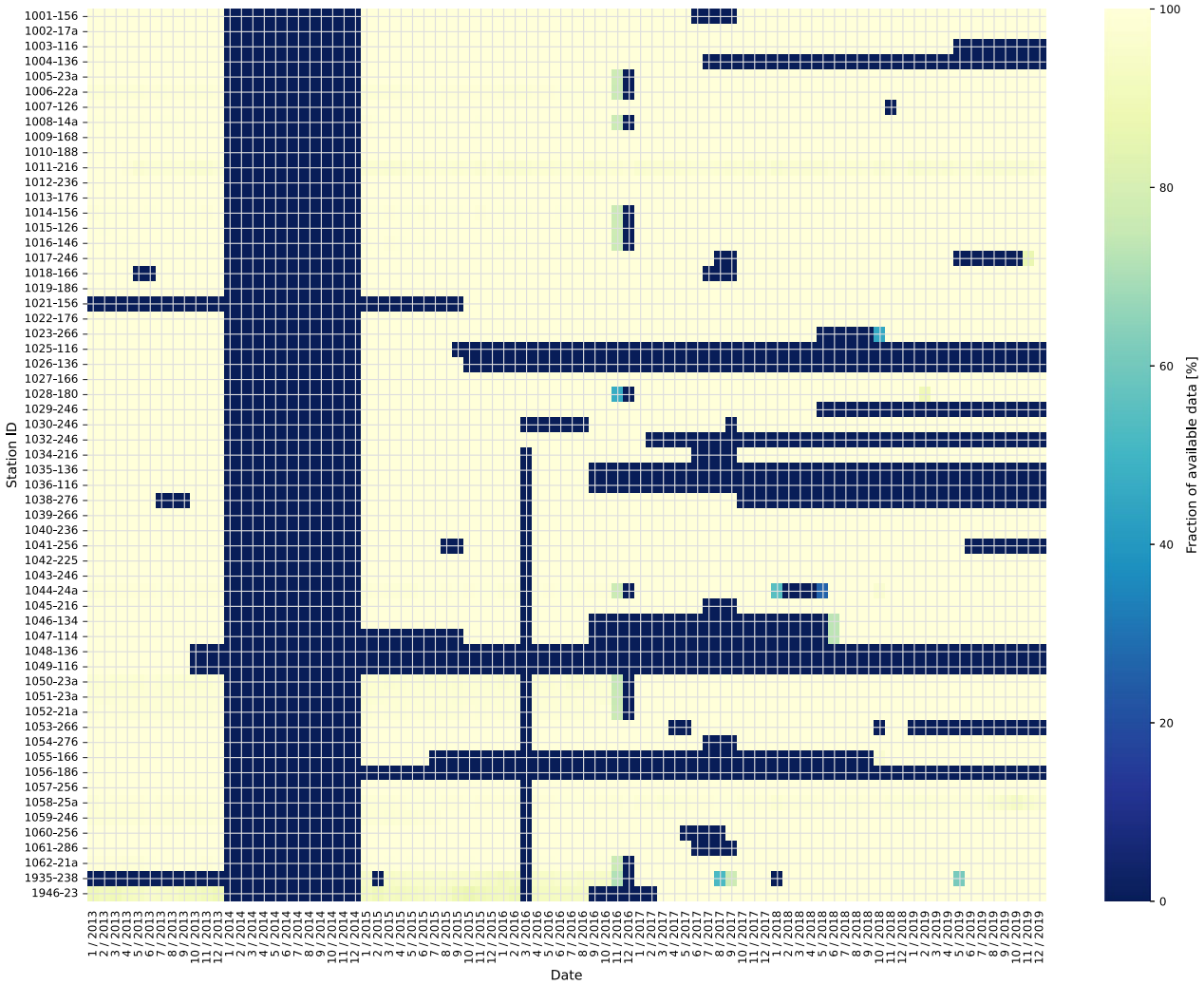

Fig. 4 The percentage of available data over the years for each measuring station. The color indicates the percentage of the data that is available and what percentage is missing

model is created. For this reason, it is not possible to use methods commonly used for forecasting, since the recent past is rarely available. In a typical forecasting framework, forecasts are made in real time based on the up-to-date data, $C1$, $C2$, $C3$, and model building is also done at the present time.

4.2 Error metric

Choosing a metric to optimize the prediction of the number of vehicles can be challenging, as it must be aligned with our desired predictions and goals [43]. The mean absolute error (MAE) [44], was selected and is used to report all results. The metric is defined as:

$$MAE = \frac{1}{C}\sum_{c}\left|y_c^{forecast} - y_c^{true}\right| \quad (1)$$

where $y_c^{forecast}$ and y_c^{true} are the predicted and measured number of vehicles passing measuring station c in a given time interval. Note that the sum is not applied over missing values. C represents the total number of measuring stations. The MAE described in Eq. 1 has numerous advantages over similar metrics such as mean squared error (MSE) [45, 46] and mean absolute percentage error (MAPE) [47]. This MAE is not as sensitive to the outliers [48], which are common due to inconsistencies in the data and can significantly affect the performance of some models. The MSE can often heavily penalize the models when the prediction errors are large. This behavior may be desirable in some cases, but results in degraded performance due to problems with the data (e.g., corrupted data, changed traffic patterns due to rare events). Another commonly used metric that minimizes percentage error

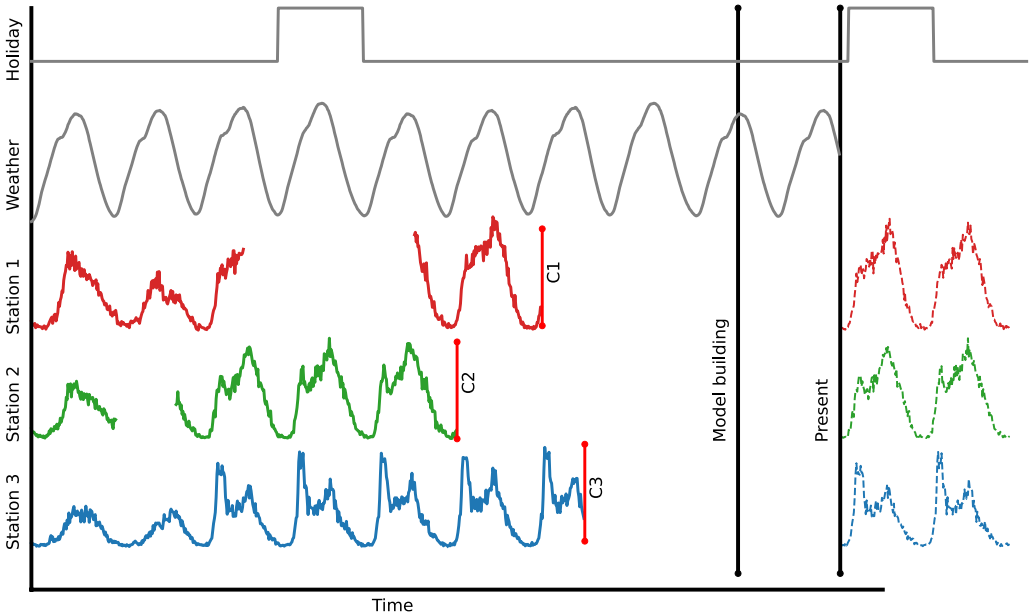

Fig. 5 Modeling framework used to make future forecasts with data collected from three measuring stations together with known past covariates such as weather data and future covariates indicating public holidays

is MAPE. However, it can significantly penalize an incorrect prediction when the measured number of vehicles is relatively small. An example of this situation is night traffic, when there are few vehicles on the road and the actual number of vehicles could be zero or close to zero. In such cases, MAPE assigns a large error to the predictions, even if they differ by only a few vehicles. Also, note the difference between the error metrics used to indicate the quality of the predictions and the loss functions used by the models to minimize the errors. In our study, we try to ensure that the machine learning models use a loss function that is identical to the error metric to ensure a fair comparison between models.

4.3 Feature engineering

In constructing time-dependent models, it is a common practice to create domain-specific features from timestamp data to improve predictive performance [49, 50]. Since the data set consists only of the timestamp, the weather on the day of measurement, and the presence of a public holiday, it is important to engineer features that can be correctly interpreted by machine learning algorithms. Features constructed from timestamp data must capture information that can be effectively used by machine learning models to generalize learned knowledge to previously unseen instances. A common approach for time-series-based feature engineering is the encoding of cyclic features [51], in which time-based features are encoded with cosine and sine functions such that they periodically repeat with some frequency. In such coding, periodic/cyclic events are preserved and events that are close in time have similar embeddings. In addition, one-hot encoded time features are also used to improve the performance of the algorithm. This helps in modeling independent influences between events that are close in time but not similar [52]. This encoding captures additional information that is not available by using just cyclic features encoding. Table 2 shows features constructed from the date and time information of the 15-minute measuring interval along with weather-dependent features that we use to make predictions based on weather over the past five days. All generated features are scaled with a zero mean and a standard deviation of one, which helps the convergence of some algorithms (e.g., neural networks) but does not affect others (e.g., tree-based algorithms).

4.4 Models

To successfully model traffic, based on the features constructed from timestamps and weather data, is a challenging task. This section describes the models that have been used for traffic modeling and lists some advantages and disadvantages of each model. It is important to note that for real-time traffic forecasting, one can use more advanced machine learning methods, such as those described in Sect. 2, which take into account temporal dependencies (commonly known as lag features). Unfortunately, such approaches are not relevant in our case

Table 2 Feature type, their dimension and description

Feature type	Count	Description
One_hot_months	12	Binary vector of the month
One_hot_day_of_week	7	Binary vector of the day of the week
One_hot_minute	96	Binary vector of the 15-minute interval in a day
Sin_cos_day	2	Cyclic time coordinates with a period of one day
Sin_cos_week	2	Cyclic time coordinates with a period of one week
Sin_cos_year	2	Cyclic time coordinates with a period of one year
Linear_year_timespan	1	Time form 2013 to 2020 normalized to [0, 1]
Is_dst_feature	1	Indicator of a daylight saving time
Holiday	**1**	**Indicator of a public holiday**
Temperature	1*	Average daily temperature
Wind_speed	1*	Average daily wind speed
Cloud_coverage	1*	Average daily cloud coverage
Humidity	1*	Average daily humidity
Air_pressure	1*	Average daily air pressure
Rainfall	1*	Average daily rainfall
Snowfall	1*	Average daily snowfall
Sunshine	1*	Average daily sunshine duration
Dew	1*	Presence of dew
Glaze_ice	1*	Presence of glaze ice as a result of a freezing rain
Icy_ground	1*	Presence of ice on the ground due to refreezing

Features are divided according to their source. Features on the top are obtained from the timestamp of the measurement, the holiday feature comes from an eternal source and indicates if a given day is a public holiday. Lastly, weather features describe the statistics from the previous few days. Note, * indicates that we can include features from a more distant past. If more than one day is included, the number of weather features is subsequently proportional to the number of days that are included

due to the current data acquisition process described in Sect. 4.1. When making a prediction, one does not have the latest data on traffic conditions, so one cannot rely on temporal data to make the prediction.

The used machine learning models can be roughly described as single-target learning (STL) models or multi-target learning (MTL) models. The STL approach assumes that the targets are independent, meaning that traffic flow from one station does not affect traffic flow at other stations. If measuring stations are considered independent, multiple models must be constructed, with each model predicting traffic for only one specific measuring station. Unlike STL, MTL models are trained to predict multiple targets simultaneously, allowing them to exploit potential dependencies between targets that aid in model generalization. In this approach, only one model is constructed that can model traffic at all measuring stations simultaneously. In our paper, we use models in the MTL configuration, where construction and training can be much more efficient because it is not necessary to create multiple separate models for each of the traffic measuring stations.

When choosing models, it is necessary to pay attention to all constraints. For this reason, we focus on the models that meet the following criteria: a) training a model is efficient and can be done in a reasonable time (e.g., up to a few hours), b) models must be easily updated when new data are available. Due to these limitations, we exclude some of the popular machine learning algorithms such as XGBoost and Random Forest, models that are commonly used for tabular data. Both techniques are somewhat limited when it comes to i) selecting objective to optimize (degraded performance when using MAE), ii) predicting multiple values simultaneously (some XGBoost implementations do not support multi-target learning (at least not at the time of experimentation and writing this paper), and thus are not able to predict traffic at multiple stations), iii) not able to handle missing outputs in a multi-target scenario, iv) require rebuilding the model without support for partial updating as new data become available.

4.4.1 Baseline mean

A simple model for traffic flow on a road segment predicts the average traffic volume for a given measuring station regardless of time and weather conditions. It cannot capture periodic changes in traffic volume, but serves as a baseline for comparing more advanced models.

4.4.2 Linear regression

Linear regression (LR) models linear relationships between dependent and independent variables using learned coefficients. LR provides good interpretability and easy/fast optimization. For traffic flow prediction, we create one LR model per station and train with MAE loss instead of MSE loss to better meet the problem objectives. Stochastic gradient descent is used to iteratively update the LR weights during training so that it is not deterministic. One could also consider LR as a baseline model that can be achieved if only linear relations are modeled.

4.4.3 Decision trees

Decision tree learning [53] is a simple machine learning approach for creating explainable predictive models. It automatically constructs a decision tree from data, where each branch represents a test for an attribute and the target value for the prediction. There are several approaches to building decision trees [54], such as ID3, C4.5, CART, CHAID, and MARS, which differ in how they handle numerical and discrete data and create branches. Most of these algorithms are deterministic, but randomness can be introduced if the features have the same importance. While the construction of decision trees is usually quick, determining how to create branches can be time-consuming, especially when using loss functions such as MAE that require sorting of node values.

4.4.4 Tree-specific ensembles

A popular approach for increasing the prediction accuracy of a single decision tree is to train multiple trees and combine their predictions [55]. Such a procedure is often referred to as a random forest. The individual trees in a Random Forest are trained on randomly drawn training data with replacement, known as bootstrap aggregation [56]. Thus, a Random Forest is an ensemble of a large number of decision trees. The algorithms used to build trees are generally deterministic and therefore often create exactly the same tree. For this reason, the individual trees that are part of the ensemble are usually trained on a random subset of the data or with a random subset of the features to ensure that the ensemble consists of different trees. A similar technique of tree ensembling is Extremely Randomized Trees [57], where the attribute splits are randomly selected. The main advantage of such an approach is computational efficiency due to the simplified criteria for splitting the nodes.

4.4.5 Neural networks

Artificial neural networks (ANN) are mathematical models typically used to discover complex and nonlinear relationships between dependent and independent variables. A neural network consists of neurons organized into individual layers of a certain size. The layers can then be stacked to form a more complex network capable of approximating complex nonlinear functions. Once a neural network is created, it must be trained to learn the mapping between input and output data. The most common technique for this is backpropagation [58] where gradient descent is used and weights of individual layers are iteratively adjusted to minimize the loss function. Although neural networks exist in many different configurations, often tailored to specific domains [59, 60], we focus here only on fully connected neural networks.

4.4.6 Model-agnostic ensembles

Model-agnostic ensembles [61–63] are sets of various machine-learning models that use different techniques to combine their knowledge to get more reliable and more accurate predictions. Depending on the type of problem, such as regression or classification, there are different ways in which predictions can be combined. For regression problems, a popular technique is to combine weighted predictions to obtain a new, more reliable estimate. The weights between the different models can be based on the performance of the algorithms that are part of the ensemble, or all predictions can have the same weight. In the latter case, the final prediction is calculated as the average of the individual model predictions. Due to its simplicity, we use the second approach by simply averaging the predictions. In our study, model-agnostic methods are used only in combination with neural networks.

5 Results and discussion

In this section, we first present the models with their configurations and hyperparameters, and show how different hyperparameters affect the prediction accuracy. We then select the model that performs best in the validation set and examine its performance on the test set. We analyze the performance of the model for each measuring station and assess the impact of different events. We also examine how the performance changes over time. It should be noted that some measuring stations and time periods are selectively chosen to highlight interesting patterns in the data, such as the effects of rush hour and public holidays. However, not all measuring stations exhibit these distinct patterns.

5.1 Experimental setup and implementation details

To evaluate the models and their performance on previously unseen examples, the data are split into a training set and a test set that are used to train the model and measure performance, respectively. When splitting the data into two parts, it is important to maintain the temporal order of the data and ensure that the test set

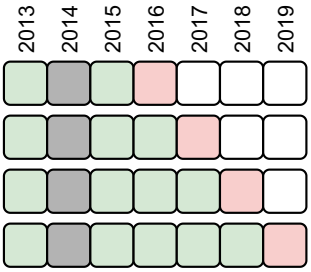

Fig. 6 Temporal split of the data into four train (green) and test (red) data sets obtained with 4-fold cross-validation. Data from the year 2014 is not available and is thus not used

contains the most recent observations and does not overlap with the training data [64]. For this reason, time-series cross-validation with four splits is used. Figure 6 shows more details on how the data was split for evaluation. Note that all presented models are stochastic in their nature, and for each train/test split and hyperparameter combination, the models are repeatedly trained 20 times to obtain a better and more reliable estimate of their performance on different data sets and with different parameters.

When selecting and evaluating models, one must simultaneously select the best model and the best set of hyperparameters for the selected model. In order to evaluate the models properly, test data must never be used during the tuning of the hyperparameters to avoid overly optimistic performance of the model. For model construction and selection, we used nested cross-validation [65] where each training set was split into two sets, one of which trains algorithms with different hyperparameters and the other serves as a validation set from which the best hyperparameters are determined.

In general, finding the optimal neural network architecture can be a complex task that requires a lot of computation time and different tuning techniques [66–68] to find optimal hyperparameters. Table 3 describes the hyperparameters that were explored during model building. A simple grid search was performed over all combinations of parameter values, which means that 20 models are constructed for each train/test split and hyperparameter combination. In the following section, we show how different hyperparameters of certain models affect the performance and report the performance of all hyperparameter combinations. In real applications, e.g., daily use, one would select only the model and hyperparameters that achieve the best validation set accuracy.

All algorithms were run independently 20 times for each pair of train/test split and hyperparameter combination on an Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz, 1 TB of RAM, and the Ubuntu operating system. The algorithms are implemented in Python 3.7. Training was performed using the frameworks *PyTorch* [69] and *PyTorch Lightning* [70]. Analysis was performed using *Snakemake* [71] and *scikit-learn* [72].

Table 3 Hyperparameter ranges where grid search was performed

Model type	Hyperparameter	Values
Neural network	Depth	{1,2,3,4,5}
	Layer size	{128, 256, 512}
	Dropout	{0, 0.2}
	Early stopping patience	{5}
	Optimizer	{Adam}
Decision tree	Maximum depth	{2, 4, 6, 8, 10, None}
	Minimum samples in leaf	{1, 2, 3, 4, 5}
Extremely randomized trees	Maximum depth	{2, 4, 6, 8, 10, None}
	Minimum samples in leaf	{1, 2, 3, 4, 5}
Linear regression	Optimizer	{SGD}

5.2 Model selection

When building and evaluating models, one has to find a trade-off between multiple constraints. Some of these are the time required to build/train a model, optimization of hyperparameters, complexity of the model, and performance at inference, as well as other limitations such as the difficulty of retraining models when new data become available. The models described in Sect. 4.4 are trained and evaluated on four train/test splits of the data, as described in Sect. 5.1. With this in mind, we first focused on different model types such as decision trees, random forests, and neural networks, and performed hyperparameter tuning for each model. Figure 7 compares the different model types in terms of MAE between the predicted and measured number of vehicles. Shown are the models with the optimal hyperparameter values determined using the validation set. The gray bars represent the standard deviation of performance over 20 repeated runs. The analysis shows that all models outperform the simple baseline model, which always predicts the mean, obtaining an MAE error of about 40.

Focusing first on the tree-based models, we see that they achieve good predictive performance and that their error is about half that of a naive baseline. For both methods, the optimal hyperparameters obtained on validation data are obtained when the tree depth is not limited and each leaf contains at least two samples. An interesting observation is also that their performance varies greatly across the years considered, with one year having a significantly higher error than the others. Since both tree

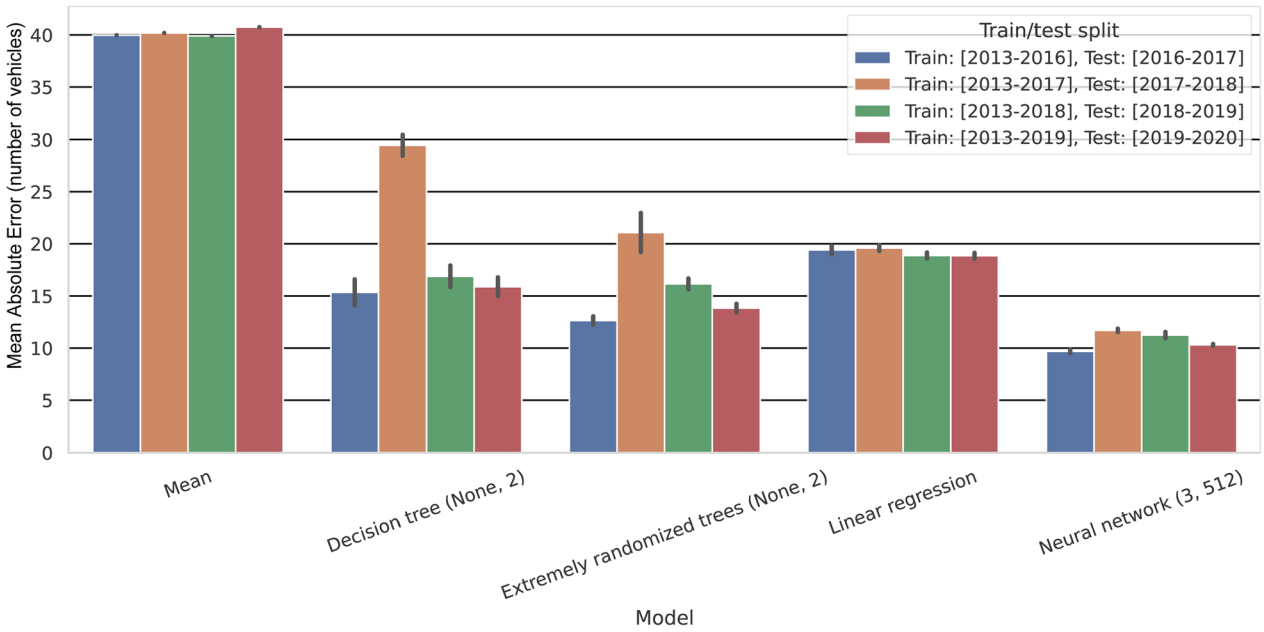

Fig. 7 MAE between measured and predicted number of vehicles for different machine learning models when evaluated on previously unseen year. Each model was trained and evaluated 20 times on 4 train/test splits. Grey bars show standard deviation in performance

models are stochastic models, they also exhibit a large variation in performance between runs of the same data set. For linear regression, the accuracy is also about twice that of the baseline. Compared to the tree-based model, the performance of the linear regression is much more stable across the years considered, with no large drops in performance for certain years. Linear regression is also much more consistent when trained repeatedly on the same data, with minimal differences between runs. Finally, the best-performing models are neural networks, which can significantly outperform other models, with the prediction being relatively stable across the years considered and between different runs. In this case, the optimal number of hidden layers is three, with each layer having 512 neurons. Even with non-optimal hyperparameters, neural networks almost always outperform all other models. The reason why neural networks perform better than regular tree-based models has not been explored in detail, but could be due to the relatively large number of training instances and the ability of neural networks to model complex decision boundaries. Since neural networks perform significantly better than other models even when hyperparameters are not optimal, emphasis has been placed on the use of neural network-based models for traffic modeling. One of the hyperparameters of neural networks that is not shown is the Dropout rate [73]. In the models, we investigated that the dropout rate did not improve the performance in any of the cases and therefore it is not shown.

Since in our case neural networks are better suited for traffic flow prediction than other methods, we further explore how their hyperparameters affect training. Figure 8 shows various hyperparameters and how they affect traffic flow prediction performance. We find that the choice of parameters has a large impact on performance. Networks with only one hidden layer generally cannot capture all the information and therefore perform poorly with MAE at around 15. Increasing the number of hidden layers and thus parameters can help neural networks make better predictions. The best-performing neural network is the one with three hidden layers with 512 neurons each. This is true for both the validation set (this was therefore selected as the best performing in the previous section) and the test set.

5.3 Neural network ensembles

We now focus specifically on neural network ensembles. When building machine learning models used for forecasting, it is often the case that combining predictions from multiple machine learning models can improve performance. Figure 9 compares regular neural network models with ensembles of neural networks combined into an ensemble of size 10. The neural networks within the ensemble are initialized and trained independently, and thus learn different knowledge. We can observe that forming an ensemble is always beneficial and can further improve the prediction performance. As before, the ensemble that achieves the best performance is one

Fig. 8 MAE between measured and predicted number of vehicles for neural network-based models when evaluated on the previously unseen years with different hyperparameters for a number of hidden layers and their size. Each model was trained and evaluated 20 times on 4 train/test splits. Gray bars show standard deviation in performance

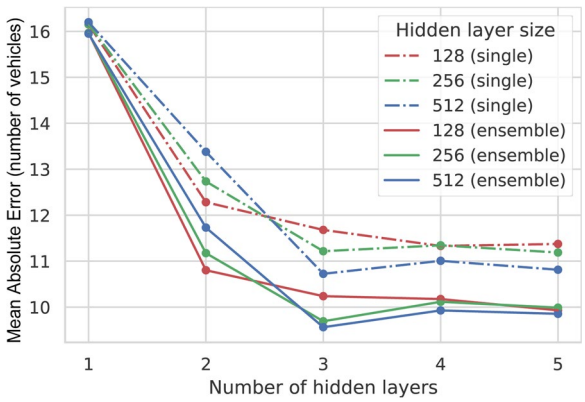

Fig. 9 Effects of neural network architecture on the MAE with changing number of hidden layers (1–5) and width of those layers (128, 256, 512) for regular neural networks and ensembles

that consists of neural networks with three hidden layers of width 512. Further increasing the complexity of the models within the ensemble does not lead to better performance.

5.4 Training time

Minimizing the time required to build the model is critical because traffic data may need to be updated frequently and it may be time consuming to train or build a new model from scratch each time. See Table 4 for the training and prediction times of each model. Since the models are trained on different sized train/test splits, as described in Sect. 5.1, comparing the models across all folds is not useful. For this reason, we focus only on one

Table 4 Training time for different types of models

Model type	Train time [s]	Inference time [s]
Mean	0.19	0.01
Regressor model	125.68	0.16
Decision tree (None, 2)	23.79	0.06
Extremely randomized trees (None, 2)	145.51	0.57
Neural network (128, 1)	229.69	0.30
Neural network (256, 1)	236.93	0.31
Neural network (512, 1)	266.45	0.32
Neural network (128, 2)	655.69	0.52
Neural network (256, 2)	655.69	0.57
Neural network (512, 2)	642.70	0.59
Neural network (128, 3)	819.35	0.74
Neural network (256, 3)	820.70	0.87
Neural network (512, 3)	899.02	1.86
Neural network (128, 4)	930.53	1.47
Neural network (256, 4)	1080.80	2.08
Neural network (512, 4)	1293.85	2.64
Neural network (128, 5)	1263.38	1.54
Neural network (256, 5)	1486.36	2.45
Neural network (512, 5)	1510.61	3.40

Obtained values are averaged over 20 runs for train test split 2018

train/test split that occurred in 2018. All the reported values are averages across all 20 runs. We can observe that the different model types have significantly different training times. Training a baseline model that always predicts the mean traffic of a given station is incredibly fast, but does not provide great accuracy. On the other hand, tree-based models provide better accuracy at a higher computational cost. Also, comparing one decision tree to extreme trees, one can observe that training an ensemble of trees is more computationally expensive. The most computationally expensive models are neural networks and ensembles built on top of them. The time required to build them depends heavily on the size of the network and its architecture. As might be expected, deeper and wider networks tend to be slower to train and slower to make predictions. The most computationally expensive network to train consists of five hidden layers, each with 512 neurons. Note that the time required to train ensembles is usually proportional to their size, so we do not include their training times in the Table 4. Ensembles consisting of 10 neural networks require on average 10 times more time to train if the training is performed sequentially. Note that neural networks that use Dropout were excluded from the list because their training time is similar to that of networks without Dropout.

5.5 Exploring predictions

Based on the accuracies reported in the previous sections, an ensemble of 10 neural networks with three hidden layers and a width of 512 was selected as the best model for the validation and test set. Figure 10 shows the distribution of prediction errors (subtracting the true value from the predicted one) for individual measuring stations with the selected models. Two important observations can be derived from the figure. It is more difficult to predict traffic volumes for certain measuring stations than for others. This is mainly because they are located on busy roads, where daily traffic variations are greater than on less busy roads. The figure illustrates the presence of outliers in the prediction errors. Most forecasts are accurate, but errors of magnitude 100 or greater are common. These errors usually result from extreme events that existing models struggle to capture accurately.

Figure 11 shows a more detailed comparison between the actual data and the predictions of the selected model. Each point represents the traffic flow at either 6 AM or 8 AM. The first pattern that can be observed are clear differences between weekdays and weekends. Weekend traffic is less frequent and with less variability than weekday traffic, so weekend forecast errors tend to be lower. The other distinct change in patterns that can be observed is the change between standard time and daylight saving time, as shown in Fig. 11. This change will significantly alter vehicle frequency when it occurs. The selected model can account for this and predict the change in traffic distribution.

Figure 12 shows a small example of predicted versus actual traffic for measuring station 1021-156 for two weeks in February and March. Although this is a hand-picked sample, some interesting observations can be made. First, the model can accurately learn to model daily patterns, such as morning rush hours and low traffic flow during nights and weekends. This suggests that it is possible to learn predictable periodic patterns that appear in the train data and use them to predict future traffic volumes. A more problematic pattern that cannot be easily captured by the model is the shift in traffic volumes. We can observe that in the example shown, the model consistently underestimates traffic volumes in the test group. This is consistent with the idea that traffic volumes at a given measuring station have increased over time. Therefore, the most recent test data, containing only the most recent measurements, may have a different distribution than the training data. The consistent overestimation or underestimation of traffic flow by the model implies that traffic volumes can change suddenly, but the models consistently produce biased predictions due to a lack of real-time information. Recent traffic count values would likely allow the models to account for this shift and make more accurate forecasts. Unfortunately, the current data transfer scheme does not provide the model with these features.

To get a more detailed look at the errors, Fig. 13 shows the predicted values and the actual measured values for station 1021-156 over the four-day period. We can see that in this case the actual observations are relatively noisy between the 15-minute intervals and that the predicted values can capture the overall traffic patterns without overfitting to the noise. While we do not include visualizations for all stations and all time periods, the pattern of the models being able to accurately estimate the distribution without predicting the noise can be observed for other stations and time periods as well.

Data drift on test data is further analyzed in Fig. 14, where we show how the prediction error changes over the one-year period without updates to the model (no retraining). We present MAE between the measured and predicted number of vehicles, aggregated across all measuring stations for which data are available for each day, one year into the future. Note that measuring stations with missing data were not included, which means that MAE in this figure cannot be directly compared to MAE in other figures. While the single aggregated daily error is noisy, the linear line reveals a trend that is present over time. This shows an important pattern that traffic patterns change over time and that if the model is not

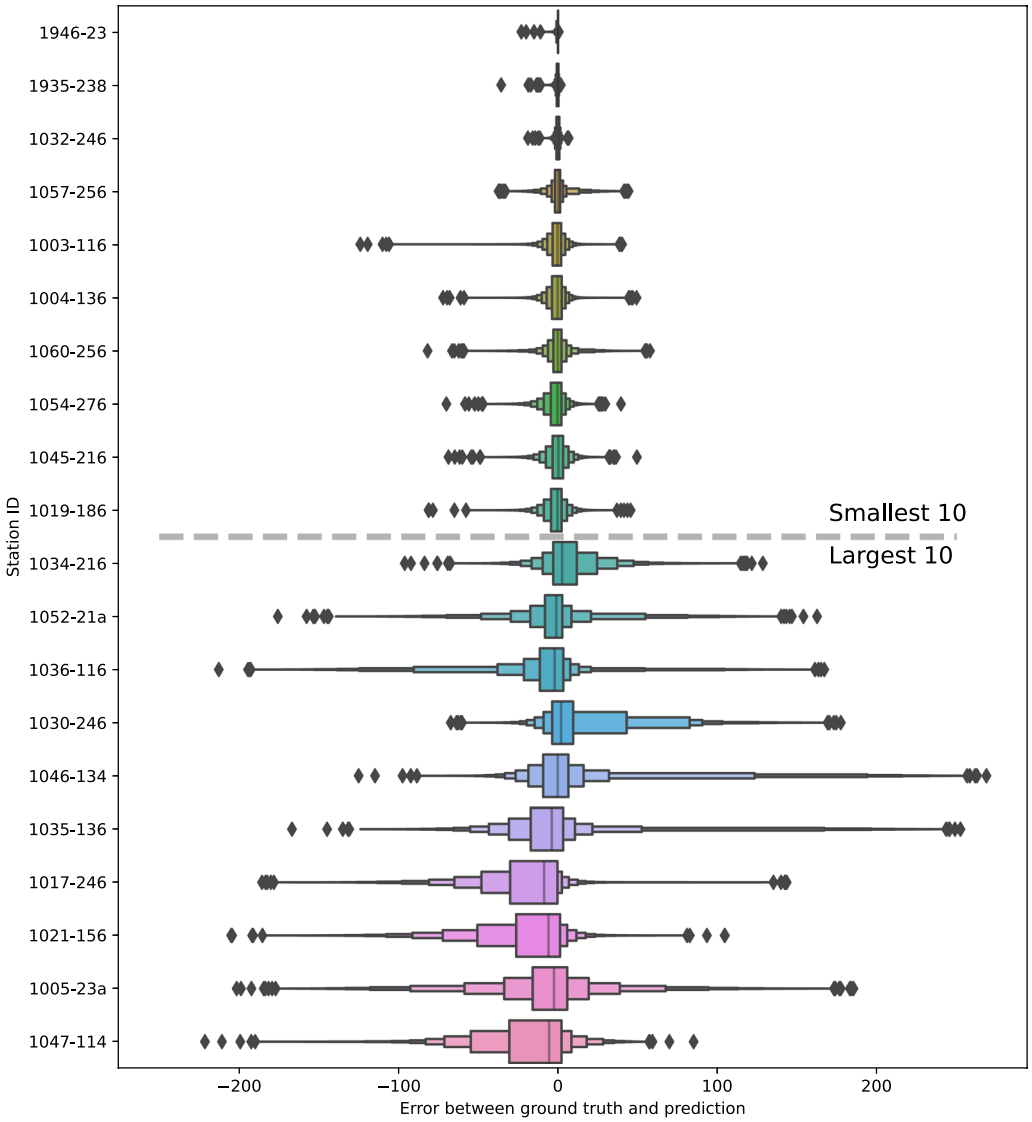

Fig. 10 Prediction errors made with a neural network with 3 hidden layers and 512 neurons per layer for 10 individual measuring stations with largest and smallest errors

updated regularly, a performance drop can be expected. In addition, a distinct jagged line can be observed over the course of the year. This pattern is a result of higher traffic volumes and weekday variability. Therefore, the predictions made for weekdays carry a larger MAE.

Finally, we provide important insights into the meaning of various feature types used for predictive purposes. As outlined in Sect. 4.3, three different feature categories are introduced. These categories are derived from dates and times, the presence of public holidays, and past weather descriptions. In Sect. 5.3, the best ensembles have a MAE between 10 and 12 when all three feature categories are used. However, not all feature groups have the same importance. The most important are the features formed by date and time, while the features indicating holidays and weather features contribute much less to the prediction. If you take the best ensemble and retrain it without the weather features, the MAE increases by only about 3%. Similarly, if the ensemble is trained with only date/time and weather features without holidays data. Then the MAE increases by about 3.5%. But even if the overall MAE without holidays or weather features only decreases by a few percent, these features are still extremely important. When holidays or severe weather events occur, these features greatly improve the predictions. In other words: Weather and vacation features can significantly

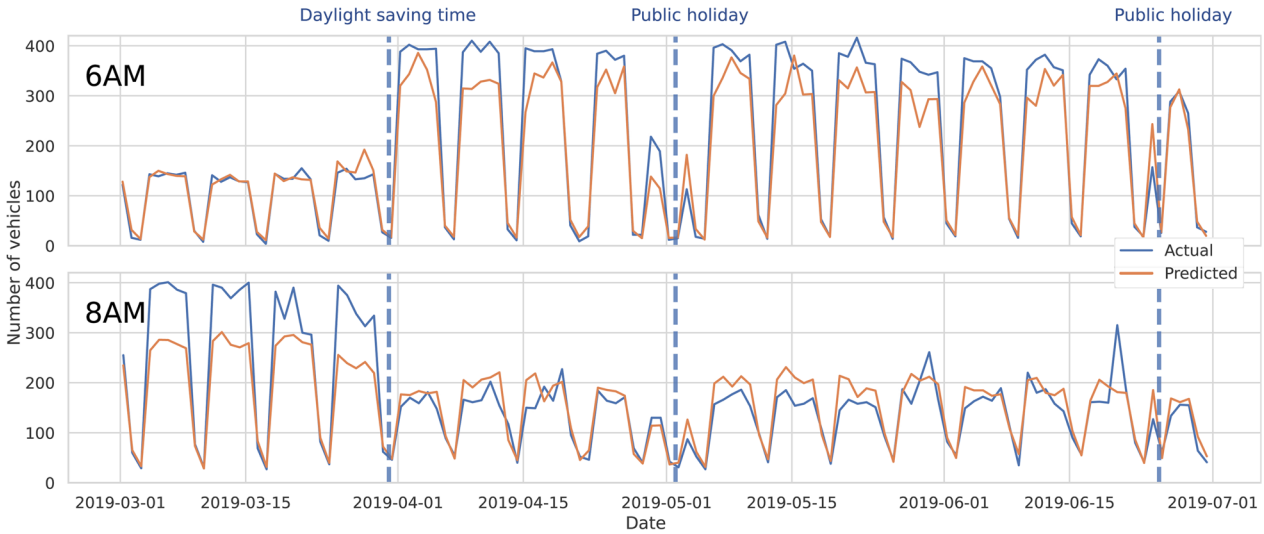

Fig. 11 A small subset of test data showing actual traffic vs prediction using model Neural network(3, 512) for station 1006-22a and period of 4 months from the beginning of March to end of July at 6.00 AM (top) and 8.00 AM (bottom). The orange line shows the predicted number of vehicles for 15-minute intervals while the blue line shows the actual amount of vehicles

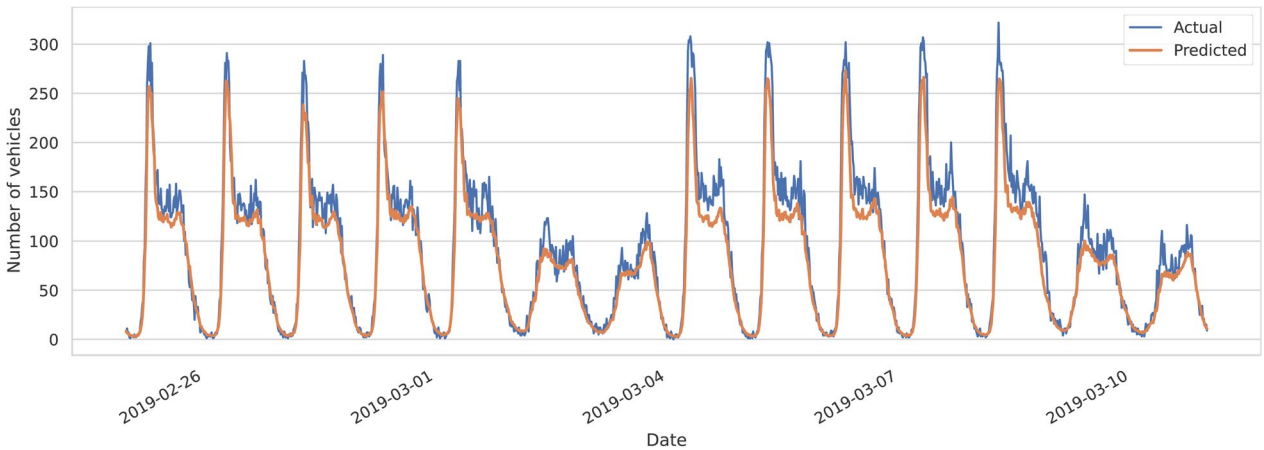

Fig. 12 A small subset of test data showing actual traffic (blue) vs prediction (orange) using model Neural network (3, 512) for station 1021-156 and period of two weeks. Each prediction is made for the number of vehicles that will drive over the measuring station in 15-minute intervals

increase model accuracy for rare events, but since such events are rare, this is not reflected in the overall MAE summary.

6 Limitations

This section describes some of the limitations and trade-offs of the study. The MOL-TR data set has some missing data and data that may have been recorded incorrectly. To mitigate the missing data problem, we manually merged some stations that represent the same station but were moved or assigned to a different ID during the data acquisition period. The process of combining the data from stations with different IDs is human-made and therefore prone to error. Also, when modeling traffic volumes, we decided to forecast only the total number of vehicles and not their exact type. Thus, we aggregated vehicle types and did not distinguish between individual vehicle types, which could contain additional information not captured by the models.

When splitting the data for training and validating the models, we perform only four splits. The reason for this is the computational complexity of repeating the training 20 times and optimizing the hyperparameters for each fold. We chose to perform at least 20 runs to find out how independent model-building procedures affect model performance. The hyperparameter tuning in this study

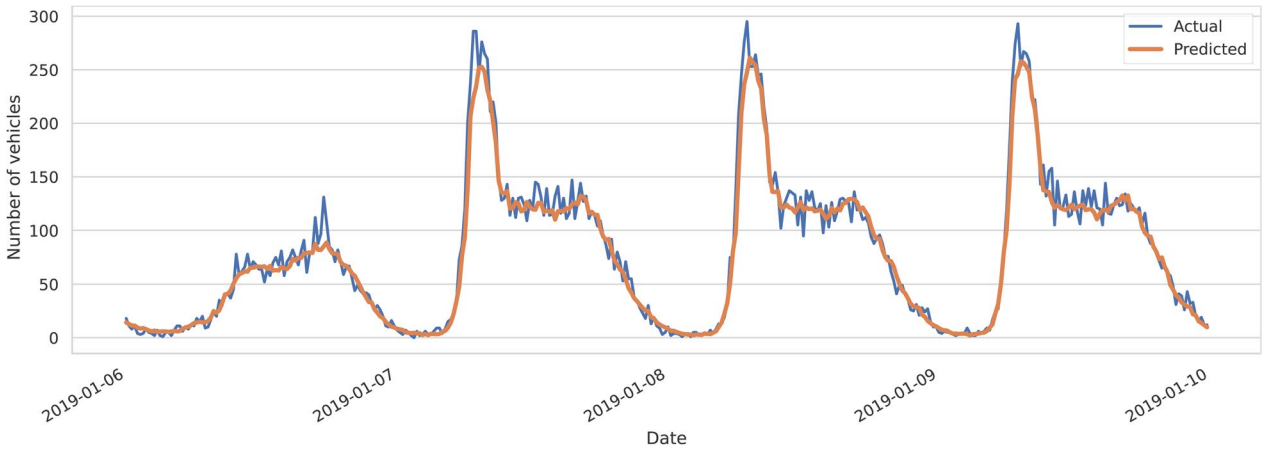

Fig. 13 A small subset of test data showing actual traffic (blue) vs prediction (orange) using model Neural network (3, 512) for station 1021-156 over a period of four days for individual 15-minute periods

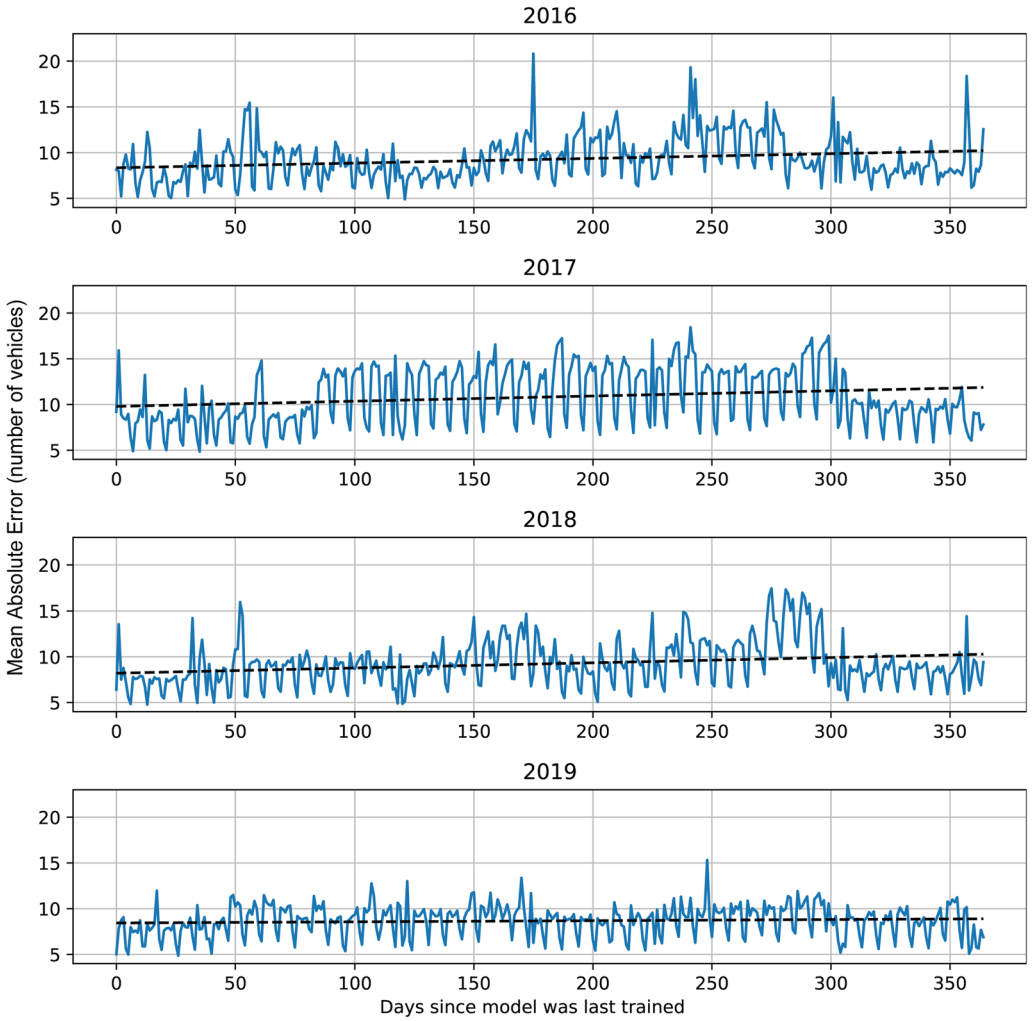

Fig. 14 The MAE is calculated daily for 365 days into the future, comparing the measured and predicted number of vehicles across all stations. The line reveals an interpolated linear trend, demonstrating a gradual decline in predictive accuracy as time progresses

was done in two steps. First, we examined the hyperparameters for a number of different learning algorithms. Once it was clear that neural networks outperform other models, we performed additional hyperparameter tuning for neural networks only. This was necessary to avoid extensive parameter tuning for all models and the associated high computational costs. Similarly, when selecting models to evaluate, we primarily looked for models that inherently support the minimization of the MAE metric and can do that efficiently. When used with MAE, some tree-based methods search for median values when splitting on a feature, which slows down the whole process. Finally, some of the machine learning methods may have problems with feature representation. This is especially true for many one-hot coded features, which can have a negative impact on tree construction [74].

7 Conclusion

In this paper, we first introduce a new MOL-TR data set that tracks traffic flow for several vehicle types on different road sections between the years 2013 and 2020 in Ljubljana, Slovenia. Using inductive loop measuring stations located under the roads, vehicles are counted and classified at 15-minute intervals. To the best of our knowledge, this is one of the larger traffic data sets that spans multiple years or different vehicles in multiple locations.

With this data set, we build models that are able to incorporate traffic data together with additional weather data and data on public holidays without using temporal dependencies in making predictions. This requirement is crucial because the data is not always collected in real-time and the most recent past is not available to provide accurate predictions of vehicle flows. We show that, despite existing limitations, useful models can be created with sufficient feature engineering. These models are particularly useful for traffic forecasting because they can capture and account for various factors and trends that may affect traffic patterns, providing valuable insights for short-term planning and decision-making in traffic management. We compare the models and the impact of hyperparameters on model performance and conclude that neural networks offer the best trade-off between accuracy, training/inference time, and the simplicity of incremental updating compared to other linear and tree-based models. In addition, we analyze the advantages of combining multiple models, which can lead to higher accuracy, but also have the disadvantage of increased computational complexity.

Finally, we analyze the best-performing model to gain further insight into when it works well and where it does not provide accurate predictions of traffic flow. We show that the predictions obtained are usually relatively accurate for the majority of measuring stations. However, there are some scenarios that are more difficult to model. Such examples are holidays when traffic volumes can change significantly, and the switch to daylight saving time. We also show that the models can consistently underestimate or overestimate traffic flow and that the performance of the model can decrease over time due to data drift. Therefore, it is important that the models are updated regularly as new data become available.

The direction of future work concerns the deeper analysis of features, their construction, and the quantification of their importance. The focus should be on, first, improving current features to achieve better performance, and second, quantifying their importance to make the models more explainable. A large part of the paper deals with parameter tuning. Given the growing popularity of AutoML techniques, finding a framework that meets all requirements (i.e., speed of training and inference, incremental updates) and produces good results is desirable. In modeling time-dependent problems, a common technique is to assign higher weights to more recent samples. Further analysis should be performed if assigning weights can increase performance and reduce problems such as drift. Finally, efforts should be made to detect and mitigate drift caused by ever-changing traffic flows.

Abbreviations

IoT	Internet of Things
MA	Moving Average model
AR	Auto-Regressive model
ARIMA	Auto-Regressive Integrated Moving Average
CNN	Convolutional neural networks
N-BEATS	Neural basis expansion analysis for interpretable time-series forecasting
DeepAR	Probabilistic Forecasting with Autoregressive Recurrent Networks
MOL-TR	Municipality of Ljubljana traffic data set
MAE	Mean absolute error
MSE	Mean squared error
MAPE	Mean absolute percentage error
STL	Single-target learning
MTL	Multi-target learning
LR	Linear regression
ANN	Artificial neural networks

Acknowledgements

The authors would like to express gratitude to the Municipality of Ljubljana for their contribution and support within the initial research and for providing important data about the studied case.

Author contributions

GP has been responsible for data acquisition and management, designing methodology, model development and writing. RH has contributed primarily with designing methodology, model development and writing. GP has been responsible for data acquisition and management, designing methodology and writing. All authors have participated in discussions and have approved the manuscript for publication.

Funding

The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0098 and young researcher grants).

The work is also part of a project that is funded by the European Union from Horizon Europe under grant agreement No 101077049 (CONDUCTOR).

Availability of data and materials
The code is available at https://gitlab.com/ijs-e7/mol-traffic. The associated MOL-TR data set is available at https://doi.org/10.5281/zenodo.7119335.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 3 January 2023 Accepted: 28 August 2023
Published online: 07 September 2023

References

1. Lieu, H. C. (2000). Traffic estimation and prediction system. Technical report.
2. Liu, Y., Lyu, C., Zhang, Y., Liu, Z., Yu, W., & Qu, X. (2021). Deeptsp: Deep traffic state prediction model based on large-scale empirical data. *Communications in Transportation Research, 1*, 100012.
3. de Moraes Ramos, G., Mai, T., Daamen, W., Frejinger, E., & Hoogendoorn, S. P. (2020). Route choice behaviour and travel information in a congested network: Static and dynamic recursive models. *Transportation Research Part C: Emerging Technologies, 114*, 681–693. https://doi.org/10.1016/j.trc.2020.02.014
4. Pečar, M., & Papa, G. (2017). Transportation problems and their potential solutions in smart cities. In *2017 international conference on smart systems and technologies (SST)* (pp. 195–199). IEEE, Osijek, Croatia. https://doi.org/10.1109/SST.2017.8188694
5. Yin, X., Wu, G., Wei, J., Shen, Y., Qi, H., & Yin, B. (2021). Deep learning on traffic prediction: Methods, analysis and future directions. *IEEE Transactions on Intelligent Transportation Systems*.
6. Wijayaratna, K. P., Dixit, V. V., Denant-Boemont, L., & Waller, S. T. (2017). An experimental study of the online information paradox: Does en-route information improve road network performance? *PLoS ONE, 12*, 0184191. https://doi.org/10.1371/journal.pone.0184191
7. Guo, F., Polak, J. W., Krishnan, R., et al. (2018). Predictor fusion for short-term traffic forecasting. *Transportation Research Part C: Emerging Technologies, 92*, 90–100.
8. Bogaerts, T., Masegosa, A. D., Angarita-Zapata, J. S., Onieva, E., & Hellinckx, P. (2020). A graph cnn-lstm neural network for short and long-term traffic forecasting based on trajectory data. *Transportation Research Part C: Emerging Technologies, 112*, 62–77.
9. Jin, F., & Sun, S. (2008). Neural network multitask learning for traffic flow forecasting. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)* (pp. 1897–1901). IEEE.
10. Huang, W., Song, G., Hong, H., & Xie, K. (2014). Deep architecture for traffic flow prediction: Deep belief networks with multitask learning. *IEEE Transactions on Intelligent Transportation Systems, 15*(5), 2191–2201.
11. Patterson, K. (2011). An introduction to arma models. In *Unit Root Tests in Time Series* (pp. 68–122). London: Palgrave Macmillan.
12. Vu, K. M. (2007). *The ARIMA and VARIMA time series: Their modelings*. Ottawa: Analyses and Applications. AuLac Technologies Inc.
13. Peter, D., & Silvia, P. (2012) Arima vs. arimax–which approach is better to analyze and forecast macroeconomic time series. In *Proceedings of 30th international conference mathematical methods in economics* (Vol. 2, pp. 136–140).
14. Williams, B. M. (2001). Multivariate vehicular traffic flow prediction: Evaluation of arimax modeling. *Transportation Research Record, 1776*(1), 194–200.
15. Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting, 34*(4), 802–808.
16. Spiliotis, E., Makridakis, S., Semenoglou, A.-A., & Assimakopoulos, V. (2020). Comparison of statistical and machine learning methods for daily sku demand forecasting. *Operational Research* (pp. 1–25).
17. Barker, J. (2020). Machine learning in m4: What makes a good unstructured model? *International Journal of Forecasting, 36*(1), 150–155.
18. Luk, K. C., Ball, J. E., & Sharma, A. (2000). A study of optimal model lag and spatial inputs to artificial neural network for rainfall forecasting. *Journal of Hydrology, 227*(1–4), 56–65.
19. Mei, J., He, D., Harley, R., Habetler, T., & Qu, G. (2014). A random forest method for real-time price forecasting in new york electricity market. In *2014 IEEE PES general meeting| conference & exposition* (pp. 1–5). IEEE.
20. Kumar, M., & Thenmozhi, M. (2006). Forecasting stock index movement: A comparison of support vector machines and random forest. In *Indian institute of capital markets 9th capital markets conference paper*.
21. Medsker, L. R., & Jain, L. (2001). *Recurrent neural networks. Design and Applications*, *5*, 64–67.
22. Yun, S., Namkoong, S., Shin, S., Rho, J., & Choi, J. (1996). Application of a recurrent neural network to traffic volume forecasting. In *Intelligent transportation: realizing the future. Abstracts of the third world congress on intelligent transport systems ITS America* (1996).
23. Park, D.-C. (2009). Multiresolution-based bilinear recurrent neural network. *Knowledge and Information Systems, 19*(2), 235–248.
24. Gers, F.A., Eck, D., & Schmidhuber, J. (2002). Applying lstm to time series predictable through time-window approaches. In *Neural Nets WIRN Vietri-01* (pp. 193–200). Springer, Berlin.
25. Zhao, Z., Chen, W., Wu, X., Chen, P. C., & Liu, J. (2017). Lstm network: A deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems, 11*(2), 68–75.
26. O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.
27. Bai, S., Kolter, J.Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271.
28. Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2017). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. arXiv preprint arXiv:1707.01926.
29. Zhang, J., Zheng, Y., & Qi, D. (2017). Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Thirty-first AAAI conference on artificial intelligence*.
30. Li, G., Knoop, V. L., & van Lint, H. (2021). Multistep traffic forecasting by dynamic graph convolution: Interpretations of real-time spatial correlations. *Transportation Research Part C: Emerging Technologies, 128*, 103185.
31. Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2019). N-beats: Neural basis expansion analysis for interpretable time series forecasting. arXiv preprint arXiv:1905.10437.
32. Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting, 36*(3), 1181–1191.
33. Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting, 37*(4), 1748–1764.
34. Lim, B., Arik, S.O., Loeff, N., & Pfister, T. (2019). Temporal fusion transformers for interpretable multi-horizon time series forecasting. arXiv preprint arXiv:1912.09363.
35. Zhang, H., Zou, Y., Yang, X., & Yang, H. (2022). A temporal fusion transformer for short-term freeway traffic speed multistep prediction. *Neurocomputing*.
36. Dong, X., Lei, T., Jin, S., & Hou, Z. (2018). Short-term traffic flow prediction based on xgboost. In *2018 IEEE 7th data driven control and learning systems conference (DDCLS)* (pp. 854–859). IEEE.
37. Elsayed, S., Thyssens, D., Rashed, A., Jomaa, H. S., & Schmidt-Thieme, L. (2021). Do we really need deep learning models for time series forecasting? arXiv preprint arXiv:2101.02118.
38. Makridakis, S., & Hibon, M. (2000). The m3-competition: Results, conclusions and implications. *International Journal of Forecasting, 16*(4), 451–476.
39. Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting, 36*(1), 54–74.
40. Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2021). The m5 competition: Background, organization, and implementation. *International Journal of Forecasting*.
41. Cai, L., Janowicz, K., Mai, G., Yan, B., & Zhu, R. (2020). Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting. *Transactions in GIS, 24*(3), 736–755.

42. Tian, C., & Chan, W. K. (2021). Spatial-temporal attention wavenet: A deep learning framework for traffic prediction considering spatial-temporal dependencies. *IET Intelligent Transport Systems, 15*(4), 549–561.
43. Barros, J., Araujo, M., & Rossetti, R.J. (2015). Short-term real-time traffic prediction methods: A survey. In *2015 international conference on models and technologies for intelligent transportation systems (MT-ITS)* (pp. 132–139). IEEE.
44. Lana, I., Del Ser, J., Velez, M., & Vlahogianni, E. I. (2018). Road traffic forecasting: Recent advances and new challenges. *IEEE Intelligent Transportation Systems Magazine, 10*(2), 93–109.
45. Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research, 30*(1), 79–82.
46. Qi, J., Du, J., Siniscalchi, S. M., Ma, X., & Lee, C.-H. (2020). On mean absolute error for deep neural network based vector-to-vector regression. *IEEE Signal Processing Letters, 27*, 1485–1489.
47. De Myttenaere, A., Golden, B., Le Grand, B., & Rossi, F. (2016). Mean absolute percentage error for regression models. *Neurocomputing, 192*, 38–48.
48. Chai, T., & Draxler, R. R. (2014). Root mean square error (rmse) or mean absolute error (mae). *Geoscientific Model Development Discussions, 7*(1), 1525–1534.
49. Zheng, H., & Wu, Y. (2019). A xgboost model with weather similarity analysis and feature engineering for short-term wind power forecasting. *Applied Sciences, 9*(15), 3019.
50. Wahab, A., Tahir, M. A., Iqbal, N., Ul-Hasan, A., Shafait, F., & Kazmi, S. M. R. (2021). A novel technique for short-term load forecasting using sequential models and feature engineering. *IEEE Access, 9*, 96221–96232.
51. Schneider, T., Helwig, N., & Schütze, A. (2017). Automatic feature extraction and selection for classification of cyclical time series data. *tm-Technisches Messen* **84**(3), 198–206
52. Khadiev, K., & Safina, L. (2019). On linear regression and other advanced algorithms for electrical load forecast using weather and time data. *Journal of Physics: Conference Series* 1352, 012027 . IOP Publishing
53. Kotsiantis, S. B. (2013). Decision trees: A recent overview. *Artificial Intelligence Review, 39*(4), 261–283.
54. Shamrat, F. J. M., Ranjan, R., Md., K., Hasib, A. Y., & Siddique, A. H. (2021). Performance evaluation among id3, c4. 5, and cart decision tree algorithms. In *Pervasive computing and social networking: Proceedings of ICPCSN 2021* (Vol. 317, p. 127).
55. Ho, T.K. (1995). Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition* (Vol. 1, pp. 278–282). IEEE.
56. Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*(2), 123–140.
57. Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning, 63*(1), 3–42.
58. Rojas, R. (1996). *Neural networks: A systematic introduction*. Berlin: Springer.
59. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
60. Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*.
61. Polikar, R. (2012). Ensemble learning. In *Ensemble machine learning* (pp. 1–34). Springer, Boston, MA
62. Dong, X., Yu, Z., Cao, W., Shi, Y., & Ma, Q. (2020). A survey on ensemble learning. *Frontiers of Computer Science, 14*(2), 241–258.
63. Zhou, Z.-H. (2021). Ensemble learning. In *Machine learning* (pp. 181–210). Springer, Singapore.
64. Cerqueira, V., Torgo, L., & Mozetič, I. (2020). Evaluating time series forecasting models: An empirical study on performance estimation methods. *Machine Learning, 109*(11), 1997–2028.
65. Wainer, J., & Cawley, G. (2021). Nested cross-validation when selecting classifiers is overzealous for most practical applications. *Expert Systems with Applications, 182*, 115222.
66. Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2623–2631).
67. Koch, P., Golovidov, O., Gardner, S., Wujek, B., Griffin, J., & Xu, Y. (2018). Autotune: A derivative-free optimization framework for hyperparameter tuning. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 443–452).
68. Victoria, A. H., & Maragatham, G. (2021). Automatic tuning of hyperparameters using bayesian optimization. *Evolving Systems, 12*(1), 217–223.
69. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., & Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems* **32**.
70. Falcon, e.a. WA (2019). Pytorch lightning. GitHub. https://github.com/PyTorchLightning/pytorch-lightning**3**
71. Mölder, F., Jablonski, K.P., Letcher, B., Hall, M.B., Tomkins-Tinch, C.H., Sochat, V., Forster, J., Lee, S., Twardziok, S.O., & Kanitz, A., et al. (2021). Sustainable data analysis with snakemake. *F1000Research* **10**.
72. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.
73. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*(1), 1929–1958.
74. Au, T. C. (2018). Random forests, decision trees, and categorical predictors: the "absent levels" problem. *The Journal of Machine Learning Research, 19*(1), 1737–1766.

Publisher's Note