engineering proceedings

MDPI

Proceeding Paper

Forecasting Traffic Flow Using Machine Learning Algorithms †

Makhamadaziz Rasulmukhamedov, Timur Tashmetov and Komoliddin Tashmetov *

Department of Information Systems and Technologies at Transport, Tashkent State Transport University, Tashkent 100093, Uzbekistan; mrasulmuxamedov@list.ru (M.R.); tima260491@gmail.com (T.T.)

* Correspondence: tashmetov1993@gmail.com

† Presented at the International Conference on Electronics, Engineering Physics and Earth Science (EEPES'24), Kavala, Greece, 19–21 June 2024.

Abstract: This article is dedicated to the study of traffic flow forecasting at the intersection of Bogishamol Street in Tashkent, Uzbekistan. In the context of the rapid growth of vehicular traffic and frequent congestion, developing effective forecasting models is a pressing task that will help optimize traffic flow management. The research examines and analyzes various machine learning methods, such as decision trees, random forests, and gradient boosting, for predicting traffic intensity. The data for the models was collected using video cameras installed at the intersection which provided accurate and up-to-date traffic flow information. The main focus of the study is on the comparative analysis of the performance of these methods. The comparison was made using various evaluation metrics, such as the coefficient of determination (R^2), mean squared error (MSE), and mean absolute error (MAE). These metrics allowed for an objective assessment of the accuracy and effectiveness of each method in the context of traffic flow prediction. The results of the study showed that the gradient boosting model demonstrated the best performance among the methods considered. It achieved the highest R^2 values and the lowest MSE and MAE values, indicating its high accuracy and ability to adequately predict changes in traffic flows. The decision tree and random forest models also showed good results but were outperformed by gradient boosting in key indicators. The findings have significant practical implications. They can be used to develop intelligent traffic management systems aimed at increasing the capacity of roads and intersections. This, in turn, can help reduce congestion, lower emissions of harmful substances into the atmosphere, and decrease economic costs associated with traffic delays.

Keywords: traffic flow forecasting; machine learning; decision tree; random forest; gradient boosting; coefficient of determination; mean squared error; mean absolute error; traffic management; intelligent systems

check for updates

Citation: Rasulmukhamedov, M.; Tashmetov, T.; Tashmetov, K. Forecasting Traffic Flow Using Machine Learning Algorithms. *Eng. Proc.* **2024**, *70*, 14. https://doi.org/10.3390/engproc2024070014

Academic Editor: Grigor Mihaylov

Published: 31 July 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In the rapidly developing city of Tashkent, the issues of managing traffic flows are becoming increasingly significant and critical. Every day, a huge number of vehicles, totaling 900 thousand units, enter the city's roads. This volume not only exceeds the designed capacity of the infrastructure but also raises serious questions about the sustainability and efficiency of urban traffic.

With a population exceeding three million people and an annual increase in motor vehicles of 76 thousand units, the challenges of ensuring the efficiency of urban transport have become an integral part of life for the residents of Tashkent. Dissatisfaction with the lack of mobility, long traffic jams, economic losses, and a negative environmental impact are all aspects that pose challenges requiring innovative and effective solutions.

To address the aforementioned traffic management issues in Tashkent, it is critically important to use innovative approaches, including the application of machine learning algorithms for forecasting traffic flows. This method allows for the analysis of vast amounts of data and the identification of patterns in traffic movement, which in turn enables the

optimization of traffic management and reduces its negative impact on the environment and the city's economy.

Forecasting traffic flow using machine learning algorithms allows not only for predicting traffic volumes based on historical data but also for considering various external factors such as weather conditions, time of day, and calendar events. This enables the creation of more accurate forecasts, which in turn helps make more informed decisions regarding the management of traffic flows.

2. Materials and Methods

The challenge of forecasting traffic flows and using these to develop solutions for managing transportation systems is considered one of the most relevant topics in the modern world. Many scientists conduct theoretical and practical research in this area, leading to the development of unique algorithms, methods, and models. In this work, we will also analyze articles related to our research, including works [1,2].

In articles [1–3], research was conducted on short-term traffic forecasting using extreme gradient boosting (XGBoost). The authors of [3] note that fast and accurate short-term traffic flow forecasting is an important condition for traffic analysis and management. In [4], the authors investigated the issue of road safety in intelligent transportation systems (ITS) focusing on intersections. They propose using nonparametric, nonlinear ensemble models of decision trees to forecast traffic volume. It is also noted that intersections are the most complex part of the road network, and most accidents are related to intersections.

Article [5] is dedicated to forecasting traffic flow in work zones on roads. It notes that most existing traffic flow forecasting models do not consider the peculiarities of work zones, which create conditions that are different from both normal operating conditions and incident conditions. The study developed four models for forecasting traffic flow in the planned work zones. These models include random forests, regression trees, multilayer feedforward neural networks, and nonparametric regression. The authors investigated both long-term and short-term traffic flow forecasting. Long-term forecasting involves predicting 24 h ahead using historical traffic data, while short-term forecasting involves predicting 1 h, 45 min, 30 min, and 15 min ahead using real-time traffic data.

The models were evaluated using data from work zones on two types of roads: a highway and an arterial road in St. Louis, Missouri, USA. The research revealed that the random forest model provided the most accurate forecasts for both long-term and short-term traffic flow in work zones.

In addition, articles [6,7] propose a new hybrid model (CEEMDAN-XGBoost) for forecasting traffic flow at the lane level, based on complete ensemble empirical mode decomposition with adaptive noise (CEEMDAN) and extreme gradient boosting (XGBoost). The CEEMDAN method is used to decompose the original traffic flow data into several intrinsic mode functions components and one residual component. Then, XGBoost models are trained and make predictions on the decomposed components, respectively. The final forecasting results are obtained by integrating the forecasting results of the XGBoost methods. For illustrative purposes, traffic flow data at the lane level obtained from remote microwave traffic sensors installed on the 3rd ring road in Beijing are used to evaluate the effectiveness of the CEEMDAN-XGBoost model.

In some literature, the issue of forecasting traffic flows when data is insufficient is studied. In [8–11], the problem of forecasting traffic flow in intelligent transportation systems (ITS) applications is considered. It is noted that using an autoregressive integrated moving average (ARIMA) or seasonal ARIMA (SARIMA) for traffic flow forecasting requires a large amount of data to develop the model, which may be impossible in cases of insufficient data.

To address this problem, the authors proposed and evaluated a forecasting scheme based on the Kalman filter (KF) method, which requires only a limited amount of input data.

In this study on traffic flow forecasting, a variety of methods and algorithms were considered, ranging from linear regression to ensemble models. However, it is known that many linear models do not have sufficient accuracy for forecasting. For this reason,

nonlinear methods and models were extensively studied, collectively referred to as additive models. Additive models, including various types of neural networks and decision trees, are able to capture nonlinear dependencies between variables and provide more accurate results.

In the context of regression, the generalized additive model (GAM) is formulated as:

$$E(Y|X_1, X_2, X_3, \ldots X_n) = a + f_1(X_1) + f_2(X_2) + f_3(X_3) \ldots f_n(X_n) \quad (1)$$

where $E(Y|X_1, X_2, X_3, \ldots X_n)$—the conditional expectation of the dependent variable Y given the values of predictors $X_1, X_2, X_3, \ldots X_n$, a—constant (intercept) representing the baseline level of the dependent variable Y when all predictors are zero, $X_1, X_2, X_3, \ldots X_n$—are predictors, Y—response, f_n—undefined smooth (nonparametric) functions.

In our work, we have focused on the nonlinearity of the problem and explored state-of-the-art machine learning methods for predicting nonlinear dependencies. We delved into the study of decision trees, random forests, and ensemble methods, as they are effective in solving such tasks. Particularly noteworthy are decision trees, which find application in many areas due to their simplicity and ease of interpreting results. Like any other method, decision trees have their drawbacks, such as limited applicability and a tendency to over fit [4,12]. The results of several studies [4,12–15] indicate significant improvements in addressing the issues of limited performance and reliability that are inherent in decision trees. Based on these studies, ensemble methods have been developed, which combine multiple decision trees with various voting methods to select the optimal target label. These enhancements help increase the accuracy and stability of predictions, making the model more efficient and reliable.

A detailed examination of decision tree methods is discussed in [4,13,16,17]. In [17,18], four of the most widely used algorithms, CART (classification and regression trees), C4.5, CHAID (chi-squared automatic interaction detection), and QUEST (quick, unbiased, efficient, statistical tree), are compared. [16] compares the algorithms ID3 and C4.5, C4.5 and C5.0, and C5.0 and CART. The research results indicate that the best algorithm for machine learning is C4.5. However, it is known that this algorithm is mainly used in classification tasks. In our case, we will use the CART algorithm since our task pertains to regression.

Having studied the mentioned sources, we can move on to how to construct a decision tree algorithm for regression tasks. To do this, we need to have data, including training and test sets. Let's assume that we have a set of input data X and corresponding output vectors Y, such that $(x_i,\ y_i)$ for $i = 1,\ 2,\ \ldots,\ n$ where $x_i = \{x_{i1}, x_{i2}, \ldots, x_{ip}\}$ with p features. Then, we should split the data based on some feature x_{ip} to divide our data into two parts X_l(left) and X_r(right), thus creating the tree structure represented in Figure 1.

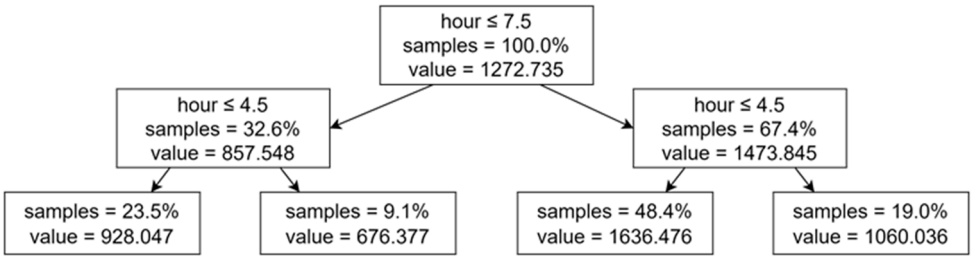


Figure 1. Structural representation of the decision tree. This decision tree represents a regression model used to predict some target variable based on two features: hour and week.

To select nodes for splitting the most informative features, we need to define the objective function that we will optimize using the tree learning algorithm. In this case, our objective function is to maximize the information gain at each split, which is defined as follows (2):

$$IG(R_m,\ p) = I(R_m) - \frac{|R_l|}{|R_m|} I(R_l) - \frac{|R_r|}{|R_m|} I(R_r), \quad (2)$$

where $IG(R_m,\ p)$ is the information gain when partitioning set R_m by feature p, $I(R_m)$ is the impurity of the original set R_m before the split, $|R_l|$ and $|R_r|$ are the number of instances in subsets R_l and R_r after the split, respectively, $|R_m|$ is the total number of instances in the original set R_m, $I(R_l)$ and $I(R_r)$ are the impurities of subsets R_l and R_r after the split, respectively.

In regression tasks, the mean squared error (MSE) is usually chosen as the impurity.

$$I(R) = \frac{1}{|R|}\sum_{i=1}^{n}\left(y_i - \hat{f}(x)\right)^2, \tag{3}$$

where $I(R)$ is the impurity of node R, $|R|$ is the number of instances in node R, y_i is the actual value of the target variable for instance i, $\hat{f}(x)$ is the predicted value of the target variable for instance i.

As mentioned above, ensemble methods have been developed to improve the drawbacks of decision trees, which have been studied in articles [4,6,11,12,14,19–25].

Bagging is an ensemble machine learning method used to improve the stability and generalization of machine learning models. The idea behind bagging is to use not just one model, but several $b_1,\ b_2, \ldots,\ b_n$ models trained on different subsets of data $R_1,\ R_2, \ldots,\ R_n$, to improve the quality of the model's prediction. Subsequently, the results of all models are combined, for example, by averaging (4), to obtain a more stable and accurate prediction [22,26].

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} b_i(x), \tag{4}$$

where n is the number of base models in bagging, $b_i(x)$ is the predicted value of the target variable for the object x using the i-th base model.

Random forest is also an ensemble machine learning method used for both classification and regression tasks. It is based on the idea of combining multiple decision trees into one model [20,24,27,28].

To build a random forest, several random subsamples $R_1,\ R_2, \ldots,\ R_n$ with replacement are first formed from the original dataset R. Then, a separate decision tree $h_1,\ h_2, \ldots,\ h_n$ is built for each subsample. When building each node of the tree, a random subset of features is selected. After all the trees are built, their results are combined, for example, by majority voting for classification tasks or averaging for regression tasks, to obtain the final prediction result of the random forest model.

Boosting is a machine learning method used to improve the performance of models by combining several weak models into one strong model. The main idea of boosting is to sequentially train models on data, focusing on examples where previous models made mistakes. Each new model attempts to correct the errors of the previous one, and, as a result, the combination of all the models allows for a more accurate and generalizable model [3,23,25,26].

Boosting can be used with various base models, but decision trees are most commonly used (5).

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha_i b_i(x), \tag{5}$$

where α_i is the weight coefficient for the i-th model, and $b_i(x)$ is the prediction of the i-th base model for object x.

In boosting, the main goal is to minimize the error functional (quality criterion) by sequentially adding new models that correct the errors of the previous ones (6).

$$I(\alpha, b) = \sum_{i=1}^{n} L\left(y_i, \hat{f}(x)\right) \to \min, \tag{6}$$

where $I(\alpha, b)$ is the loss function, n is the total number of observations in the dataset, $L\left(y_i, \hat{f}(x)\right)$ is the loss function dependent on the true value y_i and the predicted value $\hat{f}(x)$. Parameters α and b are those values adjusted to minimize this loss function.

Another method is the gradient boosting algorithm, which is widely used in classification and regression tasks. The difference between gradient boosting and classical boosting is that gradient boosting uses gradient descent to tune the model parameters, while classical boosting uses the method of reweighting objects.

3. Problem Formulation

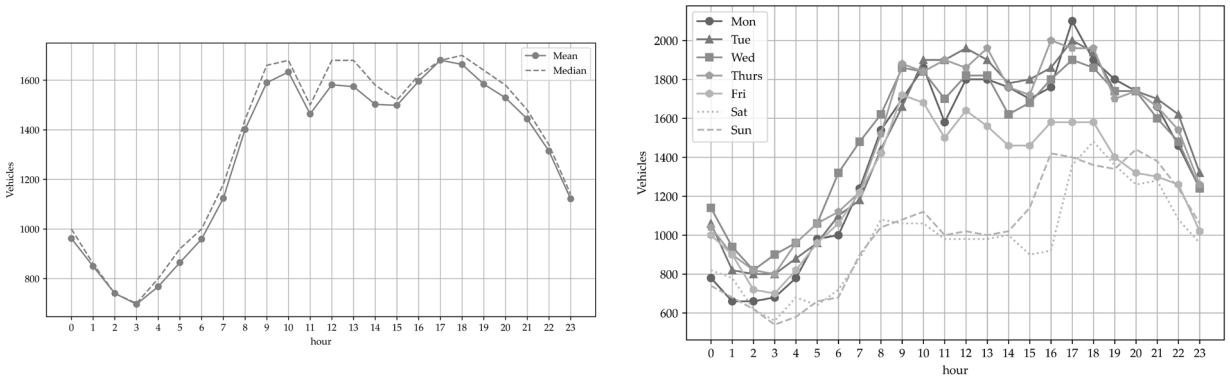
In this study, we consider an intersection located on Bogishamol Street in Tashkent, Uzbekistan, as the object of research. To manage this intersection using an intelligent system, we need parameters of traffic flow in these directions. These parameters include traffic intensity, traffic density, and average speed.

The information necessary for forecasting traffic flows can be obtained from cameras installed along this road. Here are the data on vehicles moving from west to east, collected from video cameras and presented in Table 1 below.

Table 1. Dataset traffic flow.

Index	Datetime	Junction	Vehicles	Day	Week	Month	Year	Hour
0	1 January 2023 00:00:00	1	340	1	7	1	2023	0
1	1 January 2023 01:00:00	1	320	1	7	1	2023	1
2	1 January 2023 02:00:00	1	320	1	7	1	2023	2
3	1 January 2023 03:00:00	1	280	1	7	1	2023	3
4	1 January 2023 04:00:00	1	280	1	7	1	2023	4
5	1 January 2023 05:00:00	1	360	1	7	1	2023	5
...	...	...	...	...	...	...	...	...

The data were collected from all four directions at the intersection, with a total of 17,373 records. However, we are analyzing information obtained only from one direction when developing the forecasting algorithm. The average and median values of daily traffic intensity and the average traffic volume on weekdays and weekends are presented in (Figure 2) below.



(**a**) (**b**)

Figure 2. Average and median values of daily traffic intensity (**a**); average traffic volume on weekdays and weekends (**b**).

The analysis of the graph in Figure 2a, which represents the mean and median values of traffic flow by hour of the day, indicates that the minimum road congestion occurs during nighttime, approximately from 0 to 5 h, when the number of vehicles is around 500–900. From 6 a.m., there is a significant increase in traffic flow, reaching a peak between 8 and 10 a.m. with the number of vehicles around 1500–1660. After this, there is a slight decrease. However, starting from 11 a.m., a new rise begins, reaching its maximum around 5–6 p.m. (approximately 1600–1700 vehicles). In the evening, after 6 p.m., the number of vehicles gradually decreases until the end of the day.

The second graph in Figure 2b demonstrates the differences in traffic flow by day of the week for each hour. On weekdays (Monday to Friday), there are two distinct peaks: the morning peak around 8 a.m. and the evening peak around 5 p.m., corresponding to the start and end of the workday. On Saturday and Sunday, the number of vehicles is significantly lower compared to weekdays. On weekends, traffic flow remains relatively stable throughout the day, with a slight increase around noon, especially on Sunday.

Thus, the data show that the highest road congestion occurs during the morning and evening hours of weekdays, which is associated with rush hour traffic. On weekends, the traffic flow is significantly lower and more evenly distributed throughout the day, indicating reduced activity on these days.

To achieve a more precise data analysis, it is essential to utilize a correlation matrix. Correlation is a crucial tool in data analytics as it allows us to determine the degree of association between the changes in different variables. In the context of machine learning, correlation analysis helps identify significant relationships between variables, which aids in simplifying the model, removing irrelevant data, and accelerating the training process. For instance, if the number of cars significantly varies depending on the day of the week and the time of day, as shown in the correlation matrix (Figure 3), these variables will be critical for predicting traffic flow. Correlation analysis also plays a vital role in preventing multicollinearity, thereby enhancing the stability and interpretability of the model.

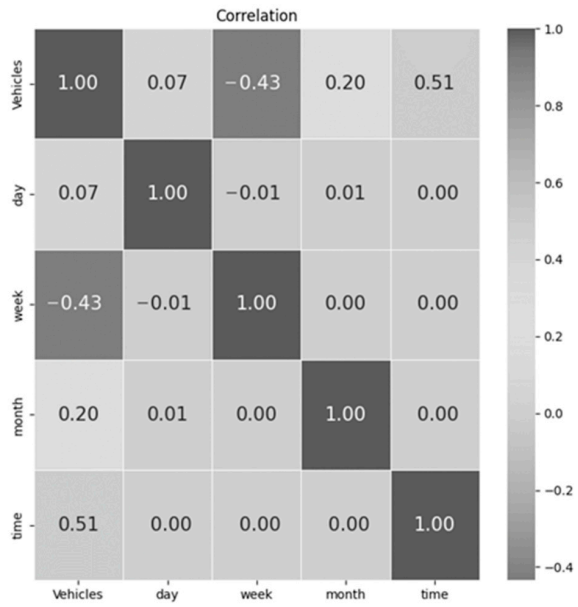


Figure 3. Correlation matrix showing the relationships between various features: vehicles, day, week, month, and time.

After identifying significant relationships, the data is divided into training and test sets in a ratio of 80/20 or 70/30. This ensures a balance between having sufficient data for training and for evaluating the model's performance. Model evaluation can be performed using metrics such as mean squared error (MSE), mean absolute error (MAE), and the coefficient of determination (R^2). These metrics provide a quantitative assessment of

the model's accuracy and predictive capabilities, enabling researchers to determine the effectiveness of the model in capturing the underlying data patterns.

In this study, we used machine learning algorithms such as decision trees, random forests, and ensemble methods for forecasting. This choice was made because these models have proven to be simple to use in practice and have demonstrated good results in many areas.

4. Results

In this section, an assessment and comparison of the results are conducted. To evaluate our models, we use the aforementioned metrics. For building traffic flow forecasting models, we use the Python programming language version 3.11.5. This work is carried out in the Anaconda environment, which provides a convenient package and virtual environment management. In the development process, we make use of several libraries, including scikit-learn (sklearn), numpy, pandas, and matplotlib.

Initially, we will forecast the traffic flow using a decision tree with a maximum branching depth (max_depth) of 5, 10, 100, and 200 (Figure 4). We will evaluate the forecast quality on a sample that will be split using the train_test_split method, as well as on a unique sample that contains data for 24 h and was not used in the model training process.

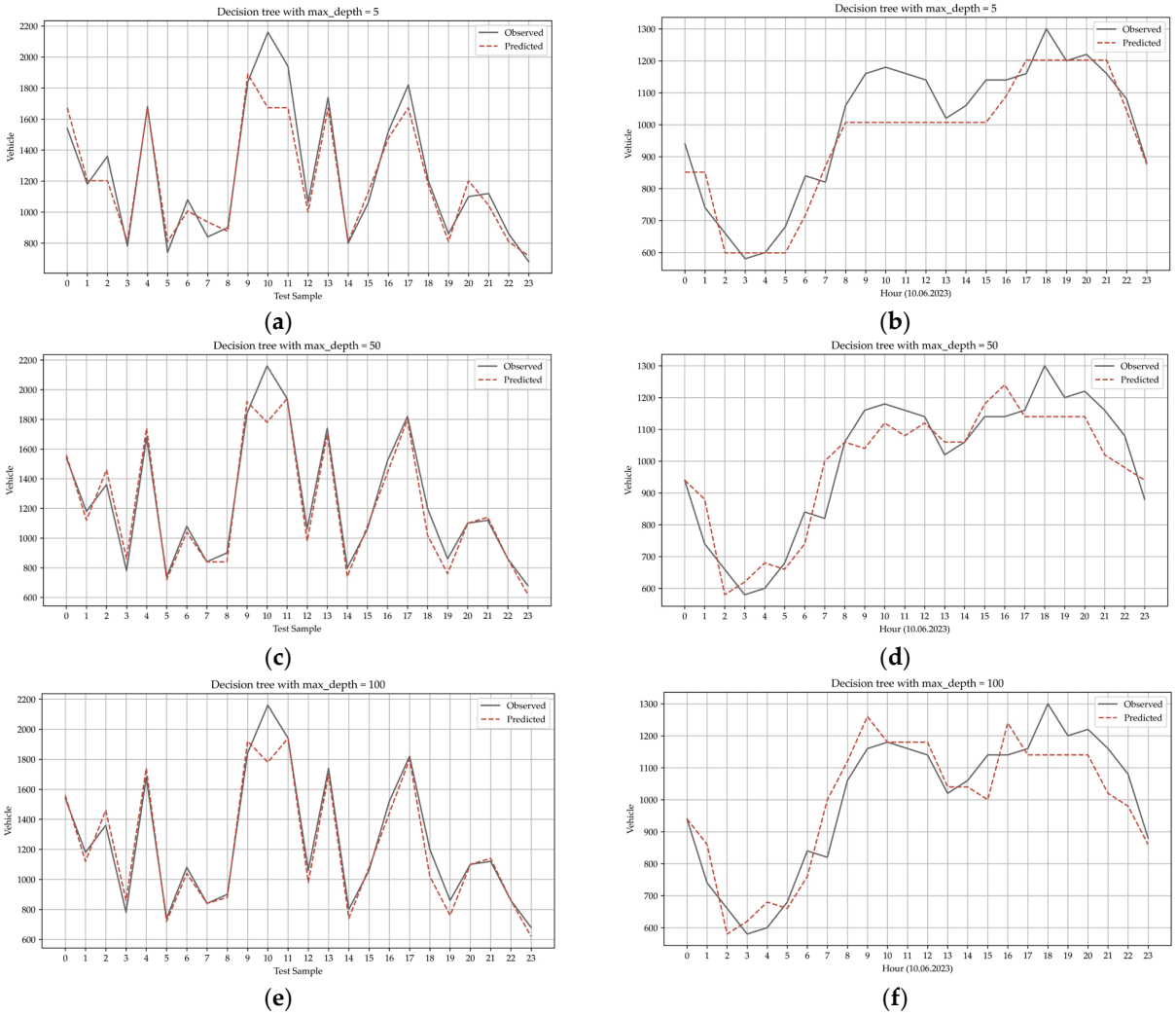


Figure 4. Prediction results using the decision tree algorithm (**a**) test dataset with max_depth = 5; (**b**) validation dataset with max_depth = 5; (**c**) test dataset with max_depth = 50; (**d**) validation dataset with max_depth = 50; (**e**) test dataset with max_depth = 100; (**f**) validation dataset with max_depth = 100.

Figure 4 presents the results of the decision tree model with depths of 5, 50, and 100 for a visual comparison. Graphs a, c, and e show the prediction results based on the test dataset, where the black lines represent actual values, and the dashed lines indicate predicted values. Graphs b, d, and f illustrate the model's performance on a validation dataset collected on 10 June 2023, which was not used in either the training or testing datasets.

To understand the impact of tree depth on model performance, we evaluated a decision tree regressor with varying maximum depths using two datasets: a test sample and a validation dataset from 10 June 2023. The performance metrics include the coefficient of determination (R^2), mean squared error (MSE), mean absolute error (MAE), and training time in seconds. The results are summarized in Table 2 below.

Table 2. Decision tree model result.

Maximum Tree Depth	Data	Metrics			Training Time (s)
		R^2	MSE	MAE	
5	Test Sample	0.89	18,110.99	89.87	0.0034
	Check data (10 June 2023)	0.83	7656.67	70.50	
50	Test Sample	0.94	10,233.33	65.0	0.0194
	Check data (10 June 2023)	0.83	7666.66	71.66	
100	Test Sample	0.94	10,100.00	63.33	0.0285
	Check data (10 June 2023)	0.83	7500.00	70.00	
200	Test Sample	0.94	10,266.66	65.00	0.0389
	Check data (10 June 2023)	0.80	8733.33	81.66	

For the test sample, the following trends were observed. As the tree depth increased, the R^2 value improved from 0.89 at depth 5 to 0.94 at depths 50, 100, and 200, indicating a better fit of the model to the data. MSE and MAE values significantly decreased with increasing depth, from 18,110.99, and 89.87 at depth 5 to approximately 10,233.33, and 65.00 at depths 50, 100, and 200, respectively. This shows that deeper trees provide more accurate predictions. As expected, the training time increased with tree depth, rising from 0.0034 s at depth 5 to 0.0389 s at depth 200. Although the increase in training time is noticeable, it is relatively small compared to the improvements in the performance metrics.

For the control data, the results were somewhat different. The R^2 value remained relatively stable around 0.83 at depths 5, 50, and 100, but slightly decreased to 0.80 at depth 200. This indicates that while deeper trees fit the training data better, they may not generalize as well to new data. MSE values showed minor changes: 7656.67 at depth 5 and a slight decrease to 7500.00 at depth 100, but MSE increased to 8733.33 at depth 200, indicating potential overfitting. MAE values remained stable around 70 at depths 5, 50, and 100, but increased to 81.66 at depth 200, further suggesting a decline in performance in the new data.

From this analysis, we can conclude that increasing the maximum tree depth generally improves model performance on the training and test samples, as evidenced by higher R^2 values and lower MSE and MAE values. However, the control data results indicate that excessively deep trees (depth 200) may lead to overfitting, reducing the model's ability to generalize to new data. The optimal tree depth should balance between training accuracy and generalization ability, with depths around 50–100 providing the best overall performance for our datasets.

Next, we will consider ensemble methods, starting with random forest. For the random forest model, we will tune two main hyperparameters: the maximum depth of the trees in the forest (max_depth) and the number of trees in the forest (n_estimators). To evaluate the forecasting quality, we will use the methods mentioned earlier, including R^2 (coefficient of determination), MSE, and MAE. The research results are presented in Figure 5 and in Table 3. The figure shows not all stages, but only those non-visual changes when tuning the hyperparameter.

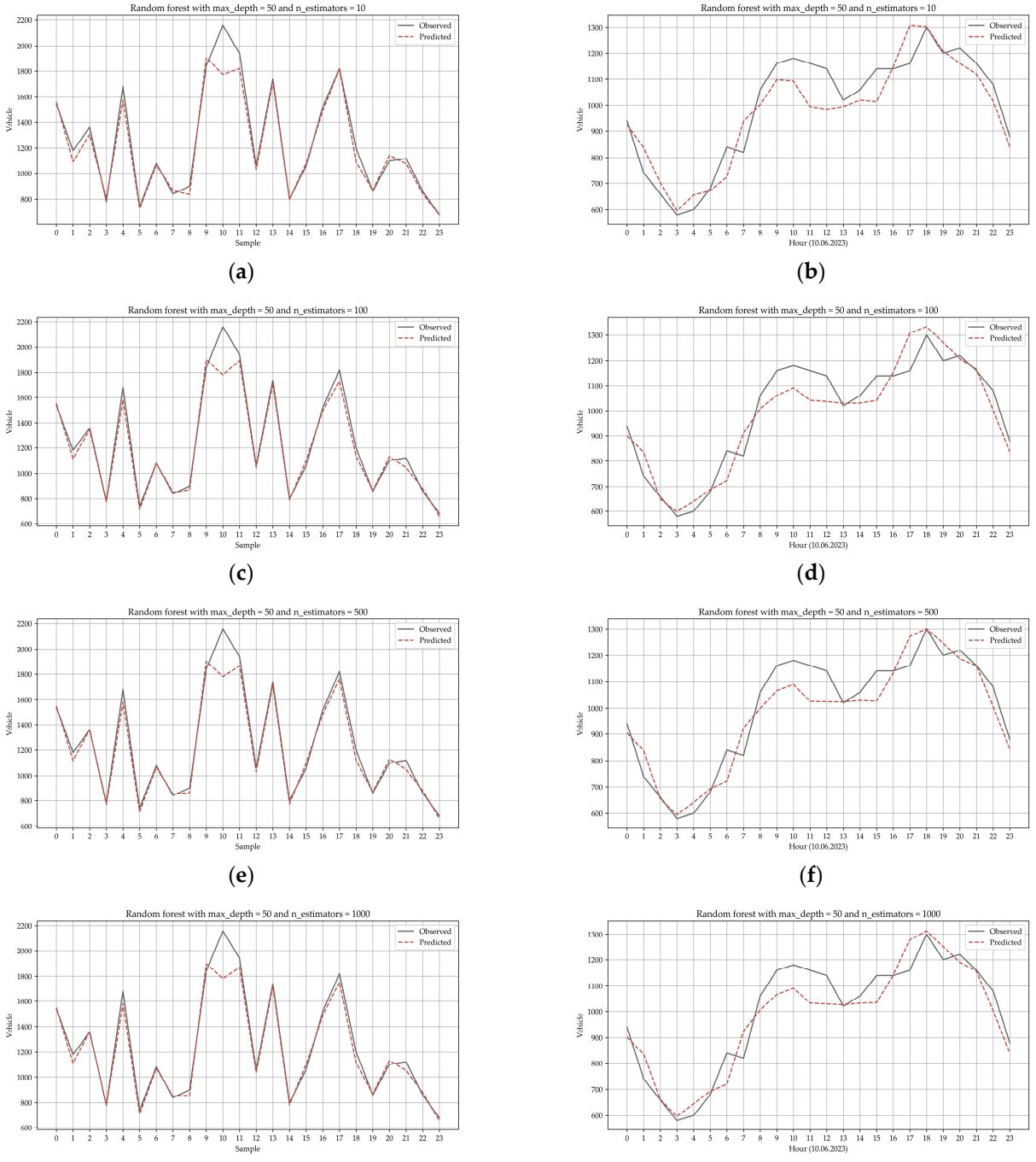


Figure 5. Prediction results using the random forest algorithm. (**a**,**c**,**e**,**g**): Predictions on the test dataset with max_depth = 50 and n_estimators = 10, 100, 500, 1000; (**b**,**d**,**f**,**h**): Predictions on the validation dataset with max_depth = 50 and n_estimators = 10, 100, 500, 1000.

Figure 5 presents 8 images depicting the predictions of the random forest model with a maximum tree depth of 50 and the number of trees varying from 10 to 1000. Graphs a, c, e, and g show predictions on the test sample, while graphs b, d, f, and h display predictions on the validation sample, similar to the decision tree models. A more detailed analysis was conducted based on the data presented in Table 3.

Table 3. Random forest model result.

Maximum Tree Depth	Data	Metrics			Training Time (s)	Number of Trees
		R^2	MSE	MAE		
5	Test Sample	0.90	17,565.05	81.47	0.0311	10
	Check data (10 June 2023)	0.84	7038.63	64.38		
50	Test Sample	0.95	8891.83	54.08	0.1154	
	Check data (10 June 2023)	0.85	6682.83	65.08		
100	Test Sample	0.95	7976.50	54.58	0.1221	
	Check data (10 June 2023)	0.83	7529.33	75.00		
200	Test Sample	0.94	9952.83	62.41	0.1279	
	Check data (10 June 2023)	0.87	5721.16	58.75		
5	Test Sample	0.90	17,075.49	80.45	0.2513	100
	Check data (10 June 2023)	0.86	5876.40	59.33		
50	Test Sample	0.95	7849.14	49.04	0.9674	
	Check data (10 June 2023)	0.88	5298.39	59.35		
100	Test Sample	0.95	7985.61	51.82	1.0987	
	Check data (10 June 2023)	0.87	5643.26	61.15		
200	Test Sample	0.95	7594.58	48.84	1.4277	
	Check data (10 June 2023)	0.88	5181.05	59.10		
5	Test Sample	0.90	17,360.18	81.02	1.3963	500
	Check data (10 June 2023)	0.86	6015.21	60.00		
50	Test Sample	0.95	7961.16	50.78	5.7024	
	Check data (10 June 2023)	0.88	5253.31	57.68		
100	Test Sample	0.95	7553.86	48.86	6.5222	
	Check data (10 June 2023)	0.88	5214.09	57.93		
200	Test Sample	0.95	7926.87	51.12	8.1768	
	Check data (10 June 2023)	0.88	5232.73	58.07		
5	Test Sample	0.90	17,337.28	81.18	4.0789	1000
	Check data (10 June 2023)	0.86	5882.79	59.38		
50	Test Sample	0.95	7955.41	50.45	12.1822	
	Check data (10 June 2023)	0.88	5181.50	57.78		
100	Test Sample	0.95	7875.57	50.86	12.6484	
	Check data (10 June 2023)	0.88	5126.79	57.56		
200	Test Sample	0.95	7786.01	49.76	13.4858	
	Check data (10 June 2023)	0.87	5533.47	59.43		

Analyzing the results, it can be observed that increasing the number of trees in the forest improves the prediction metrics. For example, with 10 trees and a tree depth of 50, the coefficient of determination R^2 on the test sample is 0.95, the mean squared error (MSE) is 8891.83, and the mean absolute error (MAE) is 54.08. The training time for this configuration is 0.1154 s. When increasing the number of trees to 100, R^2 remains at 0.95, but MSE decreases to 7849.14, and MAE to 49.04, with a training time of 0.9674 s. Similar improvements are observed with further increases in the number of trees to 200 and 1000, although increasing the number of trees beyond 100 does not significantly improve the metrics but substantially increases the training time.

The validation sample also shows improved prediction accuracy with an increasing number of trees. For instance, with 10 trees, R^2 is 0.85, MSE is 6682.83, and MAE is 65.08. When increasing the number of trees to 100, R^2 rises to 0.88, MSE decreases to 5298.39, and MAE to 59.35, with a training time of 0.2513 s. With further increases to 1000 trees, the metrics on the validation sample show consistently high results, confirming the overall trend of improved model accuracy with more trees, although the training time also increases, which must be considered when choosing model parameters.

Therefore, based on the obtained results, it can be concluded that the optimal parameters for the random forest model include a maximum tree depth of 50 and a number of trees of 100. These parameters provide high prediction accuracy on both test and validation

samples while maintaining acceptable training time. However, it should be noted that further increasing the number of trees to 1000 does not significantly improve the metrics, making it more rational to use 100 trees to avoid excessive training time without substantial improvement in prediction quality.

The last model that is most commonly used for predicting nonlinear regression tasks is gradient boosting. In gradient boosting, we need to tune 3 parameters: the maximum depth of the trees, the number of trees in the ensemble, and the learning rate. The results of the gradient boosting research are presented in Figure 6 and Table 4.

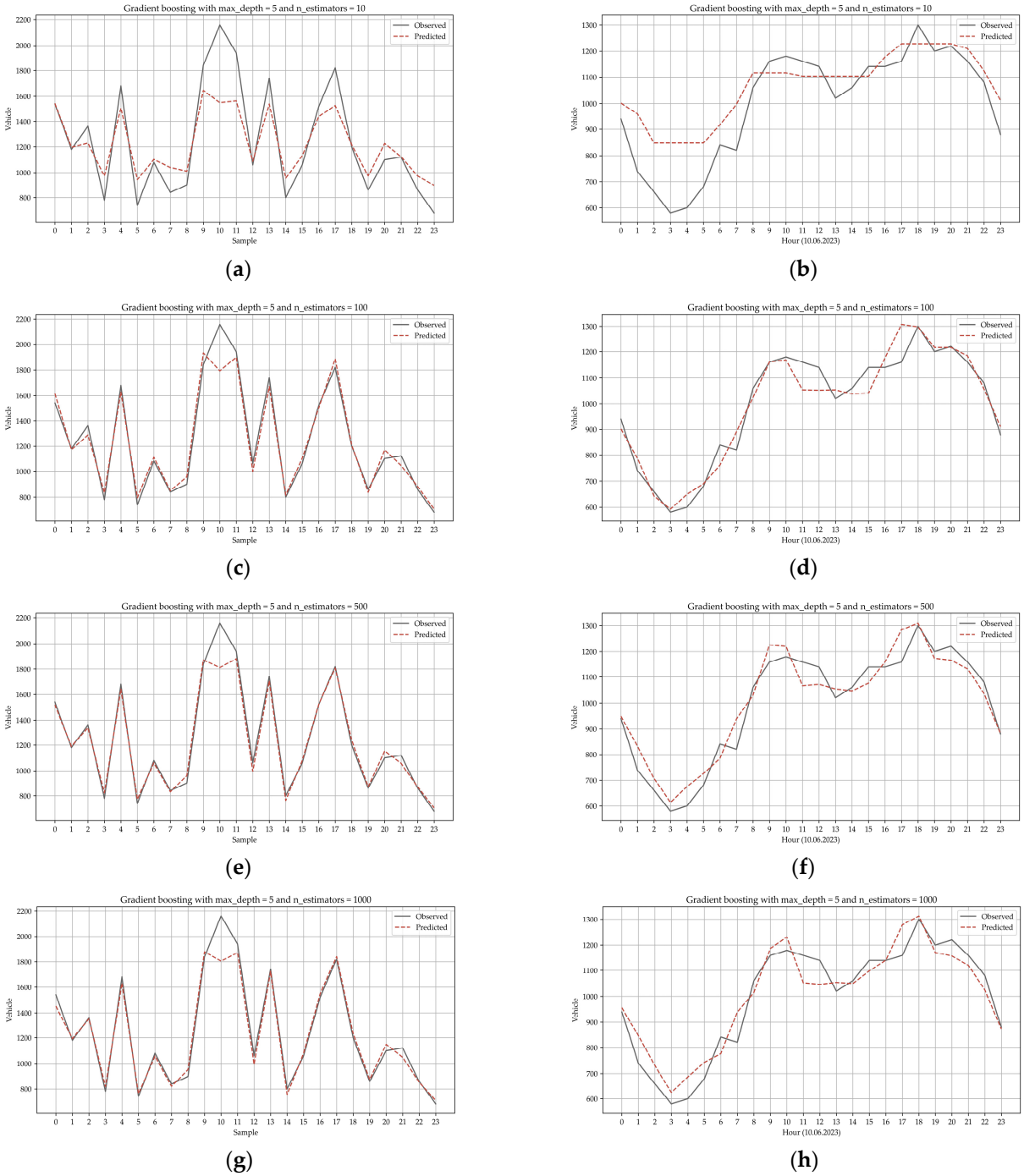


Figure 6. Prediction results using the gradient boosting algorithm. (**a**,**c**,**e**,**g**): Predictions on the test dataset with max_depth = 5 and n_estimators = 10, 100, 500, 1000; (**b**,**d**,**f**,**h**): Predictions on the validation dataset with max_depth = 5 and n_estimators = 10, 100, 500, 1000.

Table 4. Gradient boosting model result.

Maximum Tree Depth	Data	Metrics			Training Time (s)	Number of Trees
		R^2	MSE	MAE		
5	Test Sample	0.77	40,839.59	152.09	0.0336	10
	Check data (10 June 2023)	0.68	14,301.64	94.40		
50	Test Sample	0.83	29,548.97	133.62	0.2292	
	Check data (10 June 2023)	0.58	18,588.58	102.10		
100	Test Sample	0.83	29,554.94	133.61	0.2032	
	Check data (10 June 2023)	0.53	21,041.60	113.48		
200	Test Sample	0.83	29,620.03	133.85	0.3072	
	Check data (10 June 2023)	0.58	18,850.12	103.04		
5	Test Sample	0.95	8164.95	58.61	0.2658	100
	Check data (10 June 2023)	0.93	3111.93	41.69		
50	Test Sample	0.94	10,142.93	64.51	1.5672	
	Check data (10 June 2023)	0.83	7558.89	71.70		
100	Test Sample	0.94	10,169.79	64.64	1.6543	
	Check data (10 June 2023)	0.81	8572.35	80.74		
200	Test Sample	0.94	10,154.36	64.65	1.6959	
	Check data (10 June 2023)	0.84	7197.00	73.87		
5	Test Sample	0.96	6300.24	44.27	1.4633	500
	Check data (10 June 2023)	0.92	3595.85	50.44		
50	Test Sample	0.94	10,148.91	64.42	3.6159	
	Check data (10 June 2023)	0.83	7307.61	71.42		
100	Test Sample	0.94	10,162.14	64.49	3.6710	
	Check data (10 June 2023)	0.83	7425.88	71.87		
200	Test Sample	0.94	10,120.40	64.04	5.049	
	Check data (10 June 2023)	0.81	8381.52	77.81		
5	Test Sample	0.96	6917.45	48.90	4.0559	1000
	Check data (10 June 2023)	0.90	4361.04	55.85		
50	Test Sample	0.94	10,127.18	64.22	4.1851	
	Check data (10 June 2023)	0.85	6560.22	69.74		
100	Test Sample	0.94	10,170.76	64.57	4.2626	
	Check data (10 June 2023)	0.82	7809.50	72.91		
200	Test Sample	0.94	9952.83	62.41	4.7457	
	Check data (10 June 2023)	0.87	5721.16	58.75		

Analysis of Table 4 shows that with a maximum tree depth of 5 and n_estimators set to 10, the model achieves an R^2 value of 0.77 on the test sample and 0.68 on the check data. MSE and MAE values are 40,839.59 and 152.09, respectively, for the test sample, indicating moderate accuracy with relatively high errors. As the tree depth increases to 50, the R^2 value improves to 0.83 for the test sample and 0.58 for the check data, with MSE decreasing to 29,548.97 and MAE to 133.62 for the test sample, showing improved performance but still relatively high errors for the check data.

When the maximum tree depth is set to 5 and the number of trees increases to 100, the model's performance significantly improves. The R^2 value reaches 0.95 for the test sample and 0.93 for the check data, with MSE dropping to 8164.95 and MAE to 58.61 for the test sample. This demonstrates the effectiveness of increasing the number of trees to improve model accuracy and reduce errors. Further increasing the number of trees to 500 and 1000 shows only marginal improvement in R^2 values, with the test sample reaching 0.96 and the validation dataset 0.92, indicating diminishing returns in performance gains.

Similarly, with max_depth set to 50, the results show stable performance with R^2 values around 0.94–0.95 for the test sample and 0.81–0.84 for the check data, with MSE and MAE remaining relatively stable across different tree quantities. This suggests that increasing tree depth can improve model performance up to a certain point, after which gains are minimal.

Therefore, the optimal performance of the gradient boosting model is observed with a maximum tree depth of 5 and n_estimators set to 100, achieving high accuracy with an R^2 value of 0.95 for the test sample and 0.93 for the check data, as well as low errors with MSE 8164.95 and MAE 58.61. Further increasing the number of trees beyond this point yields only slight performance improvements, indicating the model fully utilizes its capabilities with this configuration.

Analyzing all three models, we came to the following conclusions: decision tree provides a simple and interpretable model, suitable for tasks where the explainability of decisions is important. However, it is prone to overfitting and less robust to data changes. Random forest improves accuracy and robustness by using an ensemble of trees but requires more computational resources and is less interpretable. Gradient boosting offers the highest accuracy and the ability to handle complex data but at the cost of high computational expense and the need for careful tuning. Therefore, the choice of model depends on the specific requirements of the task: decision tree is suitable for quick and interpretable solutions, random forest for balancing accuracy and robustness, and gradient boosting for maximum accuracy on complex data.

5. Conclusions

This study conducted research on forecasting traffic flow at the intersection of Bogishamol Street in Tashkent, Uzbekistan. Various machine learning methods were used, including decision trees, random forest, and gradient boosting. The results showed that the gradient boosting model exhibited the best performance in terms of the coefficient of determination (R^2) and mean squared error (MSE).

An important result of the study was the identification that simple machine learning models, such as decision trees, can provide good results in forecasting traffic flow. This confirms the thesis that in machine learning practice, model simplicity is often rethought as complexity.

The results of this study can be useful for the development of intelligent traffic management systems, which in turn can contribute to improving road and intersection capacity, as well as reducing environmental problems and economic costs.

In the future, it is planned to expand this research by including data from other directions at the intersection in the analysis, as well as considering other methods of forecasting traffic flow to achieve more accurate results.

Author Contributions: Conceptualization, M.R.; methodology, M.R. and T.T.; software, K.T.; validation, M.R., T.T. and K.T.; formal analysis, M.R. and K.T.; investigation, K.T.; resources, K.T.; writing—original draft preparation, T.T.; writing—review and editing, M.R. and K.T.; visualization, K.T.; supervision, T.T. and K.T.; project administration, M.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used to support the findings of this study are available from the corresponding author upon request or via the link https://github.com/Boobur/Forecast-traffic-flow.git (accessed on 28 July 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhang, L.; Yang, Y.; Deng, Y.; Kang, H. Forecasting of Road Traffic Flow Based on Harris Hawk Optimization and XGBoost. *J. Adv. Math. Comput. Sci.* **2022**, *37*, 21–29. [CrossRef]
2. Zhu, L.; Shu, S.; Zou, L. XGBoost-Based Travel Time Prediction between Bus Stations and Analysis of Influencing Factors. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, e3504704. [CrossRef]

3. Zhang, X.; Zhang, Q. Short-Term Traffic Flow Prediction Based on LSTM-XGBoost Combination Model. *Comput. Model. Eng. Sci.* **2020**, *125*, 95–109. [CrossRef]
4. Bokaba, T.; Doorsamy, W.; Paul, B.S. A Comparative Study of Ensemble Models for Predicting Road Traffic Congestion. *Appl. Sci.* **2022**, *12*, 1337. [CrossRef]
5. Cao, D.; Wu, J.; Wu, J.; Kulcsár, B.; Qu, X. A Platoon Regulation Algorithm to Improve the Traffic Performance of Highway Work Zones. *Comput. Aided Civ. Infrastruct. Eng.* **2021**, *36*, 941–956. [CrossRef]
6. Lu, W.; Rui, Y.; Yi, Z.; Ran, B.; Gu, Y. A Hybrid Model for Lane-Level Traffic Flow Forecasting Based on Complete Ensemble Empirical Mode Decomposition and Extreme Gradient Boosting. *IEEE Access* **2020**, *8*, 42042–42054. [CrossRef]
7. Jia, X.; Zhou, W.; Li, S.; Chen, X. Combined Prediction of Short-Term Travel Time of Expressway Based on CEEMDAN Decomposition. *IEEE Access* **2022**, *10*, 96873–96885. [CrossRef]
8. Zheng, J.; Huang, M. Traffic Flow Forecast Through Time Series Analysis Based on Deep Learning. *IEEE Access* **2020**, *8*, 82562–82570. [CrossRef]
9. Wang, Y.; Jia, R.; Dai, F.; Ye, Y. Traffic Flow Prediction Method Based on Seasonal Characteristics and SARIMA-NAR Model. *Appl. Sci.* **2022**, *12*, 2190. [CrossRef]
10. Giraka, O.; Selvaraj, V.K. Short-Term Prediction of Intersection Turning Volume Using Seasonal ARIMA Model. *Transp. Lett.* **2020**, *12*, 483–490. [CrossRef]
11. Deretić, N.; Stanimirović, D.; Awadh, M.A.; Vujanović, N.; Djukić, A. SARIMA Modelling Approach for Forecasting of Traffic Accidents. *Sustainability* **2022**, *14*, 4403. [CrossRef]
12. Liu, Y.; Zhang, N.; Luo, X.; Yang, M. Traffic Flow Forecasting Analysis Based on Two Methods. *J. Phys. Conf. Ser.* **2021**, *1861*, 012042. [CrossRef]
13. Gacar, B.K.; Kocakoç, İ.D. Regression Analyses or Decision Trees? *Manisa Celal Bayar Üniversitesi Sos. Bilim. Derg.* **2020**, *18*, 251–260. [CrossRef]
14. Sagi, O.; Rokach, L. Approximating XGBoost with an Interpretable Decision Tree. *Inf. Sci.* **2021**, *572*, 522–542. [CrossRef]
15. Irawan, K.; Yusuf, R.; Prihatmanto, A.S. A Survey on Traffic Flow Prediction Methods. In Proceedings of the 2020 6th International Conference on Interactive Digital Media (ICIDM), Bandung, Indonesia, 14 December 2020; pp. 1–4.
16. Supriyatin, W.; Rianto, Y. Comparative Analysis Accuracy ID3 Algorithm and C4.5 Algorithm in Selection of Candidates Basic Physics Laboratory Assistant. Komputasi. *J. Ilm. Ilmu Komput. Dan Mat.* **2024**, *21*, 1–14. [CrossRef]
17. Abedinia, A.; Seydi, V. Building Semi-Supervised Decision Trees with Semi-Cart Algorithm. *Int. J. Mach. Learn. Cyber.* **2024**, *15*, 1–18. [CrossRef]
18. Momeni Kho, S.; Pahlavani, P.; Bigdeli, B. Analyzing and Predicting Fatal Road Traffic Crash Severity Using Tree-Based Classification Algorithms. *Int. J. Transp. Eng.* **2022**, *9*, 635–652. [CrossRef]
19. Gan, J.; Li, L.-H.; Zhang, D.; Yi, Z.; Xiang, Q. An Alternative Method for Traffic Accident Severity Prediction: Using Deep Forests Algorithm. *J. Adv. Transp.* **2020**, *2020*, 1–13. [CrossRef]
20. Yan, M.; Shen, Y. Traffic Accident Severity Prediction Based on Random Forest. *Sustainability* **2022**, *14*, 1729. [CrossRef]
21. Machoke, M.; Mbelwa, J.; Agbinya, J.; Sam, A.E. Performance Comparison of Ensemble Learning and Supervised Algorithms in Classifying Multi-Label Network Traffic Flow. *Eng. Technol. Appl. Sci. Res.* **2022**, *12*, 8667–8674. [CrossRef]
22. Li, Z.; Wang, L.; Wang, D.; Yin, M.; Huang, Y. Short-Term Traffic-Flow Forecasting Based on an Integrated Model Combining Bagging and Stacking Considering Weight Coefficient. *Electronics* **2022**, *11*, 1467. [CrossRef]
23. Chen, C.-M.; Liang, C.-C.; Chu, C.-P. Long-Term Travel Time Prediction Using Gradient Boosting. *J. Intell. Transp. Syst.* **2020**, *24*, 109–124. [CrossRef]
24. Han, S.; Kim, H. Optimal Feature Set Size in Random Forest Regression. *Appl. Sci.* **2021**, *11*, 3428. [CrossRef]
25. Zhan, X.; Zhang, S.; Szeto, W.Y.; Chen, X. (Michael) Multi-Step-Ahead Traffic Speed Forecasting Using Multi-Output Gradient Boosting Regression Tree. *J. Intell. Transp. Syst.* **2020**, *24*, 125–141. [CrossRef]
26. Yaman, M.A.; Rattay, F.; Subasi, A. Comparison of Bagging and Boosting Ensemble Machine Learning Methods for Face Recognition. *Procedia Comput. Sci.* **2021**, *194*, 202–209. [CrossRef]
27. Navarro-Espinoza, A.; López-Bonilla, O.R.; García-Guerrero, E.E.; Tlelo-Cuautle, E.; López-Mancilla, D.; Hernández-Mejía, C.; Inzunza-González, E. Traffic Flow Prediction for Smart Traffic Lights Using Machine Learning Algorithms. *Technologies* **2022**, *10*, 5. [CrossRef]
28. Moumen, I.; Abouchabaka, J.; Najat, R. Adaptive Traffic Lights Based on Traffic Flow Prediction Using Machine Learning Models. *Int. J. Electr. Comput. Eng.* **2023**, *13*, 5813–5823. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.