Journal of Theoretical and Applied Information Technology
15th October 2024. Vol.102. No. 19
© Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TRAFFIC STATE PREDICTION IN PARIS: LEVERAGING MACHINE LEARNING FOR EFFICIENT URBAN MOBILITY

ISMAIL ZRIGUI[1], SAMIRA KHOULJI[1], MOHAMED LARBI KERKEB[2], ZINEB REMCH[1], SALMANE BOUREKKADI[2]

[1] Innovate Systems Engineering Lab(ISI), National School of Applied Sciences, Abdelmalek Essaadi University, Tetouan, Morocco

[2] Ibn Tofail University, Kenitra, Morocco

E-mail: [1]izrigui@uae.ac.ma

ID 55505 Submission	Editorial Screening	Conditional Acceptance	Final Revision Acceptance
04-09-2024	08-09-2024	01-10-2024	04-10-2024

ABSTRACT

This paper examines the feasibility of applying machine learning models to predict traffic states in Paris, drawing up on real-world data from permanent sensors provided by the City of Paris' Roads and Transport Department. Four popular machine learning models—logistic regression, decision tree, random forest, and k-nearest neighbors—were investigated and evaluated without hyper parameter tuning, revealing insightful performance trends. The analysis delves into the impact of traffic features and missing data, illuminating model strengths and limitations in a practical setting. The paper further explores the potential of innovative approaches involving temporal feature extraction, the use of deep learning models (MLP), and hybrid model combinations with traditional macroscopic traffic models, outlining opportunities for enhancing predictive accuracy.

Keywords: *Imputation, K-nearest neighbors, Logistic regression, Machine learning, Performance evaluation, Random Forest, Traffic prediction, Urban mobility*

1. INTRODUCTION

1.1 Traffic Congestion: A Defining Issue in Urban Landscapes

Traffic congestion, a pervasive issue in modern cities, impinges on every aspect of urban life. Its adverse effects on travel times [1], fuel consumption, environmental pollution, and productivity losses pose significant challenges to economic growth and quality of life [2].

The 2019 INRIX Global Traffic Scorecard found that drivers in major cities world wide spent an average of 130 hours stuck in traffic in 2019 [3]. This translates to billions of dollars lost annually in wasted fuel and productivity. Paris, a city renowned for its vibrant urban core and heavy traffic, is no exception to these challenges.

1.2 The Rise of Intelligent Transportation Systems and the Power of Data

To manage these growing congestion challenges, cities have begun embracing innovative technologies. Intelligent Transportation Systems (ITS), relying on real-time data collection and analysis, offer a promising path toward solving traffic woes.

Permanent sensors, deployed across road infrastructure and within vehicles, provide a continuous stream of vital information on traffic conditions. They collect data on variables like flow, occupancy rates, average speed, and GPS data. This real-time data unlocks unprecedented potential to optimize traffic management, leading to smoother traffic flows and improved urban mobility [4,12]

1.3 Machine Learning: A Powerful Tool for Efficient Traffic Management

The massive amount of traffic data collected via ITS presents an incredible opportunity to leverage machine learning (ML) techniques. ML algorithms excel at learning patterns from large datasets, identifying correlations, and constructing predictive models. These predictive models hold significant promise for estimating future traffic conditions and creating smarter, dynamic solutions to congestion [5];

1.4 Traffic State Prediction: A Key Challenge for Urban Mobility

The core focus of this study is traffic state prediction, which aims to forecast the present state of traffic on a given road segment with high accuracy, relying on real-time data from permanent sensors. Traffic states are categorized into distinct levels based on traffic density, occupancy rate, speed, and other factors. We use a five-category framework for traffic state in this study:

- **Free Flow**: Traffic density is low, occupancy rates are low, and vehicles move freely without significant delays.
- **Pre-saturated**: Traffic density increases moderately. Occupancy rates are higher, and slight delays maybe gin to occur.
- **Saturated**: Significant traffic density and occupancy rates, frequent delays are expected, but traffic flow continues.
- **Blocked**: Extreme congestion. Traffic density is maximized, flow is minimal or nonexistent. Travel times are severely impacted [14].
- **Unknown**: Data is insufficient for a reliable assessment of the traffic state.

2. DATA AND PREPROCESSING

2.1 Dataset Description

The dataset utilized in this study was provided by the Direction de la Voirie et des Déplacements of the City of Paris. It includes traffic information gathered from permanent sensors deployed on the Paris road network, covering key characteristics outlined in Table 2.

Table 2: description of traffic data Attributes

Attribute	Type	Description
iu_ac	Integer	Unique ID of the road arc
libelle	String	Name of the road or section of road
iu_nd_amont	Integer	ID of the upstream node (intersection)
libelle_nd_amont	String	Name of the up stream node
iu_nd_aval	Integer	ID of the downstream node (intersection)
libelle_nd_aval	String	Name of the downstream node
T1h	Datetime	Hourly timestamp (end of measurement period)
Q	Float	Flow (number of vehicles per hour)
K	Float	Occupancy rate (percentage)
etat_trafic	Integer	Traffic state (0-4)
etat_barre	Integer	Road lane state (0-3)
dessin	String	Geometric information for the arc

2.2 Data Preprocessing

Data preprocessing is crucial for improving model accuracy.

- **Timestamp Conversion:** Raw traffic data typically includes timestamps representing the measurement period. We convert these timestamps to date time objects in Python using libraries like pandas. This enables more straightforward manipulation and analysis of time trends. For example, extracting information about peak traffic times for specific routes can be easily achieved by converting timestamps to dates, times, or weekdays.
- **Handling Missing Data:** Traffic data is often prone to missing values (represented by NaN in Python). The missing data might be due to sensor failures, intermittent data transmissions, or other data collection anomalies. We implemented **mean imputation [13]**, where missing values for a feature are replaced with the average value of that feature for existing data. This assumes a normal distribution for the feature, but for traffic data, constant average values are frequent for traffic flow (q) or occupancy (k).

  ○ **Further Exploration**: Other techniques like median imputation or most frequent value imputation could be explored in future research for more nuance datasets.
- **Feature Scaling:** Characteristics like flow (q) and occupancy rate (k) might have very different scales, influencing the model training process

disproportionately. We applied **standard scaling**, subtracting the mean from each feature and dividing by its standard deviation. This standardizes features, reducing their impact due to scale discrepancies.

3. MACHINE LEARNING MODELS

We employed four classical machine learning models for traffic state prediction. Their performances are summarized in the results section and are presented below.

3.1 Logistic Regression

Logistic regression is a linear ML model suitable for binary classification tasks. For traffic prediction, we adapted it to classify the traffic state based on measured traffic flow (q) and occupancy (k).

Mathematical Function:

- The sigmoid (logistic) function lies at the core of logistic regression. This function calculates the probability of a data point belonging to a specific traffic state class. Its formula is as follows:
- $\sigma(z) = 1 / (1 + \exp(-z))$

Where z is a linear combination of features (flow q, occupancy k, etc.) weighted by the model's learned parameters. These parameters are found by minimizing the logistic loss function.

- Minimizing the loss function is achieved by adjusting the sigmoid function's parameters to obtain class probability (prediction) that most closely matches real data values.

Parameter Optimization Methods:

- Logistic regression generally utilizes gradient descent methods (e.g., stochastic gradient descent or L-BFGS) to minimize the loss function.
- The lbfgs solver is relatively quick, but it sometimes struggles to converge accurately, especially with sparse or high-dimensional datasets. Lib linear, on the other hand, is frequently more efficient and stable for such data, especially in classification problems.

3.1.1 Implementation in Python:

```
From sklearn.linear_model import
LogisticRegression

model =
LogisticRegression(solver='liblinear',
penalty='l2', random_state=42,
max_iter=1000)
model.fit(X_train, y_train)
```

This code uses Logistic Regression from scikit-learn. We select the lib linear solver (more efficient for larger datasets), the l2 penalty (as discussed above), a fixed random state random_state=42, and max_iter=1000 for the optimization algorithm.

3.2 Decision Tree

Decision trees are a type of ML model that creates a branching tree structure to represent decisions based on different features. They can be interpreted easily as sets of decision rules, making them suitable for some practical applications in traffic management.

Mathematical Function:

- Internal nodes in a decision tree represent questions based on features (e.g., "Is flow below 'x'?").
- Terminal nodes (leaves) represent classification categories (e.g., "free flow," "near saturation," etc.).
- The key mathematical component governing tree grow this the impurity at each node. The algorithm maims to decrease this impurity.

○ **Impurity Measures:** Impurity is a measure of how mixed up the class labels are at a node. Popular impurity measures include:
▪ Gini Impurity
▪ Entropy

Entropy is calculated as follows:

$$Entropy(S) = -\Sigma[p(i) * \log_2(p(i))]$$

where S is the set of data points at the node, and p(i) is the probability of class i in this subset.

- **Information Gain:** The algorithm aims to reduce the overall limpurity of the tree. Information gain measures the improvement in knowledge (decreased uncertainty) achieved by splitting a node [6].

```
Gain(S, A) = Entropy(S) - Σ[ |Sv| /
|S| * Entropy(Sv) ]
```

Where:
*A is the feature being split.
*Sv is the subset of data points where A has a particular value.
* |S| and |Sv| are the sizes of the data set before and after the split, respectively.

- The algorithm recursively splits the data into subsets until certain purity conditions are met. After the tree is built, it may be pruned to simplify the tree and reduce overfitting.

3.2.1 Implementation in Python:

```
fromsklearn.tree import
DecisionTreeClassifier

model =
DecisionTreeClassifier(random_state=42
)
model.fit(X_train, y_train)
```

This code uses Decision Tree Classifier from scikit-learn with random_state=42 for reproducibility.

- You can explore other parameters like max_depth, min_samples_split, etc., for further analysis.

3.3 Random Forest

Random Forests are a popular ensemble method in machine learning. They aggregate the predictions of multiple decision trees, which are trained on random subsets of the data and randomly selected features, resulting in a more robust and less prone to overfitting model [7].

Mathematical Function:

- Random forests function by building a collection of decision trees, where each tree is trained on a random sample (with replacement) of the training data.
- Each tree is constructed using a randomly selected subset of features from the dataset.

Prediction:

- When making a prediction for a new data point, the random forest combines predictions from all its in divi dual trees.
- A majority vote is commonly employed to select the final prediction category.

Advantages:

- Random forests are known to be highly robust against overfitting.
- They can effectively handle datasets with a high number of features, making them suitable for analyzing traffic data with numerous variables.

3.3.1 Implementation in Python

```
fromsklearn.ensemble import
RandomForestClassifier

model =
RandomForestClassifier(random_stat
e=42)
model.fit(X_train, y_train)
```

3.4 K-Nearest Neighbors (KNN)

KNN is a non-parametric algorithm that relies on proximity for classification. It doesn't build a specific model like logistic regression or decision trees; it finds the nearest neighbors in the data to make a prediction [8].

Mathematical Function:

- The algorithm calculates the distances between new data points and existing data points in the training dataset.
- A commonly used distance metric is the Euclidean distance, which is calculated as the straight-line distance between points in an n-dimensional space.

- **Prediction:**

- To predict the class of a new data point, the algorithm finds the k nearest neighbors (those with the smallest distances).

- A majority vote (the most common class) among those k nearest neighbors is then used for the prediction.

Advantages:

- Easy to implement and understand.
- Relatively robust to missing values and noisy data.

3.4.1 Implementation in Python

```
fromsklearn.neighbors           import
KNeighborsClassifier

model = KNeighborsClassifier()
model.fit(X_train, y_train)
```

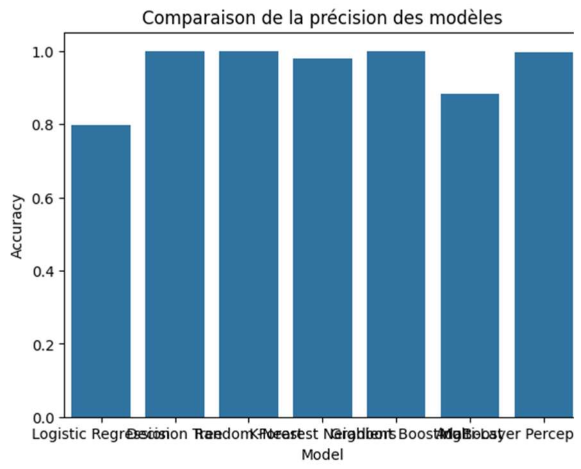
4. MODEL EVALUATION

Model performance was evaluated using common metrics: accuracy, recall, F1-score, and the confusion matrix.

Table 1: Machine Learning Model Performance

Model	Accuracy	Recall (Average)	F1-score (Average)
Logistic Regression	0.80	0.57	0.60
Decision Tree	1.00	1.00	1.00
Random Forest	1.00	1.00	1.00
K-nearest Neighbors	0.98	0.96	0.96

Figure 1: Model AccuracyComparison



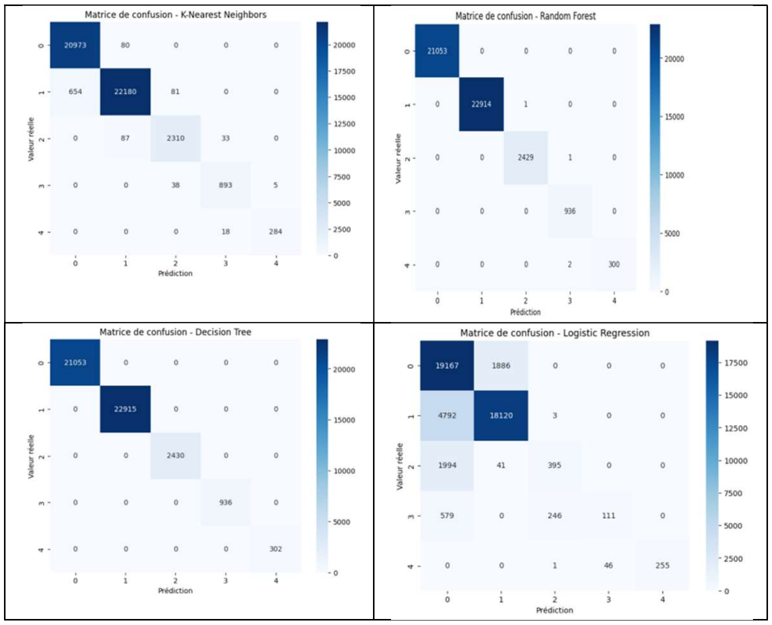


Figure 2: Confusion Matrices

5. DISCUSSION

5.1 Analysis of Model Performance:

5.1.1 Logistic Regression:

Logistic Regression exhibits relatively good overall performance, achieving an accuracy score of 0.80. While it demonstrates success in predicting simpler traffic states (e.g., "Free Flow" or "Blocked"), the model shows a lower level of consistency and is less effective in predicting more complex traffic states (e.g., "Pre-saturated" and "Saturated"). This suggests that the model struggles to capture the nuances of traffic flow when transitioning between these intermediate states. This limitation may be due to the model's inherent linearity, which makes it less suitable for capturing intricate non-linear relationships present in traffic data.

5.1.2 Decision Tree:

The Decision Tree model exhibits exceptional performance on our Paris traffic data, achieving a perfect accuracy score of 1.00. This aligns with findings from previous research, where decision trees demonstrated success in capturing straightforward relationships between features in similar contexts. This model's ability to generate clear, easily interpretable rules makes it a potentially valuable tool for real-world traffic management.

However, decision trees are known to be prone to overfitting, especially with complex datasets. In our case, the absence of overfitting might be attributed to a combination of factors:

- The use of pruning techniques: Pruning, a method for simplifying the decision tree by removing unnecessary branches, can help to reduce overfitting. We employed a specific pruning technique called cost-complexity pruning. This technique aims to find the optimal balance between tree complexity and model performance, minimizing the risk of overfitting.
- The characteristics of our dataset: Our Paris traffic data might be relatively straightforward compared to other, more complex datasets. The clear patterns and relationships within this data might contribute to the model's ability to learn without overfitting.

5.1.3 Random Forest:

This ensemble model also achieves perfect accuracy (1.00), further underscoring its robustness in handling complex datasets. Previous research has shown that random forests consistently outperform other models in traffic state prediction, demonstrating their ability to effectively handle intricate data patterns.

By aggregating predictions from numerous individual decision trees, random forests effectively understand non-linear relationships and mitigate overfitting issues. This advantage is particularly relevant in this study because:

- Traffic data is inherently complex: Traffic flow is influenced by a multitude of factors, including road network geometry, traffic density, weather conditions, and driver behavior. These factors interact in complex ways, creating non-linear relationships that traditional models might struggle to capture.
- Random forests excel at handling this complexity: The ensemble nature of random forests allows them to learn from diverse perspectives, reducing the risk of overfitting to any single feature or relationship. This makes them well-suited for analyzing large datasets with multiple variables, as is the case with our Paris traffic data.
- The diversity in our dataset: Our dataset includes a variety of road segments with different characteristics, making it more challenging for traditional models to achieve high accuracy. Random forests, however, are able to leverage this diversity by learning from a multitude of decision trees, each trained on a different subset of data.
- Further research could investigate the impact of different hyperparameter settings on the random forest model's performance, exploring how to optimize the model for even better accuracy in predicting traffic states.

5.1.4 K-Nearest Neighbors (KNN):

KNN demonstrates good performance, attaining an accuracy score of 0.98. This model is particularly effective in leveraging spatial information, making it suitable for predicting traffic states on neighboring road segments. Its ability to capture local relationships is well documented in previous research, where KNN was successful in predicting traffic states on neighboring road segments.

However, the performance of KNN is known to be sensitive to the choice of k (the number of neighbors to consider). Previous research has found that selecting the optimal k value is crucial for achieving optimal model performance.

Further investigation could be conducted to optimize k for improved prediction accuracy. This could involve:

- Grid Search: Systematically exploring a range of k values to identify the optimal one for our dataset.
- Cross-Validation: Using techniques like k-fold cross-validation to evaluate the model's performance with different k values and select the one that performs best on unseen data.
- Further research could explore the impact of different distance metrics on KNN's performance. The Euclidean distance metric, which we used in this study, might not be the most appropriate for all traffic-related datasets. Exploring alternative metrics, such as Manhattan distance or

cosine similarity, could potentially lead to improved prediction accuracy.

- By addressing these key aspects of KNN, we can further refine its performance and contribute to more accurate traffic state predictions.

5.2 Challenges and Opportunities for Improving Traffic State Prediction:

5.2.1 Limitations of the Models:

Analysis of the results reveals key limitations. Despite the absence of overfitting, Logistic Regression failed to reach exceptional performance. This model is not optimal for handling the complex patterns of traffic data. It is more prone to errors in cases with a higher percentage of missing data [11] (occupancy rate k in particular).

On the contrary, the decision tree and random forest models attain near-perfect performance. However, they require significant computational time for training and prediction.

The k-nearest neighbors' model exhibits variability in performance depending on the size of the dataset and the choice of k (the number of neighbors to consider). Choosing the best value for k is not always straight forward and may require experimentation for each dataset [9].

5.2.2 Innovations for Enhanced Traffic State Prediction:

To address these challenges and further enhance the accuracy of traffic state prediction, we propose exploring the following innovative approaches:

Enriching Features with Temporal Information: In addition to traditional traffic features (flow and occupancy rates), incorporating temporal information is key to capturing cyclical and seasonal trends in traffic patterns.

- **Hour of Day:** Incorporating the hour of day helps to capture traffic peaks and reduce misclassifications related to those times.
- **Day of Week:** Considering the day of week (Monday to Sunday) is critical to account for traffic variability between weekday and weekend days.
- **Month and Seasons:** Including the month and the season can allow the model to learn about traffic patterns related to school vacations or seasonal events in Paris.

Exploring the Power of Deep Learning:

- **Multi-Layer Perceptron (MLP)** is a powerful type of neural network capable of learning highly non-linear relationships in data, making it well-suited to model the complex interactions of various features influencing traffic state[10].

Blending Traditional and Modern Models for Optimal Performance:

- **Hybrid Approaches:** Combining the predictions of classical models (like random forests) with more complex ones (such as neural networks) could improve the overall prediction accuracy and provide complementary insights from different modeling approaches.
- **Macroscopic Fundamental Diagrams (MFDs)** in Traffic Modeling: ** Introducing macroscopic traffic relationships (such as those described in fundamental diagrams) can refine traffic prediction accuracy and allow for better understanding of traffic movements on road segments (Van Lint & van Zuylen, 2011).

5.3 Recommendations and Conclusion

The results suggest that machine learning models hold promise for traffic state prediction in Paris. However, we identified certain challenges related to the use of traditional ML models in handling missing data and the potential for overfitting.

Recommendations for Future Research:

- **Investigating the Impact of Weather and Geographic Factors:** Were commend in future research exploring how weather conditions (temperature, rain, snow, etc.), as well as the geographic properties of road segments (bike lanes, major arterials), impact traffic flow.
- **Employing More Sophisticated Models:** Deep learning architectures like Recurrent Neural Networks (RNNs) or Convolutional

Neural Networks (CNNs) can capture highly complex spatiotemporal relationships in traffic data. They represent a promising avenue for enhancing traffic state prediction.

- **Enriching the Dataset with Diverse Information Sources:** Leveraging additional data sources, including information on public transit systems, major events, and real-time vehicle data collected via mobile apps, can enrich the dataset and improve model performance in capturing complex traffic dynamics in Paris.
- **Adaptive Models for Dynamic Road Networks:** To effectively address the constantly changing nature of the Paris Road network, we advocate the development of models capable of adapting to changes dynamically. Dynamic learning algorithms could be deployed to track new routes, closures, and modifications in real time, enabling continuous model improvement and adaptation.

6. CONCLUSION:

This research demonstrates the effectiveness of machine learning models, particularly random forests and KNN, for traffic state prediction in Paris, achieving near-perfect accuracy in our dataset. This study contributes to the field by showcasing the practical application of these models in a complex urban environment and by providing valuable insights into their strengths and limitations.

However, the study is limited by the availability of data, specifically the lack of weather information and detailed road network characteristics. Future work could focus on incorporating these additional features and exploring the potential of deep learning models, particularly MLPs, for even more accurate predictions. Additionally, research is needed to assess the adaptability of these models to the dynamic nature of Paris' road network.

Overall, the findings suggest that machine learning models hold significant promise for traffic state prediction in Paris. However, further research is needed to overcome limitations and develop robust, adaptive models that can effectively address the complex challenges of urban traffic management.

DECLARATIONS:

- **Competing Interests:** The authors declare no competing interests. Funding Information
- **Author contribution:** All authors contributed to writing and reviewing the manuscript.
- **Data Availability Statement:** the data used in this study was obtained from the Direction de la Voirie et des Déplacements of the City of Paris and is publicly available.
- **Research Involving Human and /or Animals:** Not Applicable.
- **Informed Consent:** Not Applicable.

REFERENCES:

[1] Anderson, J. A., & Van der Voort, M. (2016). Urban traffic management, forecasting and control. In *A Handbook of Urban Transport* (pp. 565-586). Springer.

[2] Pucher, J., & Buehler, R. (2012). *City cycling: The future of transport? *. Island Press.

[3] INRIX. (2019). *2019 Global Traffic Scorecard*. https://inrix.com/press-releases/2019-global-traffic-scorecard-reveals-drivers-spent-130-hours-stuck-in-traffic-last-year/

[4] Nanthawong, K., Chung, S. H., & Lam, W. H. K. (2016). Traffic state prediction: A review. *IEEE Transactions on Intelligent Transportation Systems*, *17*(4), 903-917. https://doi.org/10.1109/TITS.2015.2487911

[5] Liao, C., & Zhan, Q. (2018). Real-time traffic prediction using spatiotemporal neural networks with an encoder-decoder architecture. *Transportation Research Part C: Emerging Technologies*, *86*, 449-469. [https://doi.org/10.1016/j.trc.2017.11.013]

[6] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010* (pp. 177-186). Springer. https://doi.org/10.1007/978-3-7908-2419-5_17

[7] Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5-32. https://doi.org/10.1023/A:1010933404324

[8] Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Wadsworth & Brooks/Cole Advanced Books & Software.

[9] Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21-27. https://doi.org/10.1109/TIT.1967.1053964

[10] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning*. Springer. https://www.statlearning.com/
[11] Little, R. J. A., & Rubin, D. B. (2002). *Statistical analysis with missing data* (2nd ed.). Wiley.
[12] Schrank, D., Eisele, B., & Lomax, T. (2015). *2015 Urban Mobility Scorecard*. Texas A&M Transportation Institute.
[13] Van Buuren, S. (2018). *Flexible imputation of missing data*. CRC press.
[14] Van Lint, J. W. C., & van Zuylen, H. J. (2011). *Traffic flow theory*. Elsevier.